CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

The objective of this dissertation is to study the routing algorithms in wireless sensor networks. The performance of routing algorithm can be evaluated on the basis of energy × delay metrics, because sensors have limited battery power. It will also compare the performance of Fibonacci series based energy aware algorithm, PEGASIS (Power-Efficient GAthering in sensor information systems) and Chain based Binary scheme.

1.2 BACKGROUND

Due to the explosive growth in cellular subscriber over the past couple of years, users want the flexibility and mobility that come with wireless networks. At the same time device technology has improved to the point where today's devices are smaller, cheaper, and more energy efficient than in the past. Such portable devices with wireless networking capability allow us to get closer to the goal of "anytime, any where" connectivity to voice, video and data services.

The wireless channel presents several networking challenges not found in the wired domain, due to the limited resources, of the channel and the portable devices.

The challenges in designing network protocols therefore are to overcome these limitations.

- **Limited channel bandwidth**

  Authorities regulate what Bandwidth particular networks can access and with how much power nodes are allowed to transmit. This limits the amount of BW that can be given to each user of the network, requiring band width efficient protocols to maximize utility of the network

- **Limited node energy**

  Devices that access the wireless channel are often portable, obtaining, energy from a local a battery .This limits the amount of energy available to the node, affecting the lifetime of the networks .Protocols should therefore try to minimize energy dissipation to maximize node lifetime.

- **Electromagnetic wave propagation**

  The radio wave is scattered as it propagates through the environment. Therefore the power in the wave at the receiver (and hence the signal to noise ratio (SNR) depends on the distance between the transmitter and receiver. This precludes the use of collision detection to determine if two nodes are trying to access the channel at the same time .thus it is necessary to create clearly defined media access control (MAC) protocols to minimize collisions.

- **Error prone channel**

  Errors can be caused by environment, e.g. when buildings, cars block a direct, line of sight path between the transmitter and receiver. So collision can occur between messages from different nodes.

- **Time varying conditions**

  The errors on the channel will vary over time, as the environment in which the transmitter and receivers are located changes and different nodes begin and end their own transmissions.

- **Mobile nodes:** Node mobility creates routing difficulties as nodes move in and out of communication range with each other.

These are the general ideas that can be used to overcome these limitations. Low energy protocols will help extend the limited node energy. Finally, adaptive routing, MAC and link layer protocols can be used to overcome the time varying conditions of the wireless channels and node mobility.

1.3 PROBLEM STATEMENT

Sensor webs consisting of nodes with limited battery power and wireless communication are deployed to collect useful information from the field. Gathering sensed information in an energy efficient manner is critical to operating the sensor network for a long period of time. If each node transmits its sensed data directly to the base station, then it will deplete its power quickly. So the problem is to minimize the energy and it is also important to consider delay incurred in gathering sensed data. We have suggested a Fibonacci series based energy aware algorithm to optimize energy, energy × delay metric.

1.4 THE APPROACH

The work started with a requirement specification that were quite unspecific so the initial weeks were used to try out what could be done and what couldn’t. Since it was hard to find an obvious solution to the optimization problem at the start, the implementation part didn’t get so extensive. The work in the beginning was also of mixed characteristics, literature studying and experimental implementation. However at the end implementation part was finished.

CHAPTER 2

WIRELESS SENSOR NETWORKS

2.1 OVERVIEW OF WIRELESS SENSOR NETWORKS

A network is formed because we need things to be formed as collective or collaborative way so that all parts of the network share the work load in distributive way. So the main stress will be in reference to the communication field.

Wireless network communication area has been in limelight in recent years and will continue to be the center of attraction because of its obvious advantages. But on the flipside of the advantages there are many limitations which invite some serious research work in this field. We also have different situation specific applications. So we need wireless protocols to, implement the specific needs.

A wireless sensor network is a collection of nodes organized into a cooperative network Each node consists of processing capability (one or more microcontrollers, CPUs or DSP chips), may contain multiple types of memory (program, data and flash memories), have a RF transceiver (usually with a single omni-directional antenna), have a power source (e.g., batteries and solar cells), and accommodate various sensors and actuators. The nodes communicate wirelessly and often self-organize after being deployed in an ad hoc fashion [5].

Currently, wireless sensor networks are beginning to be deployed at an accelerated pace. It is not unreasonable to expect that in 10-15 years that the world will be covered with wireless sensor networks with access to them via the Internet. This new technology is exciting with unlimited potential for numerous application areas including environmental, medical, military, transportation, entertainment, crisis management, homeland defense, and smart spaces [12].

Also the boundary associated with a node can not be well defined because it changes randomly as it depends upon many constraints like noise level and environmental factors.

One major problem is that the medium in which we are going to transmit the information is open air and hence, more prone to error. For example in wireless communication the bit error rates are of order 10^{-4} but in wired communication it is 10^{-9} (in fiber optic cables)$^{.}$

2.2 MAJOR CONSTRAINTS IN WIRELESS NETWORKS

While using wireless network some constraints are:

- It should be simple in use, like wired network communication.
- It should be generic i.e., it should be compatible with other related technologies.
- The security threats must be focused.
- The efficiency in power usage must be there.
- The quality of service (QoS) must be guaranteed.

2.3 CHARACTERISTICS OF SENSOR NETWORKS [19]

- The sensor nodes are small in size.
- The nodes are tough enough to be put in harsh climate condition (excessive pressure and temperature).
- The nodes have very limited in resources such as memory, computational power, communication range and most importantly battery power.
- The sensor networks are self-organizing and self-induced because there is rarely any human intervention. All nodes have to behave like in proactive manner.
- The deployment of sensor nodes is totally distributive in nature. The node density is thus varying at different places. Due to this reason one can find dense as well as sparse region in the same topology.
- Like other networks there is no scope of the hierarchical addressing or flat addressing in case of sensor networks. And due to this reason there is lack of any IP address or a single global identifier for a single node.

- The data which is to be communicated in the sensor network is very redundant because of large number of nodes to sense the same data.
- In order to maintain the connectivity nodes must be active even if they need not to be active for sensing data [11].

2.4 QoS CHALLENGES IN SENSOR NETWORKS

While sensor networks inherit most of the QoS issues from the general wireless networks, Their characteristics pose unique challenges. The following is an outline of design considerations for handling QoS traffic in wireless sensor networks [8]:

2.4.1 Bandwidth Limitation

A typical issue for general wireless networks is securing the bandwidth needed for achieving the required QoS. Bandwidth limitation is going to be a more pressing issue for wireless sensor networks. Traffic in sensor networks can be burst with a mixture of real-time and non-real-time traffic. Dedicating available bandwidth solely to QoS traffic will not be acceptable. In addition, simultaneously using multiple independent routes will be sometime needed to split the traffic and allow for meeting the QoS requirements. Setting up independent routes for the same flow can be very complex and challenging in sensor networks due energy constraints, limited computational resources and potential increase in collisions among the transmission of sensors.

2.4.2 Removal of Redundancy

As mentioned in section 2.3, sensor networks are characterized with high redundancy in the generated data. For unconstrained traffic, elimination of redundant data messages is somewhat easy since simple aggregation functions would suffice. However, conducting data aggregation for QoS traffic is much more complex. Comparison of images and video streams is not computationally trivial and can consume significant energy resources. A combination of system and sensor level rules would be necessary to make aggregation of QoS data computationally feasible. For example, data aggregation of imaging data can be selectively performed for traffic generated by sensors pointing to same direction since the images may be very similar. Another factor of consideration is the amount of QoS traffic at a particular moment. For low traffic it may be more efficient to cease data aggregation since the overhead would become dominant. Despite the complexity of data aggregation of imaging and video data, it can be very rewarding from a network performance point-of-view given the size of the data and the frequency of the transmission.

2.4.3 Energy and Delay trade-off

Since the transmission power of radio is proportional to the distance squared or even higher order in noisy environments or in the non-flat terrain [14], the use of multi-hop routing is almost a standard in wireless sensor networks. Although the increase in the number of hops dramatically reduces the energy consumed for data collection, the accumulative packet delay magnifies. Since packet queuing delay dominates its propagation delay, the increase in the number of hops can, not only slow down packet delivery but also complicate the analysis and the handling of delay-constrained traffic. Therefore, it is expected that QoS routing of sensor data would have to sacrifice energy efficiency to meet delivery requirements. In addition, redundant routing of data may be unavoidable to cope with the typical high error rate in wireless communication, further complicating the trade-off between energy consumption and delay of packet delivery.

2.4.4 Buffer size limitation

Sensor nodes are usually constrained in processing and storage capabilities. Multi-hop routing relies on intermediate relaying nodes for storing incoming packets for forwarding to the next hop. While a small buffer size can work, buffering of multiple packets has some advantages in wireless sensor networks. First, the transition of the radio circuitry between transmission and reception nodes consumes considerable energy [6] and thus it is advantageous to receive many packets prior to forwarding them. In addition, data aggregation and fusion involves multiple packets. Multi-hop routing of QoS data would typically require long sessions and buffering of even larger data, especially when the delay jitter is of interest. The buffer size limitation will increase the delay variation that packets incur while traveling on different routes and even on the same route. Such an issue will complicate medium access scheduling and make it difficult to meet QoS requirements.

2.4.5 Support of Multiple traffics

Inclusion of heterogeneous set of sensors raises multiple technical issues related to data routing. For instance, some applications might require a diverse mixture of sensors for monitoring temperature, pressure and humidity of the surrounding environment, detecting motion via acoustic signatures and capturing the image or video tracking of moving objects. These special sensors are either deployed independently or the functionality can be included on the normal sensors to be used on demand. Reading generated from these sensors can be at different rates, subject to diverse quality of service constraints and following multiple data delivery models. Therefore, such a heterogeneous environment makes data routing more challenging.

CHAPTER 3
WIRELESS SENSOR NETWORK MODELS AND DESIGN ISSUES

3.1 ARCHITECTURE DESIGN ISSUES

System Architecture and Design Issues Depending on the applications, different architectures and design goals/constraints have been considered for sensor networks. Since the performance of a routing and MAC protocols are closely related to the architectural model, in this section we try to capture architectural design issues and highlight their implications [8].

3.1.1 Network Dynamics

There are three main components in a sensor network. These are the sensor nodes, sink and monitored events. Aside from the very few setups that utilize mobile sensors, most of the network architectures assume that sensor nodes are stationary. On the other hand, supporting the mobility of sinks or cluster-heads is sometimes deemed necessary. Routing messages from or to moving nodes is more challenging since route stability becomes an important optimization factor, in addition to energy, bandwidth etc. The sensed event can be either dynamic or static depending on the application . For instance, in a target detection/tracking application, the event (phenomenon) is dynamic whereas forest monitoring for early fire prevention is an example of static events. Monitoring static events allows the network to work in a reactive mode, simply generating traffic when reporting. Dynamic events in most applications require periodic reporting and consequently generate significant traffic to be routed to the sink.

3.1.2 Node Deployment

Another consideration is the topological deployment of nodes. This is application dependent and affects the performance of the routing protocol. The deployment is either deterministic or self-organizing. In deterministic situations, the sensors are manually placed and data is routed through pre-determined paths. In addition, collision among the transmissions of the different nodes can be minimized through the pre-scheduling of medium access. However in self-organizing systems, the sensor nodes are scattered randomly creating an infrastructure in an ad hoc manner [5]. In that infrastructure, the position of the sink or the cluster-head is also crucial in terms of energy efficiency and performance [9]. When the distribution of nodes is not uniform, optimal clustering becomes a pressing issue to enable energy efficient network operation.

3.1.3 Ease of deployment

Sensor networks may contain hundreds or thousands of nodes, and they may need to be deployed in remote or dangerous environments allowing users to extract information in ways that would not have been possible otherwise. This requires that nodes be able to communicate with each other even in the absence of established network infrastructure and predefined node location.

3.1.4 Node Communications

During the creation of an infrastructure, the process of setting up the routes is greatly influenced by energy considerations. Since the transmission power of a wireless radio is proportional to distance squared or even higher order in the presence of obstacles [14], multi-hop routing will consume less energy than direct communication [1] [10]. However, multi-hop routing introduces significant overhead for topology management and medium access control. Direct routing would perform well enough if all the nodes were very close to the sink [15]. Most of the time sensors are scattered randomly over an area of interest and multi-hop routing becomes unavoidable. Arbitrating medium access in this case becomes cumbersome.

3.1.5 Data Delivery Models

Depending on the application of the sensor network, the data delivery model to the sink can be continuous, event-driven, query-driven and hybrid [16]. In the continuous delivery model, each sensor sends data periodically. In event-driven and query-driven models, the transmission of data is triggered when an event occurs or a query is generated by the sink. Some networks apply a hybrid model using a combination of continuous, event-driven and query-driven data delivery.

3.1.6 Node Capabilities

In a sensor network, different functionalities can be associated with the sensor nodes. In early work on sensor networks**,** all sensor nodes are assumed to be homogenous [17], having equal capacity in terms of computation, communication and power. However, depending on the application a node can be dedicated to a particular special function such as relaying, sensing and aggregation since engaging the three functionalities at the same time on a node might quickly drain the energy of that node. Some of the hierarchical protocols proposed in the literature designate a cluster-head different from the normal sensors. While some networks have picked cluster- heads from the deployed sensors [15], in other applications a cluster-head is more powerful than the sensor nodes in terms of energy, bandwidth and memory [11]. In such cases, the burden of transmission to the sink and aggregation is handled by the cluster-head.

3.1.7 Data Aggregation/Fusion

Since sensor nodes might generate significant redundant data, in some applications similar packets from multiple nodes can be aggregated so that the number of transmissions would be reduced. Data aggregation is the combination of data from different sources by using functions such as suppression (eliminating duplicates). Some of these functions can be performed either partially or fully in each sensor node, by allowing sensor nodes to conduct in-network data reduction. As computation would be less energy consuming than communication [16], substantial energy savings can be obtained through data aggregation. This technique has been used to achieve energy efficiency and traffic optimization in a number of routing protocols. In some network architectures, all aggregation functions are assigned to more powerful and specialized nodes. Data aggregation is also feasible through signal processing techniques. In that case, it is referred as data fusion where a node is capable of producing a more accurate signal by reducing the noise and using some techniques such as beam forming to combine the signals [16]. Data aggregation makes medium access control complex since redundant packets will be eliminated and such elimination will require instantaneous medium access arbitration. In such case, only CSMA and CDMA-based MAC protocols are typically applicable leading to an increase in energy consumption.

3.1.8 Latency

Data from sensor networks are typically time sensitive, so it is important to receive the data in a timely manner [2].

The above design issues are summarized in the Table 3.1;

Design Issue	Primary Factors
Network Dynamics	Mobility of node, target and sink
Node Deployment	Deterministic or ad hoc
Node Communication	Single-hop or multi-hop
Data Delivery Models	Continuous, event-driven, query-driven, or hybrid
Node Capabilities	Multi-or single function; homogeneous or heterogeneous capabilities
Data Aggregation/Fusion	In-network (partially or fully) or out-of-network
Latency	Synchronization

Table 3.1: Architectural Design Issues

3.2 NETWORK MODELS FOR SENSOR NETWORKS

There are basically four communication paradigms for dissemination and aggregation (routing, querying and discovery).in the next section we comparatively discuss the trade of these paradigms, how they deal with design challenges.

3.2.1 DATA CENTRIC

In this type of network model the sink node sends queries to certain WSN regions and waits for data from WNs located in the region selected. Because data are being requested through queries, attribute-based naming is necessary to specify the properties of data. Due to the large number of nodes deployed, in many WSNs it is not practical to assign global identifier to each node [13]. This, along with potential random deployment of WNs, makes it challenging to select specific WNs to be queried. Hence data are typically transmitted from every WN with in the deployment region; this gives rise, however, to significant redundancy along with inefficiencies in terms of energy consumption. It follows that it is desirable to have routing protocols that will be able to select a set of sensor nodes and utilize data aggregation during the relaying of data. This has led to the development of data centric routing. The following are the protocols which uses data centric network model:

- Sensor protocols for information via negotiation (SPIN)
- Rumor routing
- Directed diffusion
- Gradient-based routing (GBR)
- Constrained anisotropic diffusion routing (CADR)

3.2.2 HIERARCHICAL

A single-tier (cluster point) network can cause the cluster head node to become overloaded, particularly as the density of sensors increases. This, in turn, can cause latency in event status delivery. To permit WSNs to deal with a large population of WNs and to cover a large area of interest, multipoint clustering has been proposed [18]. The goal of hierarchical routing is to manage the energy consumption of WNs efficiently by establishing multihop communication within a particular cluster [10], and by performing data aggregation and fusion to decrease the number of transmitted packets to the sink. The following are the protocols which uses hierarchical network model [1]:

- Energy-adaptive clustering hierarchy (LEACH)
- Power-efficient gathering in sensor information systems (PEGASIS)
- Threshold-sensitive energy-efficient sensor network protocol (TEEN)
- Adaptive threshold-sensitive energy-efficient sensor network protocol (APTEEN)

3.2.3 POSITION CENTRIC

If we see sensor networks as a way to instrument the physical world, the reported data almost always has to be associated with a position (e.g., temperature map or motion detection). While global coordinates and compatibility are desirable, the Global Positioning System (GPS) may not always be used because of line-of-sight conditions, form factor and power requirements, or cost. The problem of positioning nodes in the field has been addressed by many research communities: vision, networking, robotics, and signal processing. Many of these solutions do not adapt directly to the size and power constraints of the sensor, but careful design of the sensor may enable hardware features that allow for effective positioning.

Because sensor networks' main goal is to monitor physical space, their operation is intrinsically linked to location. In many cases it makes more sense to address an area of sensors by their location rather than by their IP addresses [6]. The position-centric approach uses positions of nodes as a primary means to address and route packets. In its simplest form, called Cartesian forwarding, if a source knows the position of the destination; it forwards packets to the neighbor closest to the destination. This method was actually mentioned in the 1970s, in the context of what were then called packet radio networks, precursors of today's ad hoc networks.

The position-centric way of addressing comes with a number of advantages and disadvantages. One good thing about it is that there is no need for routing tables in the network, since every node can decide how to forward packets based only on the destination of the packet and some locally gathered information about its immediate neighbors. Another positive aspect is independence from mobility: as long as intermediate nodes with known positions exist between source and destination, routing is performed without the penalty of route discoveries and updates.

The disadvantage is that the source must know the position of the destination. However, this is an implicit requirement in many applications, like sensor networks that relay all data to a unique known collection of static sinks, or when the requester of the data includes its position with the request.

The following are the protocols which uses position centric network model [1]:

- Minimum energy communication network (MECN)
- Small minimum energy communication network (SMECN)
- Geographic adaptive fidelity (GAF)
- Geographic and energy aware routing (GEAR)

3.2.4 QoS-ORIENTED

Quality of service (QoS) aware protocols consider end-to-end delay requirement in setting up the paths in the sensor network. The following are the protocols which uses hierarchical network model [1]:

- Sequential assignment routing (SAR)
- Stateless protocol for end-to-end delay (SPEED)

CHAPTER 4

WIRELESS SENSOR NODE ARCHITECTURE

4.1 FUNCTIONS OF WIRELESS SENSOR NODES

The term sensor network constitute of different sensors in any particular area where the network is established. Generally, a sensor network is highly distributed in nature. In this network very large number of sensors is deployed. A sensor is a micro-electro-mechanical device.

Typically, a wireless sensor node (or simply sensor node) consists of sensing, computing, communication, actuation, and power components. These components are integrated on a single or multiple boards, and packaged in a few cubic inches. With state-of-the-art, low-power circuit and networking technologies, a sensor node powered by 2 AA batteries can last for up to three years with a 1% low duty cycle working mode. A WSN usually consists of tens to thousands of such nodes that communicate through wireless channels for information sharing and cooperative processing. [16].

Wireless node design needs to be supported [1]:

- Intrinsic node functionality
- Signal processing
- Compression
- Forward error correction and encryption
- Control and actuation
- Clustering and in-network computation
- Self assembly communication
- Routing and forwarding
- Connectivity management

4.2 BASIC HARDWARE COMPONENTS OF WSNs:

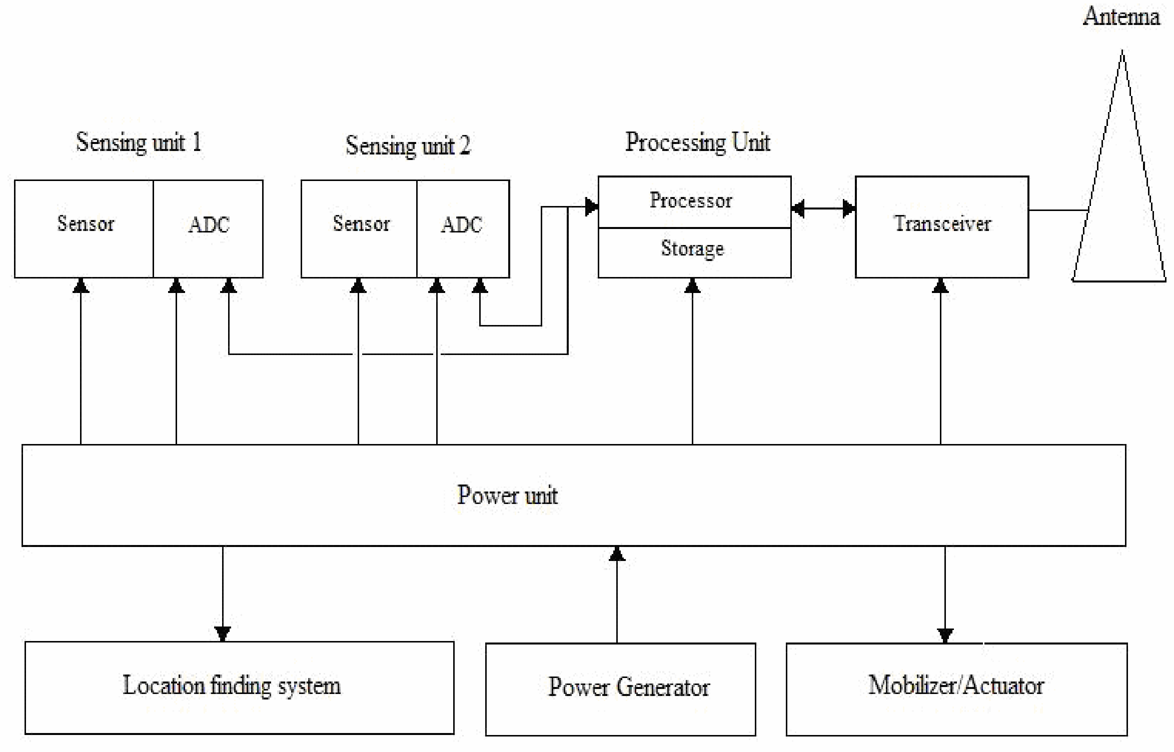


Figure 4.1: Hardware component of WSNs

- **Power:** An appropriate energy infrastructure or supply is necessary to support operation from a few hours to months or years (depending on the application).

- **Computational logic and storage:** These are used to handle onboard data processing and manipulation, transient and short-term storage, encryption, forward error correction (FEC), digital modulation, and digital transmission. WNs have computational requirements typically ranging from an 8-bit microcontroller t a 64-bit microprocessor. Storage requirements typically rang from 0.01 to 100 gigabytes (GB).

- **Sensor transducer:** The interface between the environment and the WN is the sensor. Basic environmental sensors include, but not limited to, acceleration, humidity, light, magnetic flux, temperature, pressure, and sound.

- **Communication unit:** This unit mainly responsible for the data exchange with other nodes. There are paths between this unit with processing unit as well as with storage unit. Whenever a request will come for any particular data/ message then the processing unit will perform the required computation and then inform the communication unit to transfer it further. When a node will receive any data/message then also it will inform the processing unit and store it accordingly instructed by the processing unit.

4.3 BASIC SOFTWARE COMPONENT OF WSNs:

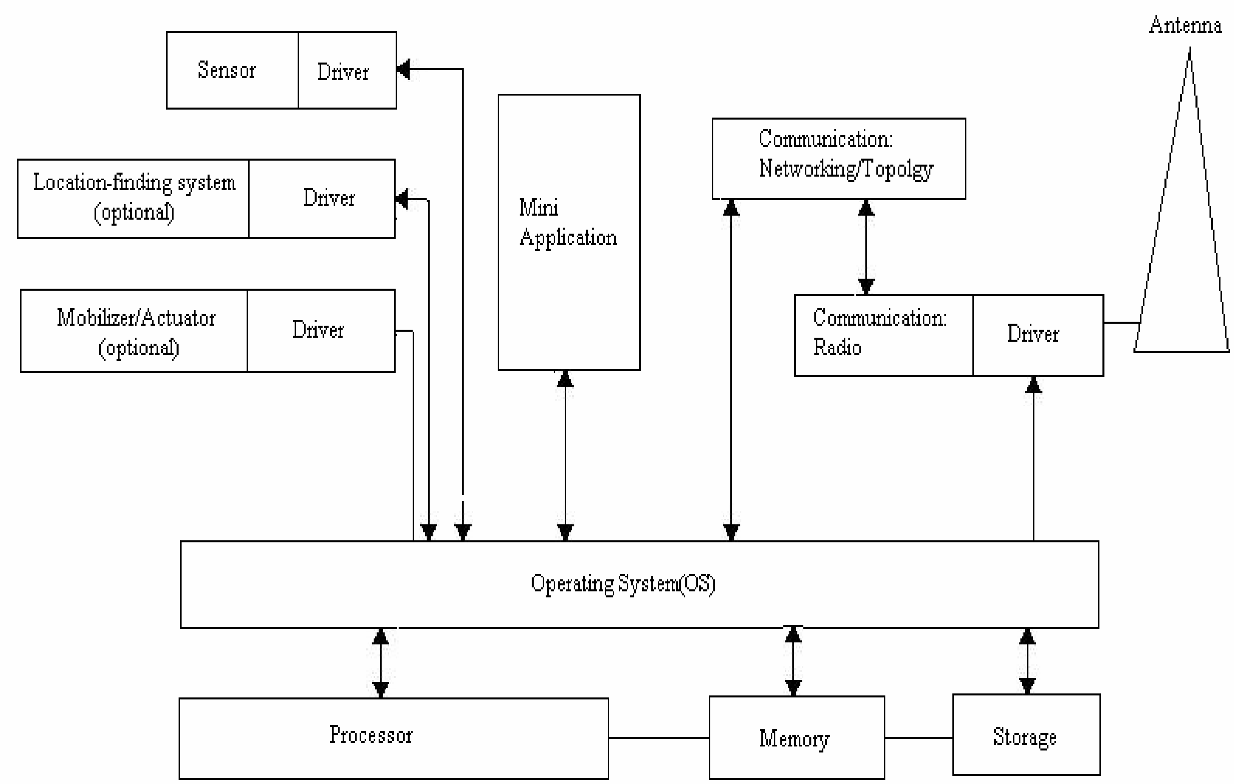


Figure 4.2: software components of WSNs

- **Operating system (OS) microcode (also called middle- ware):** This is the board common microcode that is used by all high-level node-resident software modules to support various functions. The purpose of an operating system is to shield the software from the machine level functionality of the microprocessor. It is desirable to have open-source operating systems designed specifically for WSNs; OSs typically utilizes an architecture that enables rapid implementation while minimizing cod size. TinyOS is on such example f a commonly used OS.

- **Sensor drivers:** These are the software modules that manage basic functions of the sensor transceivers; sensors may possibly be the modular/plug-in type, and depending on the type and sophistication, the appropriate configuration and settings must be uploaded into the sensor (drivers shield the application software from the machine –level functionality of the sensor or other peripheral).

- **Communication processors:** This code manages the communication functions, including routing, packet buffering and forwarding, topology maintenance, medium access control (e.g., contention mechanisms, direct-sequence spread-spectrum mechanisms), encryption, and FEC.

- **Communication drivers:** These software modules manage the radio channel transmission link, including clock synchronization, signal encoding, bit recovery, bit counting, signal levels, and modulation.

- **Data processing mini-apps:** These are numerical, data-processing, signal-value storage and manipulations, or other basic applications that are supported at the node level for in-network processing.

CHAPTER 5

ROUTING STRATEGIES IN WIRELESS SENSOR NETWORKS

5.1 INTRODUCTION

Before discussing about routing strategies, we will take following assumptions about network [14].

- Each sensor node has power control and the ability to transmit data to any other sensor node or directly to the BS.
- Sensor network contains homogeneous and energy constrained sensor nodes with initial uniform energy.
- Every node has location information of other nodes
- There is no mobility**.**

A simple approach to accomplishing this data gathering task is for each node to transmit its data directly to the BS. Since the BS is typically located far away, the cost to transmit to the BS from any node is high so nodes will die very quickly. Therefore, an improved approach is to use as few transmissions as possible to the BS and reduce the amount of data that must be transmitted to the BS in order to reduce energy. Further, if all nodes in the network deplete their energy levels uniformly, then the network can operate without losing any nodes for a long time.

Another important factor to consider in the data gathering application is the average delay per round. Here, we assume that data gathering rounds are far apart and the only traffic in the network is due to sensor data. Therefore, data transmissions in each round can be completely scheduled to avoid delays in channel access and collisions. The delay for a packet transmission is dominated by the transmission time as there is no queuing delay and the processing and propagation delays are negligible compared to the transmission time. With the direct transmission scheme, nodes will have to transmit to the base station one at a time, making the delay a total of N units (one unit per transmission, where N is equal to the number of nodes). To reduce delay, one needs to perform simultaneous transmissions.

In sensor networks, data fusion helps to reduce the amount of data transmitted between sensor nodes and the BS. Data fusion combines one or more data packets from different sensor measurements to produce a single packet, as described in [5].

5.2 ENERGY × DELAY REDUCTION FOR DATA GATHERING IN SENSOR NETWORKS

Why energy × delay metric? Clearly, minimizing energy or delay in isolation has drawbacks. For battery operated sensors, longevity is a major concern and priorities can be entirely different when energy reserves become depleted. Energy efficiency often brings additional latency along with it. Minimizing delay is not always practical in sensor network applications. Maximizing the throughput is not the best strategy for energy-critical links. Generally, increased energy savings come with a penalty of increased delay. However, several practical applications set limits on acceptable latency, as specified by QoS requirements. For example, the data gathering delay per round may have a bound. Therefore, there is a tradeoff between energy spent per packet and delay; energy × delay is an appropriate measure to optimize for wireless sensor networks.

5.3 LOW - ENERGY ADAPTIVE CLUSTRING HIERARCHY (LEACH)

Low–energy adaptive clustering hierarchy (LEACH) is a routing algorithm designed to collect and deliver data to the data sink, typically a base station. The main objective of leach is [3]:

- Randomized, adaptive, self-configuring cluster formation
- Extension of the network lifetime
- Low -energy media access control (MAC)
- Use of data aggregation and compression to reduce the number of communication messages

To achieve these objectives, LEACH adopts a hierarchical approach to organize the network into a set of clusters. Each cluster is managed by a selected cluster head. The cluster head assumes the responsibility to carry out multiple tasks.

The first task consists of periodic collection of data from the member of the cluster. Upon gathering the data, the cluster head aggregates it in an effort to remove redundancy among correlated values [1]**.**

The second task of a cluster head is to transmit the aggregate data directly to the base station. The transmission of the aggregate data is achieved over a single hope. The network model used by LEACH is shown in fig. 5.1.

The third task of the cluster head is to create a TDMA- based schedule whereby each node of the cluster is assigned a time slot that it can used for transmission. The cluster head advertise the schedule to its cluster member through broadcasting. To reduce the likelihood of collision among sensors within and outside the cluster, LEACH nodes use a code division multiple accesses (CDMA) based scheme for communication.

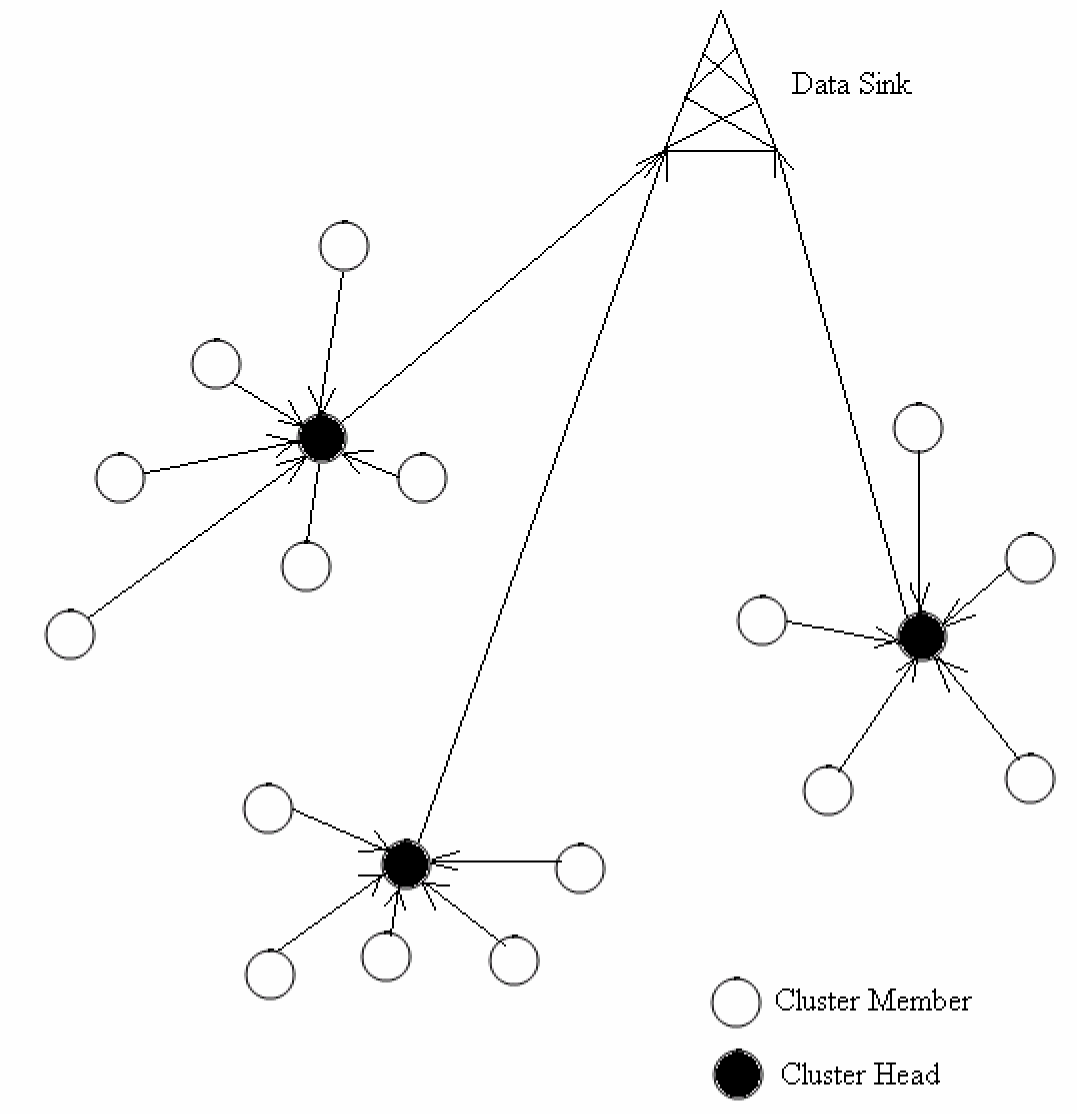


Figure 5.1: LEACH network model

The basic operations of LEACH are organized in two distinct phases [3].

The first phase, the setup phase, consist of two steps, cluster head selection and cluster formation. The second phase, the steady-state phase, focuses on data collection, aggregation, and delivery to the base station. The duration of the setup is assumed to be relatively shorter than the steady-state phase to minimize the protocol overhead.

At the beginning of the setup phase, a round of cluster-head selection starts. The cluster head selection process ensures that this role rotates among sensor nodes, there by distributing energy consumption evenly across all network nodes.

At the completion of cluster head selection process every node that was selected node to become a cluster head advertisement its new role to the rest of the network. Upon receiving the cluster head advertisement, each remaining node selects a cluster to join. The selection criteria may be based on the received signal strength among other factors. The nodes then inform their selected cluster head of their desire to become a number of clusters.

Upon cluster formation each cluster head create and distributes the TDMA schedule, which specifies time slots allocated for each member of the clusters. Each cluster head selects a CDMA code, which is than distributed to all members of its clusters. The code is selected carefully so as to reduce inter cluster interference. The completion of the set up phase signals the beginning of the steady state phase, during this phase nodes collects their information and use their allocated slots to transmit to the cluster head the data collected. This data collection performs periodically. Simulation results show that LEACH achieves significant saving. These savings depend primarily on the data aggregation ration achieved by the cluster head.

Despite these benefits, however LEACH suffers several shortcomings. The assumption can reach the base station in one hop may not be realistic, as capabilities and energy reserves of the nodes may vary over times from one node to another. Further more the length of the steady state period is critical to achieving the energy reduction necessary to offset the overhead cost by the cluster selection process. A short steady state period increases the over head, whereas a long period may lead to cluster head energy depletion.

LEACH exhibits several properties which enables the protocol to reduce energy consumption. Energy requirement in LEACH is distributed across all sensor nodes, assumes the cluster head role in round robin fashion based on their residual energy. LEACH is a completely distributed algorithm, requiring no control information from the base station. The cluster management is achieved locally which obliterates the need for global network knowledge. Furthermore, data aggregation by the cluster also contributes greatly to energy saving as nodes are no longer required to send their information directly to the sink.

Further improvements can be obtained if each node communicates only with close neighbors and only one designated node sends the combined data to the BS in each round in order to reduce energy. A new protocol based on this approach, called PEGASIS (Power-Efficient GAthering in Sensor Information Systems) would be discussed in the next section.

5.4 POWER EFFICIENT GATHERING IN SENSOR INFORMATION SYSTEMS (PEGASIS)

Power efficient gathering in sensor information systems (PEGASIS) and its extension, hierarchical PEGASIS, are a family of routing and information-gathering for WSNs. The main objectives of PEGASIS are twofold. First the protocol aims at extending the lifetime of the network by achieving a high level of energy efficiency and uniform energy consumption across all network nodes [4]. Second the protocol strives to delay incur on their way to the sink.

Nodes are assumed to have global knowledge about other sensors' positions and they have the ability to control their power to cover arbitrary ranges. The nodes may also be equipped with CDMA capable radio transceivers. The nodes responsibility is to gather and deliver data to the sink, typically to a wireless base station. PEGASIS uses a chain structure for data routing**.**

The greedy algorithm would be used for constructing the chain. We could have constructed a loop. However, to ensure that all nodes have close neighbors. 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 (select a node randomly if there is a tie).

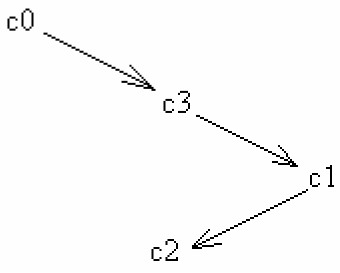


Figure 5.2: Chain construction using the greedy algorithm

The closest neighbor to this node will be the next node on the chain. Successive neighbors are selected in this manner among unvisited nodes (with ties broken arbitrarily) to form the greedy chain. We begin with the farthest 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. Fig.5.2 shows node c0 connecting to node c3, node c3 connecting to node c1, and node c1 connecting to node c2, in that order. When a node dies, the chain is reconstructed in the same manner to bypass the dead node.

Alternatively, in a given round, we can use a simple control token passing approach initiated by the leader to start the data transmission from the ends of the chain. The cost is very small since the token size is very small [14].

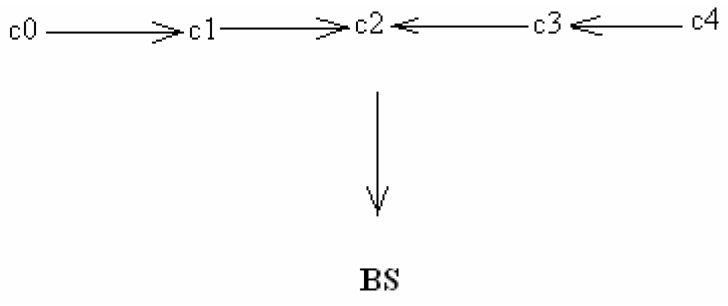


Figure 5.3: Token passing approach

In Fig. 5.3, node c2 is the leader and it will pass the token along the chain first to node c0. Node c0 will pass its data toward node c2. After node c2 receives data from node c1, it will pass the token to node c4, and node c4 will pass its data towards node c2 with data fusion taking place along the chain. PEGASIS performs data fusion at every node except the end nodes in the chain. Each node will fuse its neighbor's data with its own to generate a single packet of the same length and then transmit that to its other neighbor (if it has two neighbors). In the above example, node c0 will transmit its data to node c1. Node c1 fuses node c0's data with its own and then transmits to the leader. After node c2 passes the token to node c4, node c4 transmits its data to node c3. Node c3 fuses node c4's data with its own and then transmits to the leader. Node c2 waits to receive data from both neighbors and then fuses its data with its neighbors' data. Finally, node c2 transmits one message to the BS. Thus, in PEGASIS, each node, except the two end nodes and the leader node, will receive and transmit one data packet in each round and be the leader once every N rounds. In addition, nodes receive and transmit very small control token packets. The simulation results of the hierarchical extension of PEGASIS show considerable improvement over scheme such as LEACH [14].

The chain based binary approach leads to significant energy reduction, as nodes operate in highly parallel manner. The chain based binary aggregation scheme has been used in PEGASIS as an alternative to achieving high degree of parallelism with CDMA capable sensor nodes will be discussed in the next section.

.

Problems of the current the PEGASIS protocol

The current PEGASIS protocol may have several problems
as follows [9]:

- Each sensor node is required to have extra local information about the wireless sensor network.
- When the PEGASIS protocol selects the head node, there is no consideration about the energy of nodes.
- When the PEGASIS protocol applies to the greedy algorithm to the construct chain, some delay may occur.
- Since the head node is a single, it may occur a bottleneck at the head node.
- When the PEGASIS protocol selects the head node, there is no consideration about the location of the base station.

5.5 A CHAIN-BASED BINARY SCHEME

First, we consider a sensor network with nodes capable of CDMA communication. With this CDMA system, it is possible for node pairs that communicate to use distinct codes to minimize radio interference. Thus, parallel communication is possible with 50 pairs for the 100-node network of interest. In order to minimize the delay, we will combine data using as many pairs as possible in each level which results in a hierarchy of $\log_2 N$ levels. At the lowest level, we will construct a linear chain among all the nodes, as was done in PEGASIS, so that adjacent nodes on the chain are nearby. For constructing the chain, we assume that all nodes have global knowledge of the network and employ the greedy algorithm. 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. For gathering data in each round, each node transmits to a close neighbor in a given level of the hierarchy. This occurs at every level in the hierarchy, but the only difference is that the nodes that are receiving at each level are the only nodes that rise to the next level. Finally, at the top level the only node remaining will be the leader.

Suppose node i will be in some random position j on the chain. Nodes take turns transmitting to the BS, and we will use node number i mod N (N represents the number of nodes) to transmit to the BS in round i.

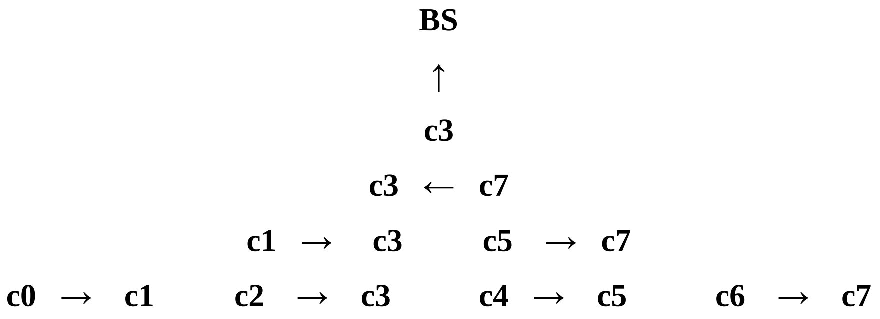


Figure 5.4: Data gathering in a chain-based binary scheme

In Figure 5.4, for round 3, node c3 is the leader. Since, node c3 is in position 3 (counting from 0) on the chain, all nodes in an even position will send to their right neighbor. Now at the next level, node c3 is still in an odd position so again; all nodes in an even position will fuse its data with its received data and send to their right. At the third level, node c3 is not in an odd position, so node c7 will fuse its data and transmit to c3. Finally, node c3 will combine its current data with that received from c7 and transmit the message to BS.

The chain-based binary scheme performs data fusion at every node that is transmitting except the end nodes in each level. Each node will fuse its neighbor's data with its own to generate a single packet of the same length and then transmit that to the next node. In the above example, node c0 will pass its data to node c1. Node c1 fuses node c0's data with its own and then transmits to node c3 in the next level. In our simulations, we ensure that each node performs equal number of sends and receives after N rounds of communication, and each node transmitting to the BS in one of N rounds.

The chain-based binary scheme improves on LEACH by saving energy and delay in several stages. At the lower levels, nodes are transmitting at shorter distances compared to nodes transmitting to a cluster head in the LEACH protocol, and only one node transmits to the BS in each round of communication. We calculate the average energy cost per round and the delay cost in the next chapter.

5.6 FIBONACCI SERIES BASED ENERGY AWARE ALGORITHM:

An efficient Fibonacci series based hierarchical protocol-HFTM (Hierarchical Fibonacci Tree Multicast) is proposed for application layer and for multicasting [7].

The input of the algorithm is, number of nodes deployed in an area or cluster. This algorithm adopts the idea of Fibonacci series to partition the number of nodes into parts with different sizes. The Fibonacci series $\{f_i\}$ satisfies the following condition: $f_0 = 0$, $f_1=1$; $f_n = f_{n-1} + f_{n-2}$, if $n > 1$.

Algorithm description:

Input: member sequence Φ = {d_1, d_2, d_3,........,.d_K) , d_s is the cluster leader which serves as the source node, d_s € Φ. The number of members in Φ is K . $f_n \le K < f_{n+1}$

Output: a multicast tree constructed for all members in Φ

1. If (K = 2) d_s send packets to the only destination;

2. If (K > 2) Φ is partitioned into two subsequences Φ_1 and Φ_2 where d_s is
in the larger subsequence and the smaller one includes f_{n-2} members;

2.1 If ($s > f_{n-2}$) {$\Phi_1 = (d_1, d_2, d_3 \ldots d_{f\,n-2})$; $\Phi_2 = (d_{fn-2+1}, d_{fn-2+2}, \ldots\ldots d_k)$; }
Else {$\Phi_1 = (d_{1,} d_2 \ldots\ldots d_{K-fn-2})$ $\Phi_2 = (d_{K-fn-2+1,} d_{K-fn-2+2, \ldots,} d_K)$; }

2.2 If ($s > f_{n-2}$) { d_s firstly sends packets to d_1 , then d_2 is in charge of multicasting in Φ_1 and d_s is in charge of multicasting in Φ_2 ; }
Else { d_s firstly sends packets to $d_{K-fn-2+1}$ then d_s is in charge of multicasting in Φ_1 and $d_{K-fn-2+1}$ is in charge of multicasting in Φ_2 ;}

3. Multicast packets from d_1 to all members in Φ_1 and from d_s to all members in Φ_2 (or multicast packets from d_s to all members in Φ_1 and from $d_{K-fn-2+1}$ to all members in Φ_2) by recursive calls Algorithm 1.

Details of the above algorithm will we discussed in the next chapter.

.

CHAPTER 6

ENERGY × DELAY ANALYSIS FOR DATA GATHERING

6.1 RADIO MODEL FOR ENERGY CALCULATIONS

There is some fixed amount of energy cost in the electronics when transmitting or receiving a packet and a variable cost when transmitting a packet which depends on the distance of transmission and this additional amount of energy is directly proportional to d^2. If each node transmits its sensed data directly to the base station, then it will deplete its power quickly. The equations used to calculate transmission costs and receiving costs for a k-bit message and a distance d [14] are shown below:

A) Transmitting:

$E_{Tx}(k, d) = E_{Tx\text{ - }elec}(k) + E_{Tx – amp}(k, d)$

$E_{Tx}(k, d) = E_{elec} \times k + \epsilon_{amp} \times k \times d^2$

B) Receiving:

$E_{Rx}(k) = E_{Rx – elec}(k)$

$E_{Rx}(k) = E_{elec} \times k$

There are several elegant routing algorithms to solve the above problem. In this chapter we will compare the delay and energy for PEGASIS, Chain Based Binary Scheme and Fibonacci Series Based Energy Aware Algorithm discussed in Section [5.6]**.**

Consider the example, for linear network where the N nodes are along a straight line with equal distance of d between each pair of nodes and the BS at a faraway distance from all nodes. The direct approach will require high energy cost and the delay will be N as nodes transmit to the BS sequentially.

6.2 DELAY AND ENERGY ANALYSIS FOR PEGASIS [4]:

The PEGASIS scheme forms a chain among the sensor nodes so that each node will receive from and transmit to a close neighbor.

- Delay = N - 1
- Energy cost = $(N - 1) \times d^2$
- Energy × Delay Cost = $(N - 1)d^2$

6.3 DELAY AND ENERGY ANALYSIS FOR CHAIN BASED BINARY SCHEME [14]

Consider the distance between two nodes is d, and they are linearly distributed. In the binary scheme with perfect parallel transmission of data, there will be N/2 nodes transmitting data to their neighbors at distance d in the lowest level. The nodes that receive data will fuse the data with their own data and will be active in the next level of the tree. Next, N/4 nodes will transmit data to their neighbors at a distance 2d and this procedure continues until a single node finally transmits the combined message to the BS. Thus, for the binary scheme, the energy cost will be:

$N/2 \times d^2 + N/4 \times (2d)^2 + N/8 \times (4d)^2 + \ldots\ldots\ldots\ldots + 1 \times (N/2 * d)^2$

Since the distance doubles as we go up the hierarchy. Therefore,

$N/2 \times d^2 \times (1 + 2 + 4 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + N/2)$

Which equals $N (N - 1)/2 \times d^2$

We can approximate the total energy cost for the binary scheme to be:

$N^2 /2 \times d^{2}$.

Hence,

- Delay = $\log_2 N$
- Energy cost = $(N^2 /2) \times d^2$
- Energy × Delay Cost = $N^2 /2 \times d^2 \times \log_2 N$

Therefore, for this linear network, the binary scheme will be more expensive than PEGASIS in terms of energy × delay but the delay performance is quite well as compared to PEGASIS [14].

6.4 DELAY AND ENERGY ANALYSIS FOR FIBONACCI SERIES BASED ENERGY AWARE ALGORITHM:

6.4.1 Basic Assumptions:

- All sensor nodes are homogeneous.
- The distance between two nodes is d unit.
- All the nodes have CDMA capability.

- All nodes are location aware and have knowledge about others location.
- Parallel transmission is possible between two nodes.
- A node cannot receive data from two nodes simultaneously, i.e., a node can receive data from a single node at any time instant.
- A node takes only one time unit to transmit data to its neighboring node and does not depend upon how far the neighboring node is located.
- The head node is heavily charged as compared to the other nodes. This can be easily done because the location of head node is fixed in Fibonacci Series based Energy aware algorithm.
- Nodes are deployed in an area such that the distance from the fixed head node to the base station is minimum.
- Data fusion occurs at every node except the end nodes, to produce a single packet.

Here, we are using the concept described in section [5.6] in wireless sensor network to reduce the delay and energy × delay. The given number of nodes divided into two parts. Then recursively grouping is done and finally a single node collects the fused data. The nodes which send final data to the base station is called head node. Head node can be any node in the given sequence.

6.4.2 Head Node selection:

Applying Fibonacci series based energy aware algorithm once to the given number of nodes, breaks the number of nodes into two parts. The last node of first part or the first node of second part is considered as a head node, because the energy can be optimized only at this node. But we have used first node of second part as a head node in our analysis.

If we follow the above criteria to select the head node, energy and delay can be optimized. This can be shown by the following examples.

For N = 3 (N is the no. of nodes), there are three possibilities.

(A) If node 1 is head node then the energy cost is $5d^2$ $(d^2 + 4d^2)$

(B) If node 2 is head node then the energy cost is $2d^2$ $(d^2 + d^2)$

(C) If node 3 is head node then the energy cost is also $2d^2$ $(d^2 + d^2)$

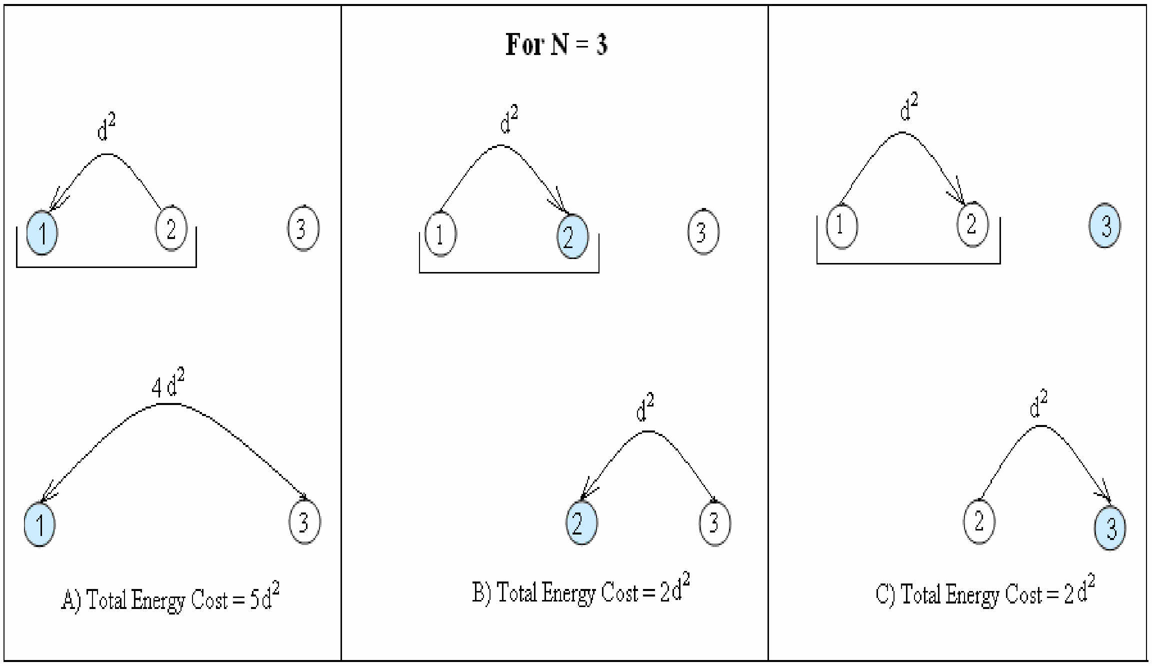

For N = 3
d^2
1
2
3
4 d^2
1
3
A) Total Energy Cost = 5d^2
d^2
1
2
3
d^2
2
3
B) Total Energy Cost = 2d^2
d^2
1
2
3
d^2
2
3
C) Total Energy Cost = 2 d^2

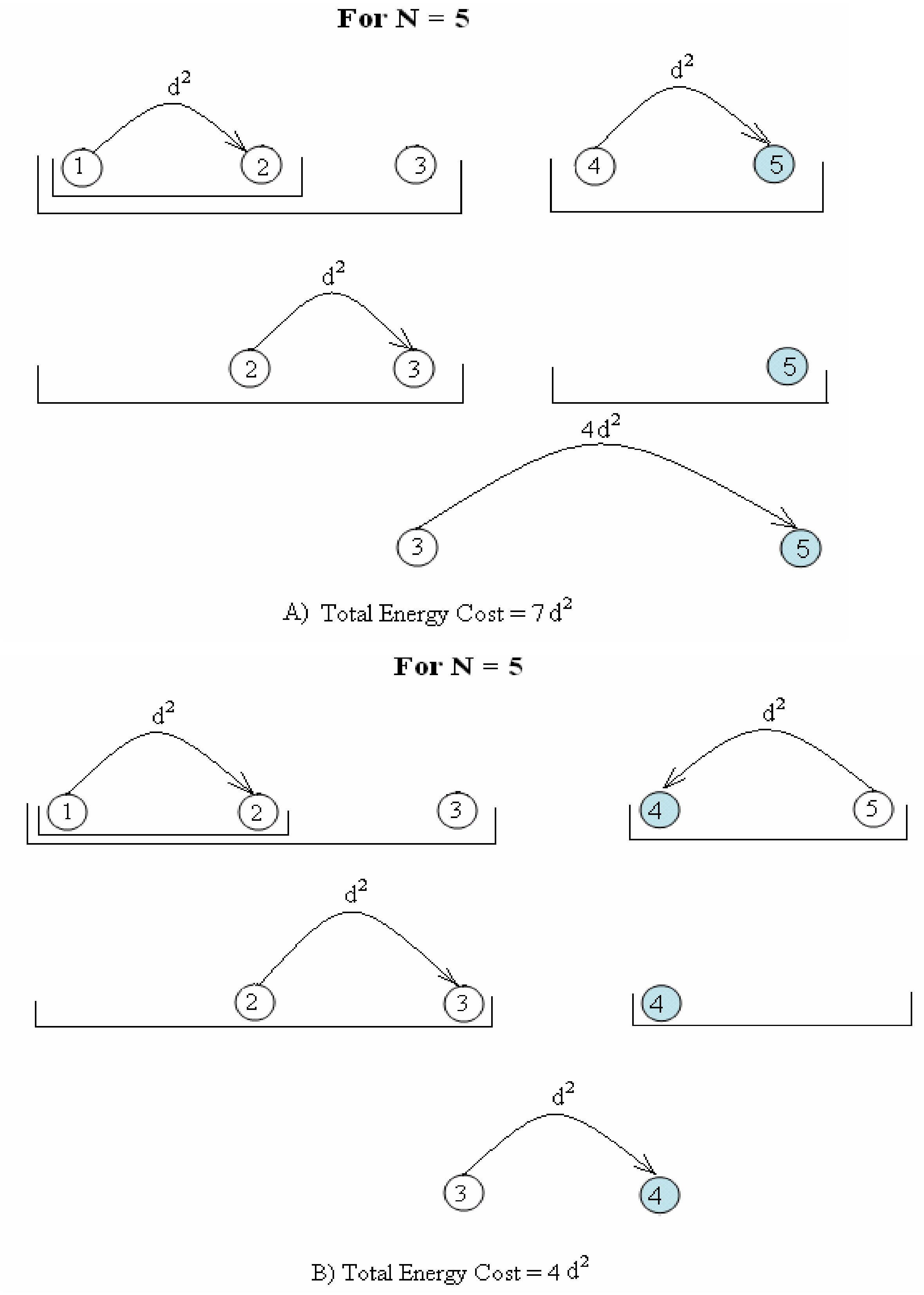
Figure 6.1: Energy cost for N = 3

The same approach can be applied for any number of nodes. Take another example.

For N = 5

(A) If node 5 is head node then the energy cost is $7d^2$ ($d^2 + d^2 + d^2 + 4d^2$)

(B) If node 4 is head node then the energy cost is $4d^2$ ($d^2 + d^2 + d^2 + d^2$)



A) Total Energy Cost = $7\,d^2$

B) Total Energy Cost = $4\,d^2$

Figure 6.2: Energy cost for N = 5

ENERGY ANALYSIS

We have calculated energy cost for some nodes. For e.g.

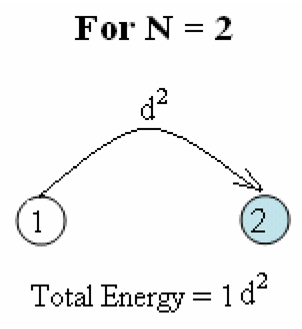


Figure 6.3: Energy cost for N = 2

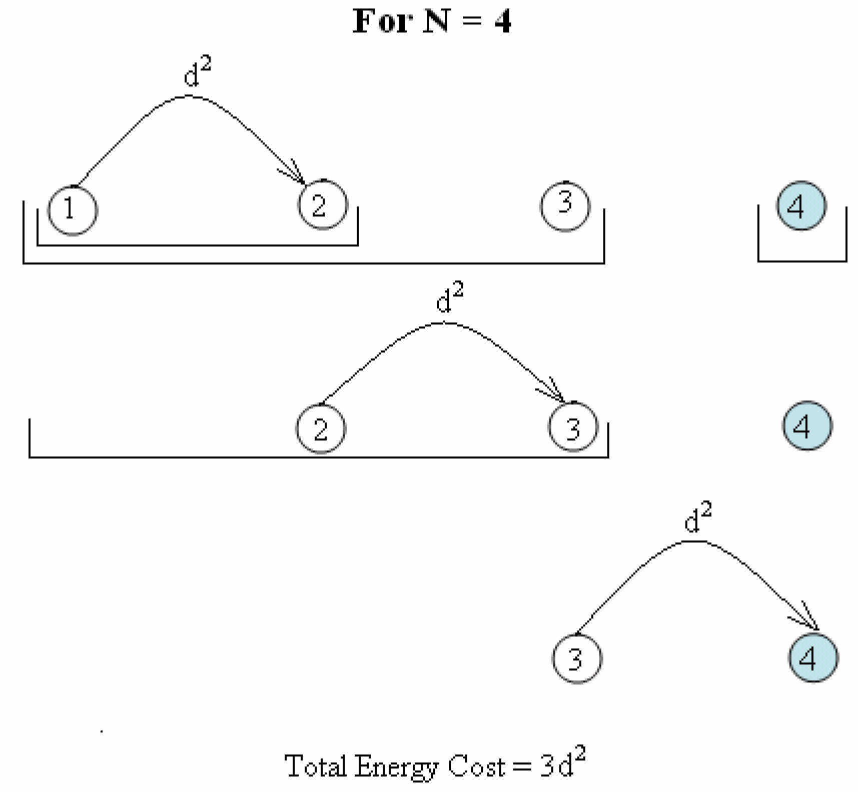


Figure 6.4: Energy cost for N = 4

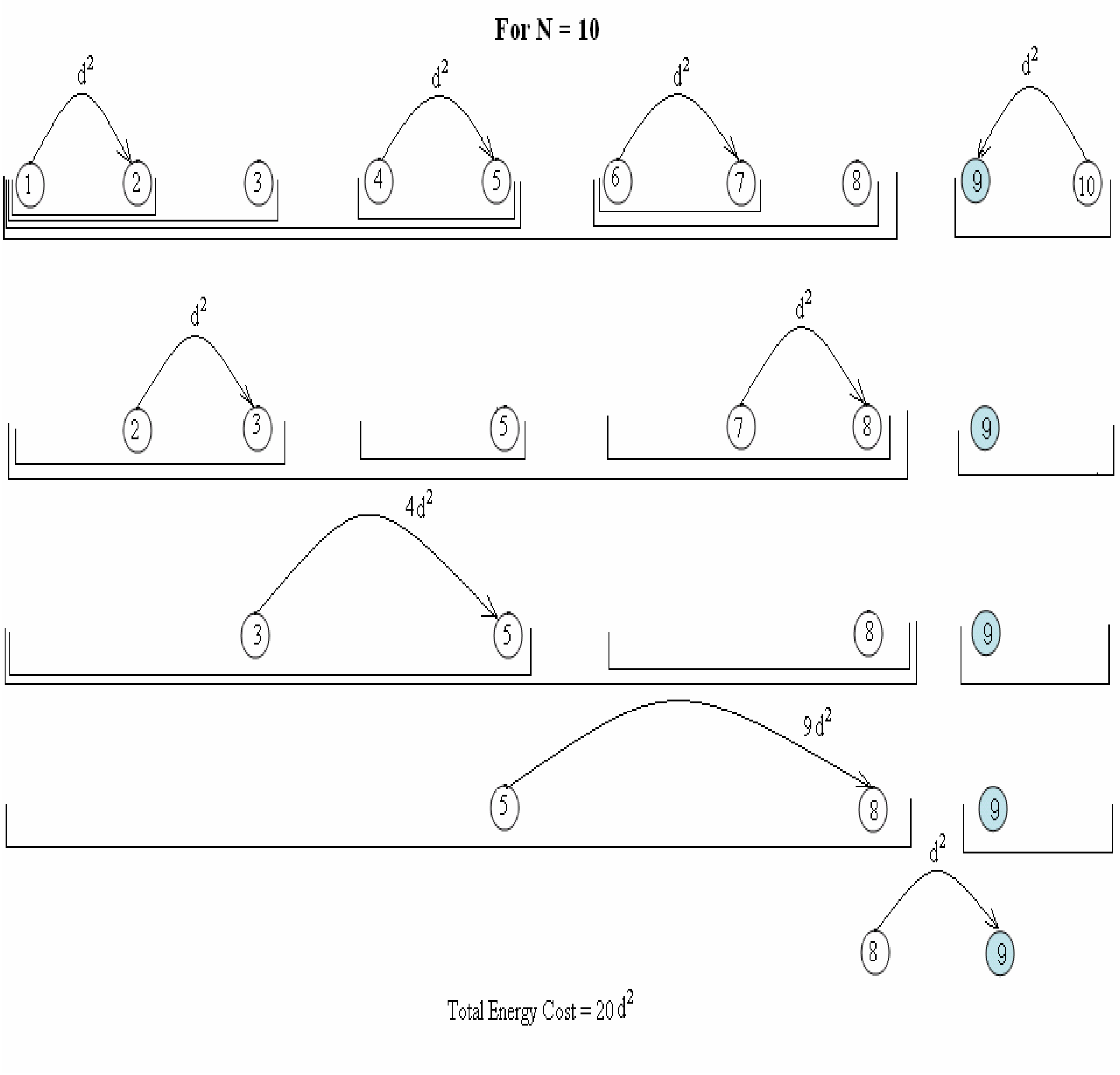


Figure 6.5: Energy cost for N = 10

DELAY ANALYSIS

We will find out total time delay to receive the data from different nodes to the head node. As we have already assumed that simultaneous transmission is possible between two nodes but two nodes can not transmit data simultaneously to a node. Consider an example for N = 10. Break these nodes into groups according to Fibonacci series based energy aware algorithm. Data transfer at different times:

- **At Time T1:** Node 1 sends data to node 2, node 4 to node 5, node 6 to node 7 and node 10 to node 9
- **At Time T2:** Node 2 sends data to node 3, node 7 to node 8
- **At Time T3:** Node 3 sends data to node 5
- **At Time T4:** Node 5 sends

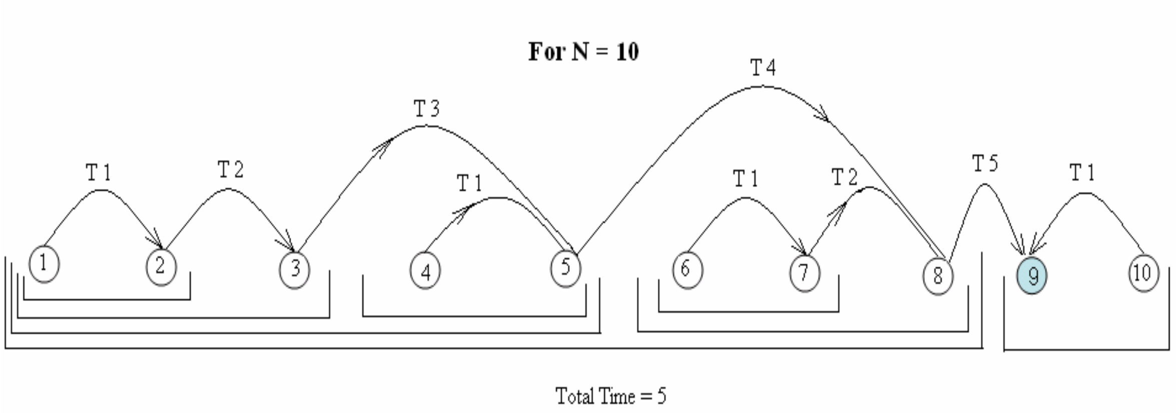


Figure 6.6: Delay cost for N = 10

Similarly for different values of number of nodes, the delay calculation takes place as follows:

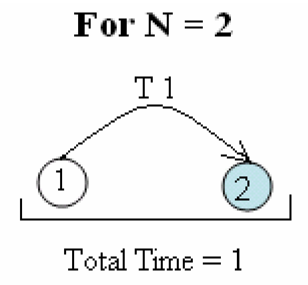


Figure 6.7: Delay cost for N = 2

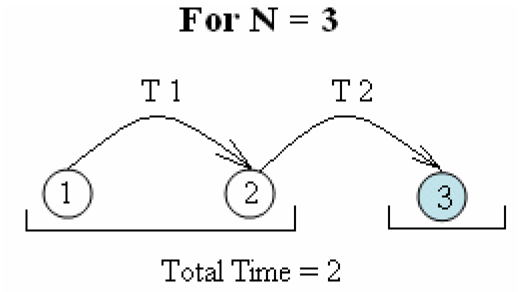


Figure 6.8: Delay cost for N = 3

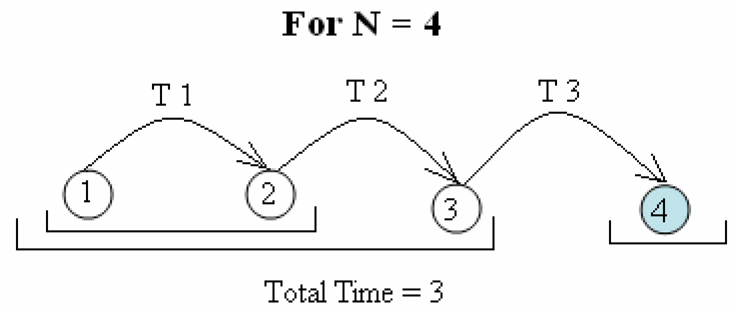


Figure 6.9: Delay cost for N = 4

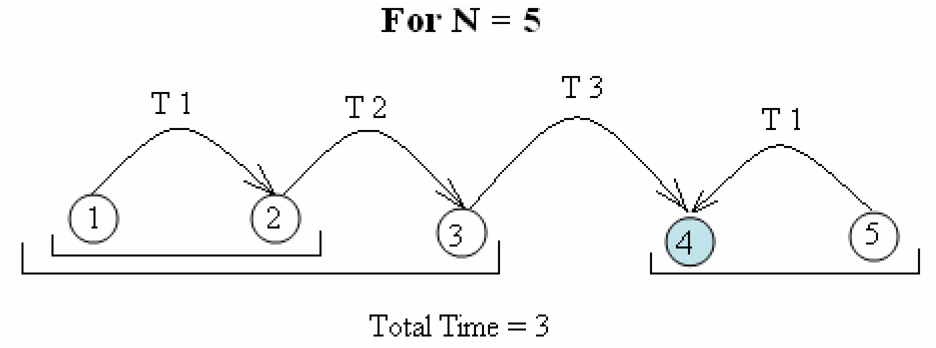


Figure 6.10: Delay cost for N = 5

CHAPTER 7

PERFORMANCE COMPARISON OF PEGASIS, CHAIN BASED BINARY SCHEME AND FIBOONACCI SERIES BASED ENERGY AWARE ALGORITHM

Based upon the formula's discussed in section [6.2], [6.3] and [6.4], we will calculate delay and energy × delay for these three algorithms.

NUMBER OF NODES	DELAY			ENERGY*DELAY		
	PEGASIS	BINARY SCHEME	FIBONACCI SERIES BASED ENERGY AWARE ALGORITHM	PEGASIS	BINARY SCHEME	FIBONACCI SERIES BASED ENERGY AWARE ALGORITHM
2	1	1	1	4	2	1
3	2	2	2	9	8	4
4	3	2	3	16	16	9
5	4	3	3	25	30	16
6	5	3	4	36	47	32
7	6	3	4	49	69	36
8	7	3	4	64	96	50
9	8	4	5	81	129	95
10	9	4	5	100	167	100
11	10	4	5	121	210	105
12	11	4	5	144	259	132
13	12	4	6	169	313	156
14	13	4	6	196	374	306
15	14	4	6	225	440	312
16	15	4	6	256	512	318

17	16	5	6	289	591	324
18	17	5	6	324	676	348
19	18	5	6	361	767	354
20	19	5	6	400	865	378
21	20	5	6	441	969	483
22	21	5	7	484	1080	931
23	22	5	7	529	1197	938
24	23	5	7	576	1321	945
25	24	5	7	625	1452	952
26	25	5	7	676	1589	980
27	26	5	7	729	1734	987
28	27	5	7	784	1885	1035
29	28	5	7	841	2043	1057
30	29	5	7	900	2209	1216
35	34	6	8	1225	3142	2816
40	39	6	8	1600	4258	2880
45	44	6	8	2025	5561	3048
50	49	6	8	2500	7055	3304
55	54	6	8	3025	8745	3872
60	59	6	9	3600	10633	8388
65	64	7	9	4225	12723	8532
70	69	7	9	4900	15017	8820
75	74	7	9	5625	17519	9324
80	79	7	9	6400	20231	9687
85	84	7	9	7225	23154	10413
90	89	7	10	8100	26293	24320
95	94	7	10	9025	29647	24440
100	99	7	10	10000	33223	24610

Table 7.1: Algorithms Comparison Analysis

SIMULATION RESULTS

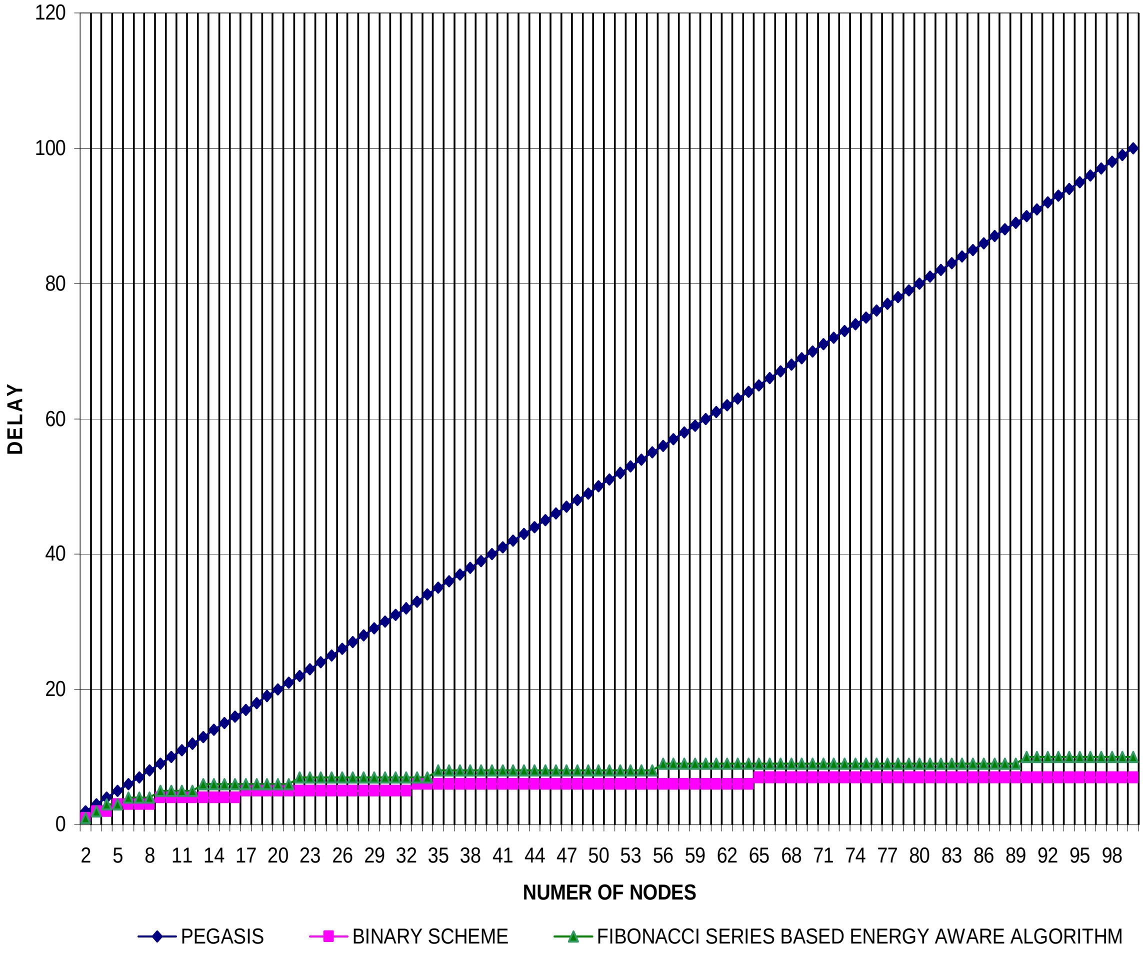


Figure 7.1: Delay performance

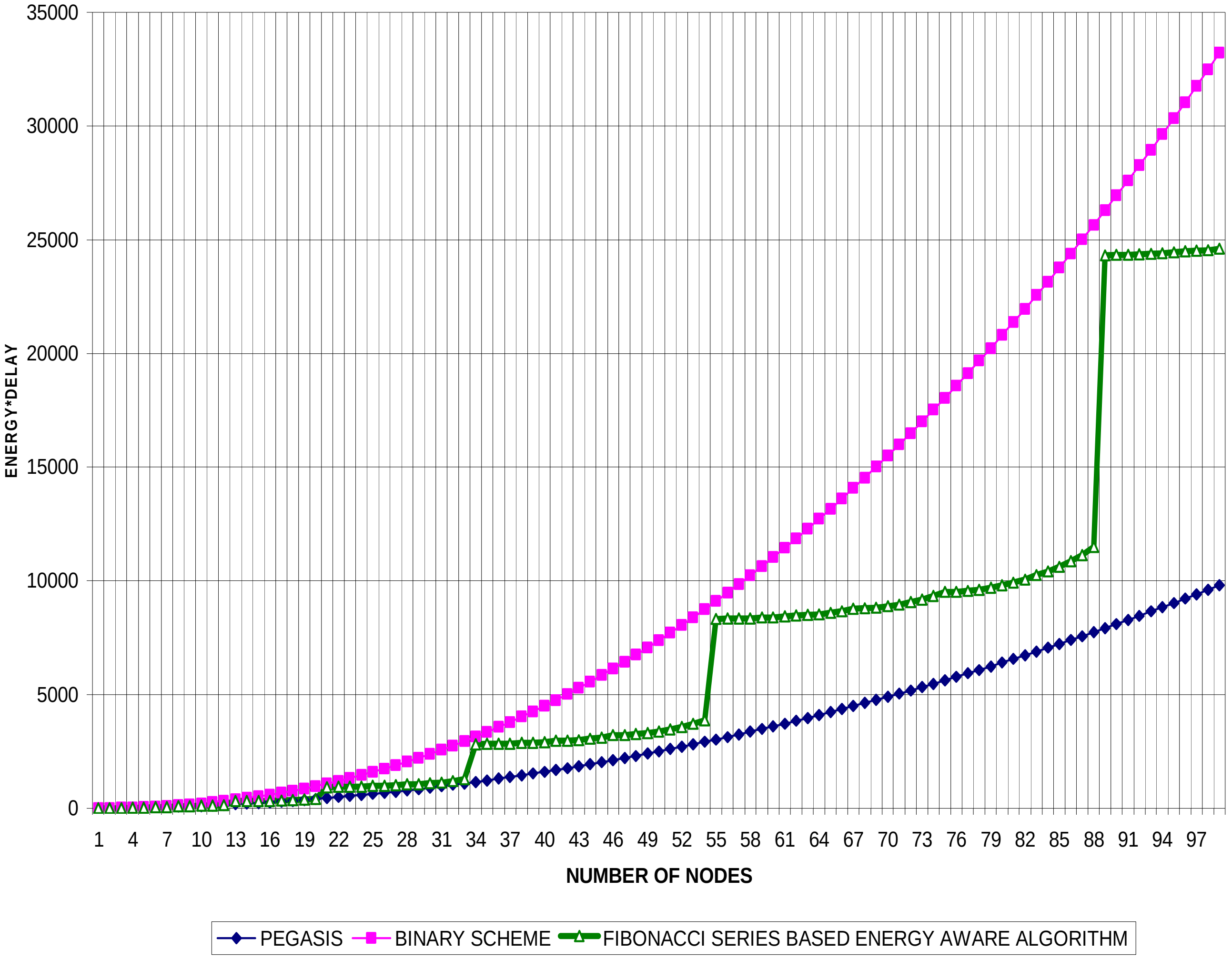


Figure 7.2: Energy × Delay performance

CHAPTER 8

CONCLUSION

Simulation results show that fibonacci series based energy aware algorithm has advantage over PEGASIS in terms of delay and, for some nodes energy × delay product also gives significant results.

Delay of fibonacci series based energy aware algorithm has delay almost equal to the chain based binary scheme. But energy × delay product of fibonacci series based energy aware algorithm is much better than chin based binary scheme.

If the number of nodes lies around 34 to 55 and 56 to 89, then the product of energy and delay comes out to be much better than PEGASIS.

It also shows that if we consider energy × delay product then there will be trade-off between delay and energy.

Energy will be drastically increased, and becomes constant until the next node evaluated will be in the fibonacci series. In short energy will be constant for the number of nodes which does not lie in fibonacci series.

Future Scope

In future we wish to research further for the solution of above mention drawback. Area can be divided into number of clusters considering the fibonacci series and the location of base station may be reduced further to increase the life time of the network.

REFERENCES

[1].”Wireless Sensor Networks Technology, Protocols, and Applications” Kazem Sohraby, Daniel Minoli, Taieb Znati, A John Willy & Sons, Inc Publications, 2007.

[2]. W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An Application –Specific Protocol Architecture for Wireless Microsensor Networks,” Proc. Hawaii Conf. System Sciences, Jan. 2000.

[3]. W. Heinzelman, “Application-Specific Protocol Architectures for Wireless Networks,” PhD thesis, Massachusetts Inst. of Technology, June 2000.

[4]. S. Lindsey, C. S. Raghavendra, “PEGASIS: Power Efficient GAthering in Sensor Information Systems,” Submitted to ICC 2001.

[5]. Wireless Sensor Networks, John A. Stankovic, University of Virginia, June 19, 2006

[6]. M. Mauve, J. Widmer, and H. Hartenstein, “A Survey on Position-Based Routing in Mobile Ad Hoc Networks,” IEEE Network, Nov./Dec. 2001, pp. 30-39.

[7]. Jing Li, Naijie Gu, and Weijia Jia, “An Efficient Fibonacci Series Based Hierarchical Application-Layer Multicast Protocol”, Springer-Verlag Berlin Heidelberg 2006.

[8]. Akkaya, Eltoweissy, Mohamed Younis and, Wadaa, “On Handling QoS Traffic in Wireless Sensor Networks” IEEE-04.

[9]. Sung-Min Jung, Young-ju Han,Tai-Myoung Chung, .”The Concentric Clustering Scheme for Efficient Energy Consumption in the PEGASIS”ICACT 2007.

[10]. Qicai shi, Spyros, Neiyer S.Correal and Feng Niu, “Performance Analysis of Relative location Estimation for Multihop Wireless Sensor Networks” IEEE journal on selected areas in communications,vol.23,no.4,april 2005.

[11]. Amit Sinha, Anantha Chandrakasan ,”Dynamic Power Management in Wireless Sensor Networks”,IEEE Design & Test of Computers 2001.

[12]. Chee-Yee Chong, Srikanta P.Kumar, “Sensor Networks:Evolutaion, Opportunities and Challenges, IEEE 2003.

[13]. Dragos Niculescu,”Communication Paradigms for Sensor Networks”, IEEE Communication Magazine 2005.

[14]. Stephanie Lindsey, Cauligi Raghavendra, Krishna M.Sivalingam “Data Gathering Algorithms in Sensor Networks Using Energy Matrics, IEEE Transaction On Parallel and Distributed Systems,vol 13, no.9, 2002.

[15]. W.Heinzelman, A. Chandrakasan, and H. Balakrishan, “ Energy-efficient communication protocol for wireless sensor networks” proc. Of Hawaii International Conference System science, Hawaii, January 2000.

[16]. S.Tilak et al., “ A Taxonomy of Wireless Microsensor Network Models,” ACM Mobile Computing and Comm. Review (MC2R),June 2002.

[17]. J.M.Rabaey, et al., “PicoRadio supports ad hoc ultra low power wireless networking,” IEEE computer, Vol. 33, pp. 42-48,July.

[18].K.Akkaya, M.Younis, “ A Survey on Routing Protocols for Wireless sensor Networks,” Deptt. Of Computer Science and Electrical Engineering, University of Maryland, Baltimore,MD, 2003.

[19]. www.wikipedia.org

APPENDIX A

CODE FOR DELAY AND ENERGY ANALYSIS IN PEGASIS

```
#include<stdio.h>
#include<conio.h>
#define MAX 100
int main()
{
        int n[MAX];

        clrscr();

        printf("number of nodes\t delay\t energy*delay\n");
        for(int i=2; i<MAX; i++) n[i] = i;

        for (i=2; i<MAX; i++)
                printf("%d\t\t   %d\t\t%d\n", n[i], n[i-1], (n[i-1]*n[i-1]));
        getch();
        return 0;
}
```

CODE FOR DELAY AND ENERGY ANALYSIS IN CHAIN BASED BINARY SCHEME

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define MAX 100
int main() {
        int n[MAX], chainBasedBinary[MAX], energyDelay[MAX];

        clrscr();
        for(int i=1; i<MAX; i++) n[i] = i;
        for (i=1; i<MAX; i++) {
                double temp =  (log10(n[i]) / log10(2));
                chainBasedBinary[i] = ceil(temp);
                energyDelay[i] = ceil ( n[i] * n[i] * temp / 2);
                printf("For node %d, ChainBasedBinary_Delay = %d, Energy*Delay = 
%d\n",n[i], chainBasedBinary[i], energyDelay[i]);
        }
        getch();
        return 0;
}
```

CODE FOR DELAY AND ENERGY ANALYSIS IN FIBOONACCI SERIES BASED ENERGY AWARE ALGORITHM

```
/*Fibonacci multicast tree*/
#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include <conio.h>
#define MAX 100
int fib[20];   //arrary to contain first 19 fibonacci numbers
//fillfib() fills fib[]
float fibo[MAX];
int getFib(int node);

void fillfib()
{
   int sizfib, j;
   sizfib = sizeof(fib)/sizeof(int);
   fib[0] = 1;
```

```
    fib[1] = 1;
    for( j=2; j<=sizfib; j++ )
    {
        fib[j] = fib[j-1] + fib[j-2];
    }
}
//largefibsearch(num) searches largest fibonacci number less than num
int largefibsearch( int searchnum )
{
    int low, high ,mid;
    searchnum--;
    high = (sizeof(fib)/sizeof(int)) - 1;
    low = 0;

    //binary search used to search array
    while( low < high )
    {
        mid = ( low + high ) / 2;
        if( searchnum < fib[mid] )
            high = mid-1 ;
        else
            low = mid+1 ;
    }
    if( fib[high] > searchnum )
        return fib[high-1];
    else
        return fib[high];
}
struct tree
{
    int val;    //number of nodes
```

```
  int dwt;       //calculated final value for val nodes
  struct tree *right;    //right fibonacci subtree
  struct tree *left;     //left fibonacci  subtree
};
//calc[culate]dwtl[eft]part will calculate and form all nodes
//of left partition subtree
int calcdwtlpart( struct tree *parent )
{
  int distwt;
  struct tree *temp;
  if( parent->val == 3 )
      parent->dwt = 2;
  else if( parent->val == 2 )
        parent->dwt = 1;
      else if( parent->val == 1 )
            parent->dwt = 0;
        else //make rt & lt son
        {
            temp = ( struct tree * ) malloc( sizeof( struct tree ) );
            parent->left = temp;
            temp = ( struct tree * ) malloc( sizeof( struct tree ) );
            parent->right = temp;

            parent->left->val = largefibsearch( parent->val );
            parent->right->val = parent->val - parent->left->val;

            parent->left->dwt = calcdwtlpart( parent->left ); //assign not needed
            parent->right->dwt = calcdwtlpart( parent->right );

            distwt = pow( parent->right->val, 2 );
            parent->dwt = parent->left->dwt + parent->right->dwt + distwt ;
            }
   return parent->dwt ;
}
//calc[culate]dwtr[ight]part will calculate and form all nodes
//of right partition subtree
int calcdwtrpart( struct tree *parent )
{
   int distwt;
   struct tree *temp;

   if( parent->val == 3 )
         parent->dwt = 2;    //2d^2
   else if( parent->val == 2 )
            parent->dwt = 1;     //1d^2
         else if( parent->val == 1 )
                  parent->dwt = 0;
            else //make rt & lt son
            {
                  temp = ( struct tree * ) malloc( sizeof( struct tree ) );
                  parent->left = temp;
                  temp = ( struct tree * ) malloc( sizeof( struct tree ) );
                  parent->right = temp;

                  parent->right->val =largefibsearch( parent->val );
                  parent->left->val = parent->val - parent->right->val;
                  parent->left->dwt = calcdwtlpart( parent->left ); //assign not needed
                  parent->right->dwt = calcdwtlpart( parent->right );

                  distwt = pow( parent->left->val, 2 );

                  parent->dwt = parent->left->dwt + parent->right->dwt + distwt ;
```

```
           }
   return parent->dwt ;
}
int main( int argc, char **argv )
{
   struct tree rootleft;
   struct tree rootright;
   int numnodes; //number of nodes for which calculation to be done
   int i, j, temp, delay[MAX];
        clrscr();
        fibo[0] = fibo[1] = 1;
        delay[0] = delay[1] = 0;
        delay[2] = 1; delay[3] = 2;
        for(i=2; i<MAX; i++)
                fibo[i] = fibo[i-1] + fibo[i-2];
        for(i=3; i<MAX; i++) {
            temp = getFib(i);
            if(temp==0)
                delay[i] = delay[i-1];
            else
                delay[i] = delay[i-1] + 1;
        }
   fillfib();
   //printf("\nnumnode\tleftsubtree\trightsubtree\ttotal\n");
   printf("\nnumNode\tDelay\tEnergy\tEnergy*Delay\n");
   for( numnodes=2; numnodes<=MAX; numnodes++ )
   {
        rootleft.val = largefibsearch( numnodes );
        rootleft.dwt = calcdwtlpart( &rootleft );
        rootright.val = numnodes - rootleft.val;
        rootright.dwt = calcdwtrpart( &rootright );
```

```
            printf("%4d\t", numnodes) ;
            //printf("%8d\t", rootleft.dwt );
            //printf("%9d\t", rootright.dwt );
            printf("%8d\t", delay[numnodes]);
            printf("%4d\t", rootright.dwt+rootleft.dwt+1 );//1 added to add 1d^2
            printf("%8d\n", delay[numnodes] * (rootright.dwt+rootleft.dwt+1) );
    }
    //return 0;
    getch();
}
int getFib(int node) {
            for(int i=0; i<MAX; i++) {
                        if (node == (fibo[i]+1) )
                                    return fibo[i];
                        else if(node < fibo[i])
                                    return 0;
            }
}
```