SIMULATION AND COMPARISON OF ROUTING PROTOCOLS OF MULTI-MEDIA WIRELESS SENSOR NETWORKS USING NS-2

By
GC Ahsan Shamim
NC Ali Ahmad Jabbar
GC Qasim Iftikhar
NC Musa Mazhar

Submitted to the Faculty of Electrical Engineering
National University of Sciences and Technology, Rawalpindi in partial fulfillment of B.E. degree in Telecommunication Engineering
June 2010

ABSTRACT

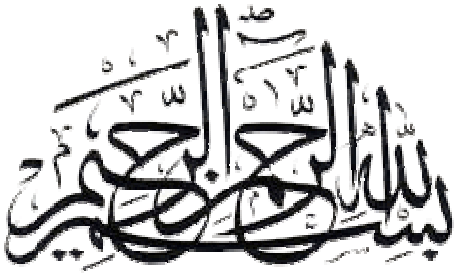
SIMULATION AND COMPARISON OF ROUTING PROTOCOLS OF MULTI-MEDIA WIRELESS SENSOR NETWORKS USING NS-2

By

GC Ahsan Shamim
NC Ali Ahmad Jabbar
GC Qasim Iftikhar
NC Musa Mazhar

The multi-media communication in Wireless Sensor Networks has got vital importance these days particularly in military applications for surveillance. Since, the nodes are deployed so randomly, making sure that communication is up and running in every scenario, is a challenging task.

The communication among the nodes (sensors) in WSNs has posed different issues at different OSI layers. Therefore, there is a need to bring efficiency in the working of protocols at these layers. But amongst them, routing layer has issues of unique kind which can be resolved only by keeping in view of the different constraints of this network like low energy consumption, and low latency, particularly for multi-media communication. We have set our goal towards routing. For that, we will simulate and compare certain routing protocols, specific to Sensor Networks, and compare their results. We will take the energy dissipation, average network latency and throughput as the deciding parameters for the efficiency of the given protocols



Dedication

All our efforts are dedicated to our beloved parents and teachers who have been a constant source of encouragement for us throughout our lives. May Allah bless them with long lives and always provide us their loving and thorough guidance.

Ameen

Acknowledgements

We are grateful to our Lord, who gave us potential to meet the challenges of this project and our parents who encouraged us throughout our project. We pay gratitude to Major Dr. Faisal Khan and Lec Obaid Khalid for their sincere supervision and effort they made in helping us complete this project in time. We would also like to acknowledge Dr. Faisal Bashir for his guidance during the course of this project.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1

1.1. INTRODUCTION

The need to monitor and measure various physical phenomena (e.g. temperature, fluid levels, vibration, strain, humidity, acidity, pumps, generators to manufacturing lines, aviation, building maintenance and so forth) is common to many areas including structural engineering, agriculture and forestry, healthcare, logistics and transportation and military applications. Wired sensor networks have long been used to support such

environments and, until recently, wireless sensors have been used only when a wired infrastructure is infeasible, such as in remote and hostile locations. But the cost of installing, terminating, testing, maintaining, trouble-shooting, and upgrading a wired network makes wireless systems potentially attractive alternatives for general scenarios. Recent advances in technology have made possible the production of intelligent, autonomous, and energy efficient sensors that can be deployed in large numbers to form self organizing and self healing WSNs in a geographical area.

Moreover, the dramatic reduction in the cost of this wireless sensor technology has made its widespread deployment feasible, and the urgent need for research into all aspects of WSNs has become evident. The WSN has great, long-term potential for transforming our daily lives, if we can solve the associated research problems.

The sensors that, when distributed in the environment, comprise WSNs include cameras as vision sensors, microphones as audio sensors, and those capable of sensing ultrasound, infra-red, temperature, humidity, noise, pressure and vibration. Although the individual sensor's sensing range is limited, WSNs can cover a large space by integrating data from many sensors. Diverse and precise information on the environment may thus be obtained. Sensor networks are an emerging computing platform consisting of large numbers of small, low-powered, wireless motes each with limited computation, sensing, and communication abilities. It is still a challenge to realize a distributed WSN comprising: small and cost effective sensor modules; high speed, low latency and reliable network infrastructures; software platforms that support

easy and efficient installation of the WSN; and sensor information processing technologies.

WSNs could potentially become a disruptive technology, for example because of social issues such as security and privacy, but the technological vision is for new and diverse types of applications for the social good..

But sensor network programming is difficult, and the human programming resource is costly. Complexities arise from the limited capabilities (processing, storage and transmission range) and energy resources of each node as well as the lack of reliability of the radio channel. As a result, application designers must make many complex, low-level choices, and build up software to perform routing, time synchronization, node localization and data aggregation within the sensor network.

Unfortunately, little of this software carries over directly from one application to another, since it encapsulates application-specific tradeoffs in terms of complexity, resource usage, and communication patterns. No WSN application will therefore be seen as typical, and application-dependency will be higher than in traditional distributed applications.

1.2. WIRELESS SENSOR NETWORKS

A WSN is a collection of millimeter-scale, self-contained, micro-electro-mechanical devices. These tiny devices have sensors, computational processing ability (i.e CPU power), wireless reception and transmitting technology and a power supply. In a WSN a large number of sensor nodes usually span a physical geographic area. For example, the prototype of a future sensor node (mote) in the Smart Dust project [188], performs the

wireless communication function, the sensor function, the power supply unit, and the information processing function on the MEMS (Micro Electro Mechanical System) chip, which has a scale only of several millimeters.

Typical WSNs communicate directly with a centralized controller or a satellite, thus communication between the sensor and controllers is based on a single hop. In future, a WSN could be a collection of autonomous nodes or terminals that communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner by forming an ad hoc network.

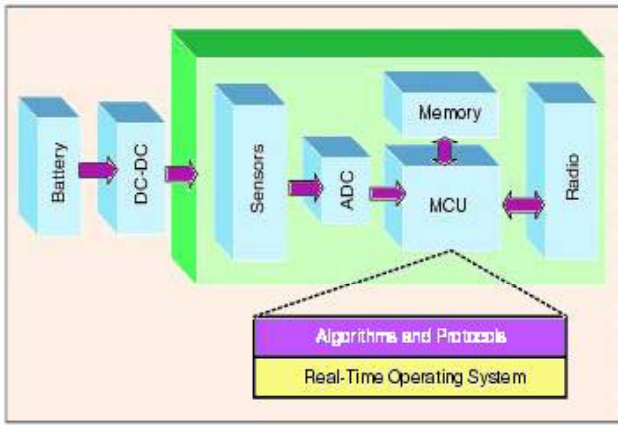


Figure 1 - WSN Node architecture and a real example

Such WSNs could change their topology dynamically when connectivity among the nodes varies with time due to node mobility. But current, real-world deployment usually consists of stationary sensor nodes.

WSNs are intelligent compared with traditional sensors, and some WSNs are designed to use in-network processing, where sensed data can be gathered in situ and transformed to more abstract and aggregated high-level data before transmission. The

combination of processing power, storage and wireless communication also means that data can be assimilated and disseminated using smart algorithms.

The vast number of sensor nodes planned for many applications also implies a major portion of these networks would have to acquire self organization capability. Intuitively, a denser infrastructure would create a more effective sensor network. It can provide higher accuracy and has more energy available for aggregation. If not properly handled, a denser network can also lead to collisions during transmission, and network congestion. This will no doubt increase latency and reduce efficiency in terms of energy consumption. One distinguishing characteristic of WSNs is their lack of strong boundaries between sensing, communication and computation. Unlike the Internet, where data generation is mostly the province of end points, in sensor networks every node is both a router and a data source.

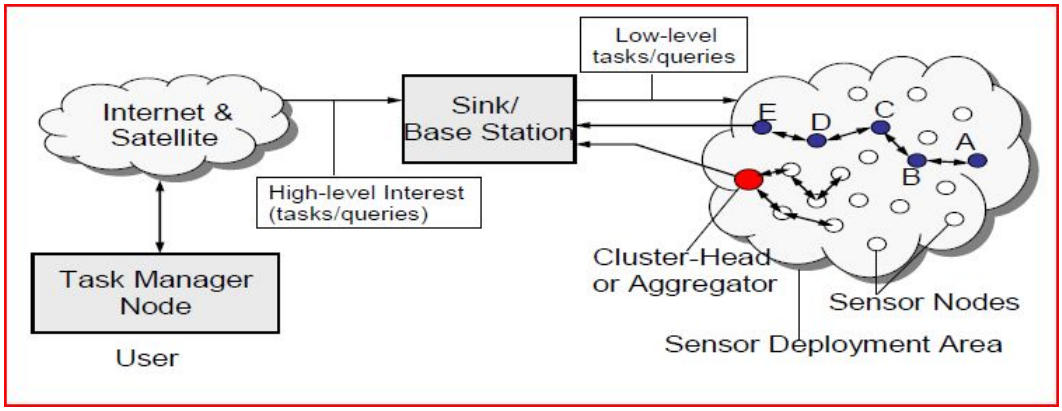


Figure 2 - Wireless Sensor Network

In contrast to multi-threaded/multi-process general-purpose operating systems, WSN nodes use less complex operating systems and event-driven programming models. In

contrast to modern operating systems, which consist of millions of lines of code, WSN operating systems codes consists of just a few thousands of lines. Some examples of WSN node operating systems are:

- TinyOS
- Contiki
- MANTIS
- BTnut
- SOS
- Nano-RK

Also it should be considered that, since WSN node hardware is similar to embedded systems, it is possible to use some embedded operating systems such as eCos, uC/OS for sensor networks.

1.3. WSNS VS MANETS

There are two major types of wireless ad hoc networks: mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs). A MANET is an autonomous collection of mobile routers (and associated hosts) connected by bandwidth-constrained wireless links. Each node is envisioned as a personal information appliance such as a personal digital assistant (PDA) fitted out with a fairly sophisticated radio transceiver. The nodes are fully mobile.

The network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet.

Factors, such as variable wireless link quality, propagation path loss, fading, multiuser interference, power expended, and topological changes, significantly increase the complexity of designing network protocols for MANETs. Security, latency, reliability, intentional jamming, and recovery from failure are also of significant concern.

A WSN, on the other hand, consists of a number of sensors spread across a geographical area. Each sensor has wireless communication capability and sufficient intelligence for signal processing and networking of the data. A WSN can be deployed in remote geographical locations and requires minimal setup and administration costs. Moreover, the integration of a WSN with a bigger network such as the Internet or a wireless infrastructure network increases the coverage area and potential application domain of the ad hoc network.

Sensed information is relayed to a sink node by using multi hop communication. The sink node is a sensor node with gateway functions to link to external networks such as the Internet and sensed information is normally distributed via the sink node (Fig.2).

WSNs differ from MANETs in many fundamental ways. Viewing a WSN as a large-scale multi-hop ad hoc network may not be appropriate for many real-world applications. The communication overhead for configuring the network into an operational state is too large. The number of nodes in a WSN can be several orders of magnitude higher than the nodes in an ad hoc network and sensor nodes that are prone to failure are densely deployed. Sensor nodes mainly use broadcast, while most MANETs are based on the Peer-to-Peer (P2P) communication paradigm. Information exchange between end-to-end nodes will be rare in WSNs. They are limited in power, computational capacity and

memory, and may not have global IDs. WSNs have a wide range of applications ranging from monitoring environments, sensitive installations, and remote data collection and analysis. In both MANETs and WSNs the nodes act both as hosts and as routers. They operate in a self organizing and adapting manner. Research and development in the areas of infrastructure-less wireless networks have been advancing at a fast pace, and more effort needs to be dedicated in this direction for wide scale adoption and deployment. Current sensor hardware is resource and power constrained, but evolution of hardware and cost reduction will be improve rapidly. WSNs may eventually share the properties of MANETs.

Processing sensor data: Processing the data gathered by sensors distinguishes sensor networks from MANETs. The end goal is the detection/estimation of some events of interest, and not just communication. To handle and improve the detection performance, it is useful to perform fusion on the data from a single or multiple sensors. Data fusion requires the transmission of data and control messages and can be part of the WSN architecture. Collaborative sensor-data processing is another factor that distinguishes WSNs from MANETs. To improve the detection rate of events of interest it is often useful to aggregate data from multiple sensors. This, again, requires the transmission of data and control messages, and may put constraints on the network architecture.

1.4. Potential Applications

Current applications in research prototype environments or industry may be classified into the four categories below:

1.4.1. Disaster/Crime Prevention and Military Applications

Sensor networks are used for environmental monitoring: to detect distortion and structural problems in buildings in order to prevent disasters. Radiation sensors can be deployed in urban districts, for example at key road intersections, to detect terrorist attacks using radioactive substances. Military WSNs can detect information about enemy movements, explosions, and other phenomena of interest.

1.4.2. Environmental Applications

WSNs can be used to detect and monitor regional environmental changes in plains, forests, oceans, flood-levels, precision agriculture etc. The common characteristic of these applications is that sensor data are aggregated in the servers and distributed with the location, and other environmental information, to the users. An example is GlacsWeb [153] which monitors glaciers in order to observe changes, possibly caused by global warming.

1.4.3. Health Applications

Tele-monitoring of human physiological data, tracking and monitoring patients and doctors inside a hospital, and assistance of the elderly are in this category. Wearable and implantable sensors can monitor a broad variety of conditions of the patients continuously at all times. Tele-monitoring may also enable testing and commencement of treatment in remote locations, as well as assisting in the precise location of accident or disaster sites.

1.4.4. OTHER COMMERCIAL APPLICATIONS:

In addition to all of above, there are many commercial applications of WSNs including

Home automation for smart home environments

Interactive museums

Environmental control in buildings

Detecting and monitoring burglary/ theft

Vehicle tracking and detection

Managing inventory control

CHAPTER 2

2.1. LITERATURE REVIEW

INTRODUCTION

Clustering, or unsupervised learning, is the task of grouping together related data objects . Unlike supervised learning, there isn't a predefined set of discrete classes to assign the objects to. Instead, new classes, in this case called clusters, have to be found. In this chapter different clustering techniques are discussed. And other issues related to clustering.

CLUSTERING

Clustering in WSNs involves grouping nodes into clusters and

Electing a CH such that:

• The members of a cluster can communicate with their CH directly.

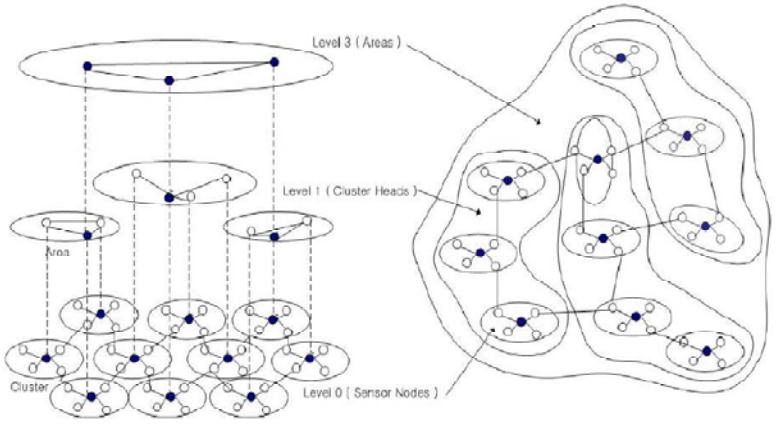
• A CH can forward the aggregated data to the central base station through other CHs.

There are a lot of possible definitions for what a cluster is, but most of them

Are based on two properties: objects in the same cluster should be related to each other, while objects in different clusters should be different. Clustering is a very natural way for humans to discover new patterns, having been studied since the ancient times. If it is regarded as unsupervised learning, clustering is one of the basic tasks of machine learning, but it is used in a whole range of other domains: data mining, statistics, pattern recognition or bioinformatics.

HIERARCHICAL CLUSTERING

The sensor nodes are organized into a hierarchy, based on their power levels and proximity. A cluster head is elected to perform various functions; with ability for re-initiation should the cluster head fail.



level hierarchical cluster-based network

Hierarchical clustering builds a tree of clusters, known as a dendrogram. A hierarchical clustering algorithm can either be top-down (divisive) or bottom-up (agglomerative). The top-down approach starts with one cluster containing all the points from a set, and then splits it iteratively until a stopping condition has been fulfilled. Agglomerative clustering begins with singleton clusters, and then, at each step, two clusters are merged. The

clusters to be joined are chosen using a predefined measure. Again, a stopping criterion should be specified. The advantage of hierarchical clustering is that the clusters can be easily visualized using the dendrogram, and is very suitable for data that inherently contains hierarchies.

To compute the distance between two clusters, as opposed to individual points, a measure called linkage metric must be used. The most used linkage metrics are:

single link (the minimum distance between any two points from the clusters)

complete link (the maximum distance between any two points from the clusters)

average link (the mean of the distances between two points from the clusters).

PARTITIONAL CLUSTERING

Partitioned clustering splits the data into a number of subsets. Because it is infeasible to compute all the possible combinations of subsets, a number relocation schemes are used to iteratively optimize the clusters. This means that the clusters are revisited at each step and can be refined, constituting an advantage over hierarchical clustering.

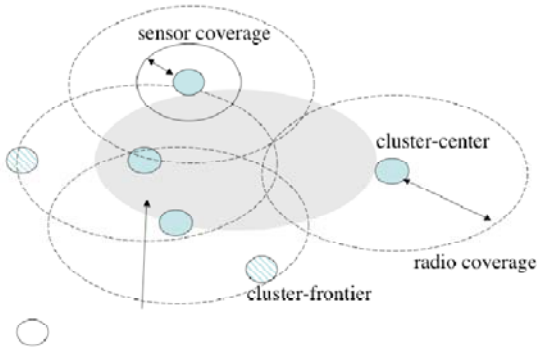
Another difference between hierarchical and partitional methods is the way the distances between clusters are calculated. The former approach computes the distances

between each individual points of a group, while the latter uses a single representative for each cluster.

CLUSTER PROCESSING

A cluster protocol where the current sensing task is transmitted from the active nodes to nodes that enter the area of interest . The nodes states in the cluster protocol are defined as cluster-centers, cluster-frontiers and idle nodes. Nodes in the cluster center state are nodes that are currently actively gathering data. To notice surrounding nodes about their state, these nodes transmit a HELLO beacon that includes the sensing task at repeated intervals. Nodes that receive this beacon, but are not cluster-centers themselves enter the cluster-frontier state. The role of the cluster frontier is to continuously monitor whether it is near enough to an event so that it can gather data about it. The hot-zone cluster concept As the cluster-frontiers are only one-hop away from the cluster-centers these nodes are close to the area of interest and are therefore candidates for future cluster centers, as they might move inside the range of the event.

Idle nodes are all other nodes. These nodes do not take part in any current sensor activities; in fact they are not even aware of any current sensor tasks at all. The only requirement for idle nodes taking part in the protocol is that they from time to time listen on the wireless channels that are used for the cluster-centers to transmit their HELLO beacons.



THE HOT ZONE CLUSTER CONCEPT

ROTATING THE ROLE OF CLUSTER HEADS

It is essential to rotate the role of CHs among nodes so as not to burden a few nodes with more duties than others. There are several possibilities for CH rotation. One way is to use a timer expiration to trigger the clustering algorithm. Another way is to use a dynamic parameter (e.g., remaining battery) for triggering the clustering algorithm at local regions. For example, a CH might trigger a new CH election process in its local region if its remaining battery lifetime goes below a pre specified threshold. The CH rotation mechanism is typically independent of the clustering protocol. It is obvious that more frequent CH rotation results in more clustering overhead and network interruption, while less frequent rotation may cause some nodes to die faster than others. The study of this trade-off is essential for achieving optimal network lifetime. Currently, applications set this rotation frequency heuristically, based on some intuitive factors such as the expected battery lifetime and the node degree.

CONNECTIVITY

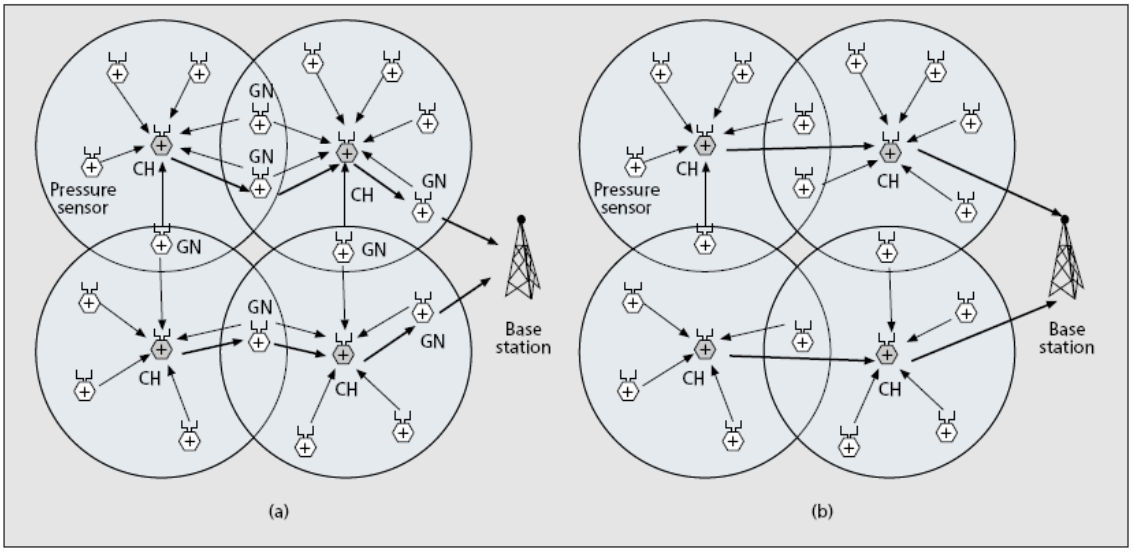
An important objective of any clustering technique is network connectivity. For intra cluster communication, a cluster member communicates with its CH either directly or via multiple hops .Connectivity in this case is a result of the success of cluster formation. For inter cluster communication, two approaches were adopted in order to maintain connectivity. In one approach nodes on cluster boundaries are used as gateways to relay data among CHs.

This approach is suitable in networks that use a fixed transmission power. Network density has to be sufficiently high in order to ensure that enough gateways are present at the intersection areas between clusters. In another approach , the CH overlay constitutes the routing infrastructure, and inter cluster routing proceeds only through CHs. This approach is appropriate if:

• A node can tune its transmission power (e.g., Berkeley motes).

• The CH density and inter cluster transmission range satisfy the connectivity conditions.

An advantage of this approach is that it enables all non-CHs to sleep while not sensing or transmitting. Selecting the optimal intra cluster and inter cluster transmission ranges to ensure connectivity and prolong the network lifetime is still an open issue.

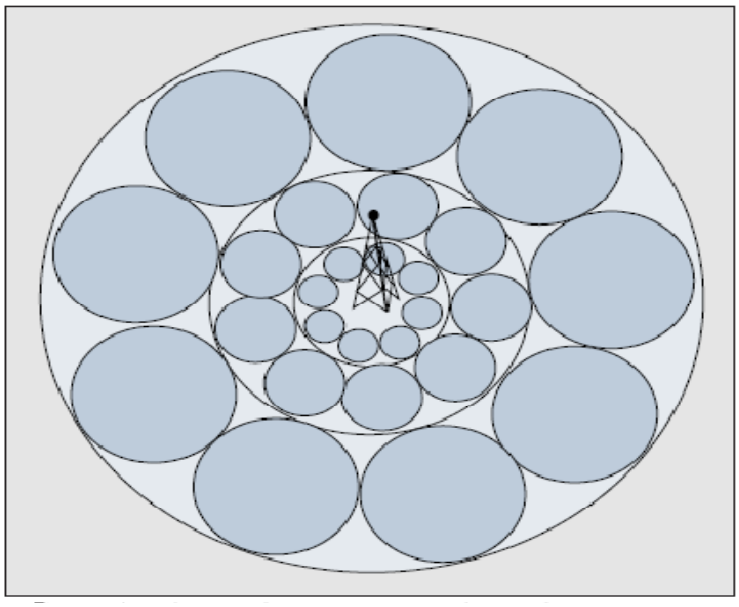


Intercluster connectivity models: a) routing via gateway nodes; b) routing via CHs only. GN: gateway node.

OPTIMAL CLUSTER SIZE

Currently, most clustering protocols assume a fixed cluster transmission range, which results in uniform cluster sizes. However, this results in a skewed load distribution on CHs (e.g., CHs that are closer to the observer will carry more inter cluster traffic and hence will deplete their batteries faster than faraway CHs). In order to balance energy consumption among CHs, both intra cluster aggregation and the data forwarding responsibilities of a CH have to be considered. Studied how to achieve this balance by assigning larger cluster sizes to CHs that are responsible for less data forwarding. An example is illustrated in Figure below in which sensors are deployed in a circular region around a base station. This approach is constrained by the maximum possible sensor transmission range, and the supporting MAC layer. Several issues are still open for

research, including how to optimally select cluster sizes without knowledge of the node locations and how to exploit knowledge of the locations of data sinks for efficient cluster formation.



SELECTING CLUSTER SIZES TO EVENLY DISTRIBUTE ENERGY AMONG CLUSTER HEADS.

NODE SYNCHRONIZATION

Distributed clustering protocols achieve their best performance when sensor nodes are synchronized. Node synchronization ensures that the clustering process starts simultaneously throughout the network. Lack of synchronization may result in a suboptimal choice of CHs, especially for probabilistic approaches. However, the clustering process can be triggered by nodes with faster clocks. This happens when such nodes start querying their neighbors for updated information in order to start the clustering process. Received queries trigger the clustering process at these neighbors, and these neighbors in turn trigger their neighbors, and so on. Note that it is not

essential that “all” the nodes in the network start the clustering process simultaneously. In fact, it is sufficient that the process starts in different regions (nodes within one or two hops) simultaneously. Probabilistic techniques that use a number of iterations are therefore less impacted by the lack of synchronization than single-iteration techniques.

ATTRIBUTE-BASED NAMING

The sensor nodes are named based on their attributes. For example, consider a system which is used to measure temperature at a particular location. Then, the name

[type = temperature, location=N-E, temperature = 103]

Refers to all the sensors located at the northeast quadrant with a temperature reading of 103F. Thus, they can reply when a query like "which area has a temperature more than 100F" is posed. Such a scheme works because the nodes are by themselves neither unique nor dependable. So, applications access a particular data element by naming it directly. has another advantage in that it eliminates the need for maintaining mapping/directory services, which is an extra overhead.

Most sensor data is associated with the physical context of the phenomena being sensed. Hence spatial coordinates are a natural way to name data. This makes localization - determination of the position of the node in some co-ordinate system - an important problem [rf based localization].

ENERGY EFFICIENCY

Energy consumption is the most important factor to determine the life of a sensor network because usually sensor nodes are driven by battery and have very low energy resources. This makes energy optimization more complicated in sensor networks because it involved not only reduction of energy consumption but also prolonging the life of the network as much as possible. This can be done by having energy awareness in every aspect of design and operation. This ensures that energy awareness is also incorporated into groups of communicating sensor nodes and the entire network and not only in the individual nodes.

A sensor node usually consists of four sub-systems :

A computing subsystem : It consists of a microprocessor(microcontroller unit,MCU) which is responsible for the control of the sensors and execution of communication protocols. MCU's usually operate under various operating modes for power management purposes. But shuttling between these operating modes involves consumption of power, so the energy consumption levels of the various modes should be considered while looking at the battery lifetime of each node.

A communication subsystem: It consists of a short range radio which is used to communicate with neighboring nodes and the outside world. Radios can operate under the Transmit, Receive, Idle and Sleep modes. It is important to completely shut down the radio rather than put it in the Idle mode when it is not transmitting or receiving because of the high power consumed in this mode

A sensing subsystem: It consists of a group of sensors and actuators and link the node to the outside world. Energy consumption can be reduced by using low power components and saving power at the cost of performance which is not required.

A power supply subsystem: It consists of a battery which supplies power to the node. It should be seen that the amount of power drawn from a battery is checked because if high current is drawn from a battery for a long time, the battery will die even though it could have gone on for a longer time. Usually the rated current capacity of a battery being used for a sensor node is lesser than the minimum energy consumption required leading to the lower battery lifetimes. The lifetime of a battery can be increased by reducing the current drastically or even turning it off .

The power consumed by the sensor nodes can be reduced by developing design methodologies and architectures which help in energy aware design of sensor networks. The lifetime of a sensor network can be increased significantly if the operating system, the application layer and the network protocols are designed to be energy aware. Power management in radios is very important because radio communication consumes a lot of energy during operation of the system. Another aspect of sensor nodes is that a sensor node also acts a router and a majority of the packets which the sensor receives are meant to be forwarded. Intelligent radio hardware that help in identifying and redirecting packets which need to be forwarded and in the process reduce the computing overhead because the packets are no longer processed in the intermediate nodes.

Traffic can also be distributed in such a way as to maximize the life of the network. A path should not be used continuously to forward packets regardless of how much energy is saved because this depletes the energy of the nodes on this path and there is a breach in the connectivity of the network. It is better that the load of the traffic be distributed more uniformly throughout the network.

It is important that the users be updated on the health of a sensor network because this would serve as a warning of a failure and aid in the deployment of additional sensors.

Localization

In sensor networks, nodes are deployed into an unplanned infrastructure where there is no a prior knowledge of location. The problem of estimating spatial-coordinates of the node is referred to as localization. An immediate solution which comes to mind is GPS or the Global Positioning System.

However, there are some strong factors against the usage of GPS. For one, GPS can work only outdoors. Secondly, GPS receivers are expensive and not suitable in the construction of small cheap sensor nodes. A third factor is that it cannot work in the presence of any obstruction like dense foliage etc. Thus, sensor nodes would need to have other means of establishing their positions and organizing themselves into a coordinate system without relying on an existing infrastructure.

Most of the proposed localization techniques today, depend on recursive trilateration/multilateration techniques. One way of considering sensor networks is taking the network to be organized as a hierarchy with the nodes in the upper level being more complex and already knowing their location through some technique (say,

through GPS). These nodes then act as beacons by transmitting their position periodically.

The nodes which have not yet inferred their position listen to broadcasts from these beacons and use the information from beacons with low message loss to calculate its own position. A simple technique would be to calculate its position as the centroid of all the locations it has obtained. This is called as proximity based localization. It is quite possible that all nodes do not have access to the beacons. In this case, the nodes which have obtained their position through proximity based localization themselves act as beacons to the other nodes. This process is called iterative multilateration. As can be guessed, iterative multilateration leads to accumulation of localization error. Since most of the localization algorithms use some form of trilateration, a brief overview of trilateration is given. Consider a person A, who wants to determine his position in 2-D space.

Suppose A knows that he is 10kms from a point x. Then he can determine that he is anywhere on the circle of radius 10kms around the point x. Now, if A also knows that he is 20 km/s from a point y, A can deduce that he is on either one of the two intersecting point of the circle of radius 10km around x and the circle of radius 20km around point y. Suppose A also has additional information that he is 15km from a point z. Now he knows at which of the two intersecting points he is one because only one of them will intersect with the third circle also. This is shown in figure 6 below. Let x be Boise, y be Minneapolis and z be Tucson.

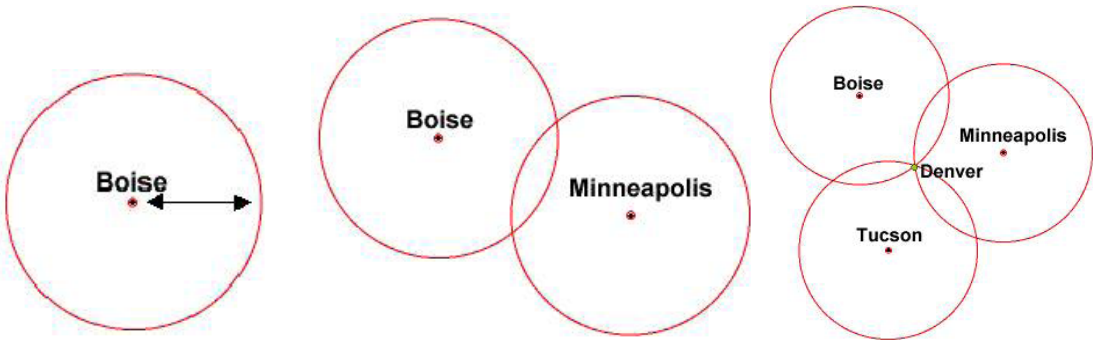


FIGURE 6 - PRINCIPLE OF TRILATERATION IN 2-D SPACE

Thus, trilateration is a geometric principle which allows us to find a location if its distance from other already-known locations are known. The same principle is extended to three-dimensional space. In this case, spheres instead of circles are used and four spheres would be needed. This is the principle used in GPS also. Figure 7 demonstrates trilateration in 3-D space as used in GPS.

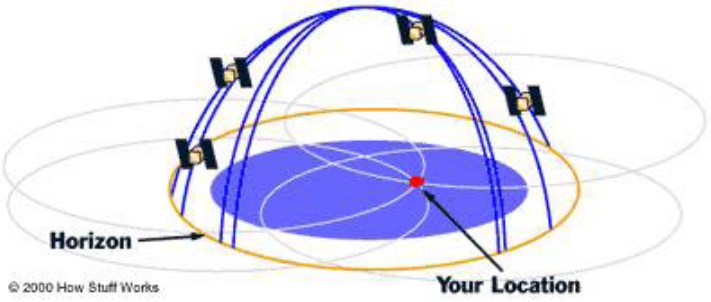


Figure 7 - Principle of trilateration in 3-D space as used in GPS.

When a localization technique using beacons is used, an important question would be 'how many initial beacons to deploy'. Too many beacons would result in self-interference among the beacons while too less number of beacons would mean that many of the nodes would have to depend on iterative multilaterationAn associated

problem would be to decide the total number of sensor nodes required in a given area. That is, determining the network density.

Localization Techniques:

Localization can be classified as fine-grained, which refers to the methods based on timing/signal strength and coarse-grained, which refers to the techniques based on proximity to a reference point.

Examples of fine-grained localization are:

Timing: The distance between the receiver node and a reference point is determined by the time of flight of the communication signal.

Signal strength: As a signal propagates, attenuation takes place proportional to the distance traveled. This fact is made use of to calculate the distance.

Signal pattern matching: In this method, the coverage area is pre-scanned with transmitting signals. A central system assigns a unique signature for each square in the location grid. The system matches a transmitting signal from a mobile transmitter with the pre-constructed database and arrives at the correct location. But pre-generating the database goes against the idea of ad hoc deployment.

Directionality: Here, the angle of each reference point with respect to the mobile node in some reference frame is used to determine the location.

Examples of coarse-grained localization are:

Proximity based localization:. RF-based transceivers would be more inexpensive and smaller compared to GPS-receivers. Also in an infrastructure less environment, the

deployment would be ad hoc and the nodes should be able to adapt themselves to available reference points.

A node, which has to calculate its position, receives signals from a collection of reference points. All these reference points have their connectivity metric above a pre-decided threshold. Connectivity metric is defined as the ratio of the total number of signals received by a node to the total number of signals sent by a node.

Once the node receives the signal, it calculates its position as the centroid of the positions of all the reference nodes as :

$$(X_{est}, Y_{est}) = ((X_{i1}+\ldots+X_{ik})/k, (Y_{i1}+\ldots+Y_{ik})/k))$$

where Xi1, Yi1 gives the position of the first reference point, Xi2, Yi2 gives the position of the second reference point and so on. The accuracy of the estimate can be determined by calculating the localization error.

$$LE= ((X_{est}-X_a)^2 + (Y_{est}-Y_a)^2)^{½}$$

By increasing the range overlap of reference points, the accuracy of the location estimate improves.

2.1.1. Classification of Routing Protocols in WSNs

In this section, we survey the state-of-the-art routing protocols for WSNs. In general, routing in WSNs can be divided into flat-based routing, hierarchical-based routing, and location-based routing depending on the network structure. In flat-based routing, all nodes are typically assigned equal roles or functionality.

In hierarchical-based routing, however, nodes will play different roles in the network. In location-based routing, sensor nodes' positions are exploited to route data in the network. A routing protocol is considered adaptive if certain system parameters can be controlled in order to adapt to the current network conditions and available energy levels. Furthermore, these protocols can be classified into multipath-based, query-based, negotiation-based, QoS-based, or coherent-based routing techniques depending on the protocol operation. In addition to the above, routing protocols can be classified into three categories, namely, proactive, reactive, and hybrid protocols depending on how the source finds a route to the destination. In proactive protocols, all routes are computed before they are really needed, while in reactive protocols, routes are computed on demand. Hybrid protocols use a combination of these two ideas. When sensor nodes are static, it is preferable to have table driven routing protocols rather than using reactive protocols. A significant amount of energy is used in route discovery and setup of reactive protocols. Another class of routing protocols is called the cooperative routing protocols. In cooperative routing, nodes send data to a central node where data can be aggregated and may be subject to further processing, hence reducing route cost in terms of energy use. Many other protocols rely on timing and position information. Now, we would show the classification of some WSN protocols according to the network structure and protocol operation (routing criteria).

DATA-CENTRIC PROTOCOLS:

In many applications of sensor networks, it is not feasible to assign global identifiers to each node due to the sheer number of nodes deployed. Such lack of global identification along with random deployment of sensor nodes makes it hard to select a specific set of sensor nodes to be queried. Therefore, data is usually transmitted from every sensor node within the deployment region with significant redundancy. Since this is very inefficient in terms of energy consumption, routing protocols that will be able to select a set of sensor nodes and utilize data aggregation during the relaying of data have been considered. This consideration has led to data-centric routing, which is different from traditional address-based routing where routes are created between addressable nodes managed in the network layer of the communication stack.

FLOODING AND GOSSIPING

Flooding and gossiping are two classical mechanisms to relay data in sensor networks without the need for any routing algorithms and topology maintenance. In flooding, each sensor receiving a data packet broadcasts it to all of its neighbors and this process continues until the packet arrives at the destination or the maximum number of hops for the packet is reached. On the other hand, gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly selected neighbor, which picks another random neighbor to forward the packet to and so on.

LOCATION-BASED ROUTING

Most of the routing protocols for sensor networks require location information for sensor nodes. In most cases location information is needed in order to calculate the distance between two particular nodes so that energy consumption can be estimated. Since, there is no addressing scheme for sensor networks like IP-addresses and they are

spatially deployed on a region, location information can be utilized in routing data in an energy efficient way. For instance, if the region to be sensed is known, using the location of sensors, the query can be diffused only to that particular region which will eliminate the number of transmission significantly.

CONCLUSION

CLUSTERING PROTOCOLS

We have simulated protocols that do clustering in which each one or a few nodes from a region transmit data on behalf of other nodes to avoid load and redundant delivery of data to a given Base Station called Sink.

OUR SIMULATED PROTOCOLS:

*HEED

*UCR

*CAODV

*LDCFR

OUR WORK

As multiple constraints of Wireless Multimedia sensor Networks have been discussed. The over-riding issue basic issue is to devise some scheme of protocols that incur less routing over-head, provide high throughput and fulfill low latency requirements.

So, keeping in view the required objectives of low routing over-head, high throughput and low latency, we have simulated four routing protocols and compared their results in terms of these parameters. All of these protocols do clustering before routing.

These protocols are:

- ⇒ LDCFR (Location-based Distributed Clustering with Flooding-based Routing)
- ⇒ UCR (Un-equal Clustered Routing)
- ⇒ HEED (Hybrid Energy Efficient Distributed clustering)
- ⇒ CAODV (Clustered Ad-hoc On-demand Distance Vector)

CHAPTER 3

3.1 LOCATION-BASED DISTRIBUTED CLUSTERING WITH FLOODING-BASED ROUTING (LDCFR)

Introduction

3.1.1 TOPOLOGY ENVIRONMENT:

Assume that N sensor nodes have been deployed which are continuously monitoring the environment. As soon as any event occurs, they sense the parameters, and send the data to the sink only after clustering is done.

We would make a few assumptions about our deployed nodes in the given WSN network.

1) There is a base station (i.e. data sink) located far from the sensing field. Sensors and stations are all stationary after deployment.

2) Sensors are homogeneous and have the same capabilities. Each node is assigned a unique identifier (ID).

3) Links are symmetric. A node can compute the approximate distance to another node based on received signal strength, if transmitting power is known.

4) A node is able to determine is location through GPS and energy as it sends data to the other node, taken from the fact the energy spent for transmission of an l-bit packet over distance d is:

$$E_{tx}(l,d) = lE_{elec} + l\epsilon_{fs} d^{\alpha} \quad \text{where } \alpha=2 \text{ for } d<d_o \text{ and } \alpha = 4 \text{ for } d \geq d_o$$

The electronics energy Eelec, depends on factors such as the digital coding, and modulation, whereas the amplifier energy, $\epsilon_{fs}d^2$ depends on the transmission distance and the acceptable bit-error rate.

5) The cluster-head doesn't aggregate the incoming packets because it is assumed that the correlation degree of sensed data from different clusters is comparatively low.

As it has been already discussed that our goal in improving the efficiency is set on reducing cluster formation over-head i.e. over-head data that is generated during the formation of clusters and cluster-heads.

Normally, the first phase in most scenarios is forming cluster heads. Second, cluster-heads send some broad-cast packets to announce themselves as heads for a particular cluster. The nodes receiving the given head-packet reply and hence, in the end associate with the given head. This happens for every cluster and in the end; we get a

synchronized topology that is capable to transmitting data with reduced latency and greater throughput compared to a flat topology.

But, there are some ambiguities in the kind of cluster-formation we have discussed above. First, there is a decision to be made as to which nodes will become cluster-heads or which not. Second, if cluster-heads are formed, they announce themselves as heads using broadcast packets which may waste energy on nodes that should not accept this packet because they have some other head closer to them than the given one. This is to be considered keeping in view the fact that we can't ensure that the packet reception from the closer head should be done first than from the farther one. Third, the synchronization time is large enough that we may lose our data traffic during this bigger interval.

So, our focus is to reduce the synchronization time by using innovative technique so that cluster-formation and cluster-rotation take very small time and our WSN is synchronized as quickly as possible for the routing of data. Thus, in order to do so, we have resorted to a mix of clustering and location-based routing approach.

3.1.2 CLUSTER FORMATION

We will break our cluster-formation algorithm into a few steps to make the working clearer:

1) Divide the WSN topology into a virtual grid of a particular size and naming each sector as 1, 2, 3 and so on.

2) Every node finds its location on the virtual grid using GPS

3) Associate each node with particular sector or zone, depending on the location of the node

4) At first, a node in a particular zone, will only communicate with node closer to the center of that zone, this node being communicated with, is called the cluster-head or zone-head for the given zone

5) This happens for every node in a particular cluster and hence, every node in every zone is associated with a zone id and zone head to forward the data to, and ultimately send it to the sink

6) If the sink is not in the range of a particular cluster-head, it will send data to it through the other cluster-heads using multihop routing otherwise send it directly to the sink

7) For the rotation of cluster-heads, we take into account the residual energy of each node and make the next cluster-head a node with the highest residual energy in a particular zone

LDCFR clustering algorithm

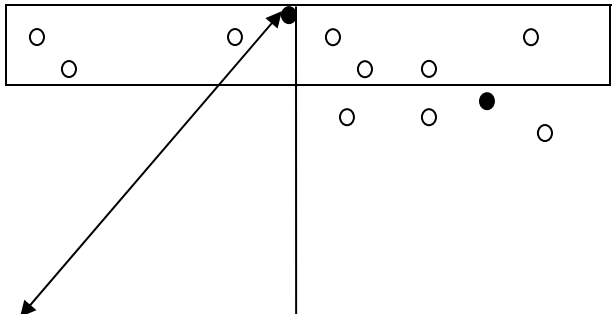
```
1: Topology Grid Division into zones
2: Every node si finds its location si.location
3: Si.location -> zone id
4: Find distance of all nodes from center of zone
5: if d(sj->center of zone 'Z') ≤ min d(sk -> center of zone 'Z')
       Select  sj as the head
   End
6: sj is the head and the next hop for si
7:  for next head in the zone
If (sj.res_energy) > (sk.res_energy)
          sj is the next head
end
8: si will set sj as the next head
```

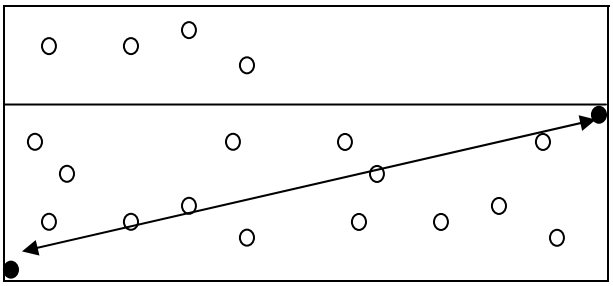
3.1.3 THE VIRTUAL TOPOLOGY AND ITS CONSTRAINTS

The set of CHs and the wireless links connecting them is modeled as a directed graph G = (V,E), where V is the set of CH nodes and E is the set of directed wireless links connecting CH nodes. Each nonempty zone corresponds to a vertex in the graph G. Vertical and horizontal adjacent vertices (CHs) are connected by an edge if the distance between them is less than or equal to the corresponding transmission range and the two adjacent zones are not empty. A link is absent from E if either one or the two neighboring zones are empty or a communication in the respected direction is not possible. Note that the CH election process selects the most eligible node in every zone to act as a CH. Thus, the resulting set of elected CHs tend to remain in that status for long time relative to other nodes in their zones resulting in stable virtual topology. Consequently, the routes found, exhibit a relatively long lifetime when the zones maintain CHs. This result proves to be helpful when performing end-to-end QoS routing.

3.1.3.1 The topology constraints

A power control scheme is needed in our network to achieve network connectivity. This scheme determines the inter-zone and intra-zone communications. In our scenario, we need to ensure the adjacent cluster to cluster head communication. For that, they should be able to communicate in the worst case distance scenario as shown in fig below, thus, the nodes should have initial power that ensures their communication in this particular scenario.





3.1.4 INTER-CLUSTER ROUTING

In WSN, we have multiple sources and single destination i.e. the sink, so, the best way is that the destination or the sink should provide itself the route to other nodes in the network. For that, a beacon message is multicast by the sink as soon as clustering is done. This packet or message comprises of a sequence no, number of hops and source id and is of only 48 bytes. The multi-cast is only received by the cluster heads. A head receiving the beacon forwards it by replacing its src_id field with its own id which in turn is received by the other cluster-heads in the network. Now, the sending head becomes the next hop for the other heads which are going to receive the beacon exclusively from the given head. The receiving head updates the next hop only if the hop count is smaller and the sequence no of the beacon is higher than the previous one it had in its routing cache.

The parameters in the beacon message are:

Pkt_type

It shows the type of message which is 'beacon' for the beacon message.

Beacon_hops

It shows the number of sensor nodes a nodes requires to pass through to send its data to the sink (Base station). The beacon containing the smaller number of hops updates the previous information about the route to the sink.

Beacon_id

It is the sequence number for each beacon message sent. It shows the freshness of the route. Since, each new beacon message generated by the sink has an increased 'seq_no'.

A head does not increment this counter while forwarding the data.

Beacon_timestamp

It stores the time at which a beacon message is sent and it is updated by each head forwarding this message.

3.1.5 CONCLUSION

It can be easily observed that inter-cluster routing depends upon only a single overhead message i.e. beacon message, thus, the protocol using this approach normally shows very little overhead for inter-clustering routing and thus has a huge influence in the average latency of packets received at the sink.

CHAPTER 4

4.1 UN-EQUAL CLUSTERED ROUTING (UCR)

Introduction

Current Clustering algorithms rarely consider the hot spot problem in multihop sensor networks. When cluster heads cooperate with each other to forward their data to the base station, the cluster heads closer to the base station are burdened with heavier relay traffic and tend to die much faster, leaving areas of the network uncovered and causing network partitions. UCR mitigates the hot spot problem. It groups the nodes into clusters of unequal sizes. Cluster heads closer to the base station have smaller cluster sizes than those farther from the base station, thus they can preserve some energy for the inter-cluster data forwarding.

4.1.1 SYSTEM MODEL

Assume that N sensor nodes have been uniformly deployed over a vast field to continuously monitor the environment. We denote the i -th sensor by si and the corresponding sensor node set $S = \{s1, s2, \ldots, sN\}$, where $|S| = N$. We make some assumptions about the sensor nodes and the underlying network model:

1. There is a base station (i.e., data sink) located far from the sensing field. Sensors and the base station are all stationary after deployment.

2. Sensors are homogeneous and have the same capabilities. Each node is assigned a unique identifier (ID).

3. Sensors are capable of operating in an active mode or a low-power sleeping mode.

4. Sensors can use power control to vary the amount of transmission power according to the distance to the desired recipient.

5. Links are symmetric. A node can compute the approximate distance to another node based on the received signal strength, if the transmitting power is known.

We used a simplified energy for the communication energy dissipation. Both the free space (*d*2 power loss) and the multi-path fading (*d*4 power loss) channel models are used, depending on the distance between the transmitter and receiver. The energy spent for transmission of an *l*-bit packet over distance *d* is:

$$ET\,x(l,\,d) = lE_{\text{elec}} + l\epsilon d^{\alpha} = \begin{cases} lE\text{elec} + l\epsilon\, f\, sd2\,, d < do \\ lE\text{elec} + l\epsilon mpd4, \quad d \geq do \end{cases} \quad \ldots\ldots (4.1)$$

The electronics energy, E_{elec}, depends on factors such as the digital coding, and modulation, whereas the amplifier energy, $\epsilon_{f\,s}d^2$ or $\epsilon_{mp}d^4$, depends on the transmission distance and the acceptable bit-error rate. To receive this message, the radio expends energy:

$$E_{Rx}\,(l) = lE_{\text{elec}} \quad \ldots\ldots (4.2)$$

It is assumed that the sensed information is highly correlated, thus the cluster head can always aggregate the data gathered from its members into a single length-fixed packet. In some algorithms, relay nodes can aggregate the incoming packets from other clusters together with its own packets. This assumption is impractical because the correlation degree of sensed data from different clusters is comparatively low. In this work, relay nodes don't aggregate the incoming packets. We assume that a cluster head consumes E_{DA} (nJ/bit/signal) amount of energy for data aggregation.

4.1.2 THE PROBLEM OF UNBALANCED ENERGY CONSUMPTION

In this work, cluster heads form a virtual backbone for inter-cluster communication. Each head node forwards the data to the base station via a multihop path through other intermediate cluster heads. The reason this is done is because multihop communication is more realistic; nodes may not be able to reach the base station due to the limited transmission range. Even if a node can use power control to send data to a farther receiver, previous research has shown that it is obviously a waste of energy. However, when multi-hop routing is adopted in inter-cluster communication, the many-to-one traffic pattern on the cluster head overlay leads to the hot spot problem.

In a clustered sensor network, each cluster head spends its energy on intra- and inter-cluster processing. The energy consumed in intra-cluster processing varies proportionally to the number of nodes within the cluster. Proposed clustering algorithms that consider then load balance issue usually produce clusters of even sizes, thus the intra-cluster load is roughly the same for all cluster heads. On the other hand, the inter-cluster traffic load of cluster heads is highly uneven. Cluster heads closer to the base station have a higher load of relay traffic. Consequently, they will die much faster than the other cluster heads, possibly reducing sensing coverage and leading to network partitioning.

A fundamental issue in wireless sensor networks is maximizing the network lifetime subject to a given energy constraint. To achieve this goal, energy consumption must be well-balanced among nodes. In homogeneous networks, the cluster head role can be periodically rotated among nodes to balance the energy dissipation. However, the hot spot problem cannot be avoided. The main objective of the rotation is to balance the

energy consumption among the sensor nodes in each cluster, and it could hardly balance the energy consumption among cluster heads in the inter-cluster multi-hop routing scenario. We also argue that using node's residual energy as the only criterion when selecting cluster heads is not sufficient to balance energy consumption across the network. Selecting cluster heads with more residual energy can only be helpful to balance energy consumption among nodes within a cluster radius in the long term. It is ineffective to balance the load among different cluster heads to avoid the hot spot problem if the cluster heads are uniformly distributed over the network. Because sensor nodes in the hot spot still die faster, it cannot make efficient use of all nodes' energy.

To mitigate the hot spot problem, unequal clustering protocol for hierarchical routing, called UCR, has been simulated and its results have been compared with certain other protocols like HEED and CAODV. Both the rotation of cluster heads and choosing cluster heads with more residual energy are adopted into the clustering algorithm EEUC. It organizes the network into clusters of unequal sizes. By decreasing the number of nodes in clusters with higher relay load near the base station, we can maintain more uniform energy consumption among cluster heads in the long run.

4.1.3 THE UNEQUAL CLUSTER-BASED ROUTING PROTOCOL

The UCR protocol consists of two parts: an energy-efficient unequal clustering algorithm called EEUC and an inter-cluster greedy geographic and energy-aware routing protocol. At the network deployment stage, the base station broadcasts a beacon signal to all sensors at a fixed power level. Therefore each sensor node can compute the approximate distance to the base station based on the received signal strength. It not

only helps nodes to select the proper power level to communicate with the base station, but also helps us to produce clusters of unequal sizes. Detailed descriptions of the unequal clustering algorithm and intra-cluster multi-hop routing protocol are in the following two subsections.

Figure 4 gives an overview of the UCR protocol, where the unequal Voronoi cells represent the unequal clusters formed by EEUC and the traffic among cluster heads illustrates our multi-hop forwarding method.

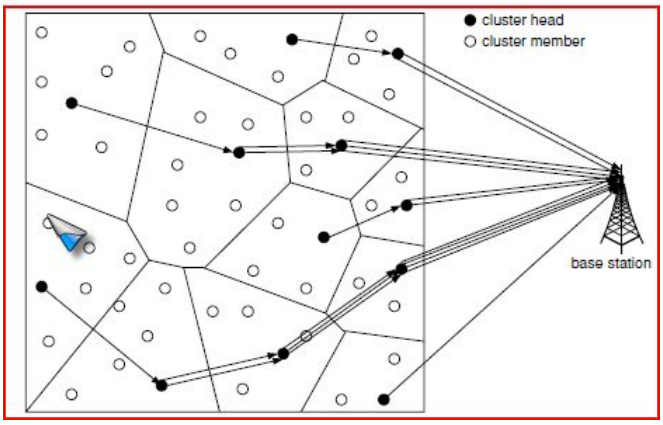


Figure 3 - An overview of UCR Protocol

4.1.3.1 Unequal clustering algorithm

Clustering a wireless sensor network means partitioning its nodes into clusters, each one with a cluster head and some ordinary nodes as its members. Similar to LEACH, the operation of UCR is divided into rounds. The task of being a cluster head is rotated among sensors in each round to distribute the energy consumption across the network. EEUC is a distributed cluster head competitive algorithm, where the cluster head selection is primarily based on the residual energy of tentative cluster heads. Furthermore, EEUC produces clusters of unequal sizes to mitigate the hot spot problem.

Clusters closer to the base station have smaller cluster sizes, thus they will consume less energy during the intra-cluster data processing, and can conserve some more energy for the inter-cluster relay traffic. The pseudocode for each sensor node at the cluster head selecting stage is given in Fig. 4.

Here we explain the clustering algorithm in detail. First, several tentative cluster heads are randomly selected to compete for final cluster heads. Ordinary nodes become tentative cluster heads with the same probability *T* which is a predefined threshold. Nodes that fail to be tentative heads keep sleeping until the cluster head selection stage ends.

Each tentative cluster head *si* has a competition range *Ri* . Different competition ranges are used to produce clusters of unequal sizes. Only one final cluster head is allowed in each competition range. If *si* becomes a cluster head at the end of the competition, there will not be another cluster head *s j* in *si* 's competition range. Figure 5 illustrates a topology of tentative cluster heads, where the circles represent different competition ranges of tentative cluster heads. In Fig. 5 both *s*1 and *s*2 can be final cluster heads, but *s*3 and *s*4 cannot.

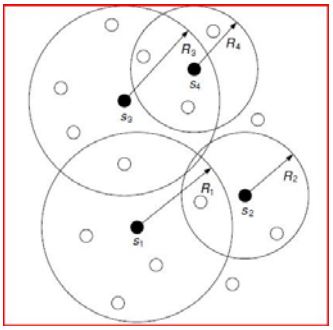


Figure 4 - The competition among tentative cluster-heads

Therefore, the distribution of cluster heads can be controlled over the network. Cluster heads closer to the base station should support smaller cluster sizes, thus more clusters need to be produced closer to the base station. That is to say, the tentative cluster head's competition range should decrease as its distance to the base station decreases. We need to select a proper scope of competition ranges in the network.

Suppose R_0 is the maximum competition range which is predefined, and the minimum competition range is set to $(1 - c)R_0$ correspondingly, where c is a constant coefficient between 0 and 1. Thus the tentative cluster head si 's competition range Ri can be expressed as a linear function of its distance to the base station:

$$R_i = \left(1 - c\frac{dmax - d(si,BS)}{dmax - dmin}\right)R_0 \quad \text{..... (4.3)}$$

where d_{max} and d_{min} denote the maximum and minimum distance between sensor nodes in the network and the base station, and $d(si\ ,\ BS)$ denotes the distance between si and the base station. According to Eq. (4.3), if c is set to $\frac{1}{3}$, Ri varies from $\frac{2}{3}R_0$ to R_0 according to the distance between si and the base station.

Each tentative cluster head maintains a set *SCH* of its "adjacent" tentative cluster heads. In lines 10–13 of Fig. 6, each tentative head constructs its *SCH*. Tentative head $s\ j$ is an "adjacent" node of si if $s\ j$ is in si 's competition diameter or si is in $s\ j$'s competition diameter. Whether a tentative cluster head si will become a final cluster head depends on the nodes in $si\ .SCH$ only, i.e., the algorithm is localized.

In the cluster head selecting algorithm, the broadcast radius of every control message is $R0$, thus si can hear all messages from nodes in its *SCH*. In line 6 of Fig. 6, each tentative cluster head broadcasts a COMPETE HEAD MSG which contains its competition radius

and residual energy (*RE*). After the construction of *SCH* has finished in lines 10–13, each tentative cluster head checks its *SCH* and makes a decision as to whether it can act as a cluster head in lines 14–26. Before deciding what its role is going to be, *si* needs to know what each node *x* in its *SCH* such that *x.RE > si .RE* has decided for itself. In case of a tie, the smaller node ID is chosen. In lines 15–17, once *si* finds that its residual energy is more than all the nodes in its *SCH*, it will win the competition and broadcast a FINAL HEAD MSG to inform its adjacent tentative cluster heads. In lines 18–21, if *s j* belongs to *si .SCH* and *si* receives a FINAL HEAD MSG from *s j* , *si* will give up the competition immediately, and inform all nodes in its *SCH* by broadcasting a QUIT ELECTION MSG. In lines 22–25, if *si* receives a QUIT ELECTION MSG form *s j* and *s j* belongs to *si .SCH*, *si* will remove *s j* from its *SCH*.

After cluster heads have been selected, sleeping nodes now wake up and each cluster head broadcasts a CH ADV MSG across the network field. Each ordinary node chooses its closest cluster head with the largest received signal strength and then informs the cluster head by sending a JOIN CLUSTER MSG. A Voronoi diagram of sensor nodes is then constructed. The cluster head sets up a TDMA schedule and transmits it to the nodes in the cluster. After the TDMA schedule is known by all nodes in the cluster, the setup phase is completed and the steady-state operation (data transmission) can begin. The organization of intra-cluster data transmission is similar to LEACH after clusters have been set up, so we omit it in this section.

1: $\mu \leftarrow RAND(0, 1)$

2: **if** $\mu < T$ **then**

3: *beTentativeHead* ← *TRUE*

4: **end if**

5: **if** *beT entativeHead* = *TRUE* **then**

6: broadcast a COMPETE HEAD MSG(*si*. *ID*,*Ri*, *si*. *RE*)

7: **else**

8: EXIT

9: **end if**

10: on receiving a COMPETE HEAD MSG from node *sj*

11: **if** *d*(*si*, *sj*)<*max*(*Ri*,*Rj*) **then**

12: add *sj* to *si*. *SCH*

13: **end if**

14: **while** the time slot for cluster head competing has not expired **do**

15: **if** *si*.*RE* > *sj* .*RE*, ∀*sj* ∈ *si*.*SCH* **then**

16: broadcast a FINAL HEAD MSG(*si* .*ID*) and then EXIT

17: **end if**

18: on receiving a FINAL HEAD MSG from node *sj*

19: **if** *sj* ∈ *si*.*SCH* **then**

20: broadcast a QUIT ELECTION MSG(*si* .*ID*) and then EXIT

21: **end if**

22: on receiving a QUIT ELECTION MSG from node *sj*

23: **if** *sj* ∈ *si*.*SCH* **then**

24: remove *sj* from *si*. *SCH*

25: **end if**

26: **end while**

Figure 5 - Cluster head competitive algorithm for node si

4.1.4 INTER-CLUSTER MULTIHOP ROUTING

When cluster heads deliver their data to the base station, each cluster head first aggregates the data from its cluster members, and then sends the packet to the base station via a multi-hop path through other intermediate cluster heads.

The routing problem here differs substantially from that of traditional ad-hoc wireless networks because of the many-to-one traffic pattern. On the other hand, both query-driven and event-driven routing protocols for wireless sensor networks are not suitable for the cluster head virtual backbone. In [9], Younis et al. prove that HEED can produce a connected multihop cluster head backbone using a fixed inter-cluster transmission range. Using the fixed transmission power facilitates its implementation on the TinyOS platform, where the multihop routing uses a shortest-path-first algorithm [27]. By using adjustable transmission range and weak location information, we design a greedy geographic and energy-aware multihop routing protocol to extend the network lifetime.

Before selecting the next hop node, each cluster head broadcasts a short beacon message across the network at a fixed power which consists of its node ID, residual energy, and distance to the base station. Distance between each pair of cluster heads can be calculated approximately according to the received signal strength. We introduce a threshold TD MAX in the multihop routing protocol. If a node's distance to the base station is smaller than TD MAX, it transmits its data to the base station directly; otherwise it's better to find a relay node which can forward its data to the base station.

It is worth explaining that the value of TD MAX is always smaller than the actual maximum transmission range of a sensor node as we try to avoid the long-distance direct communication of heavy traffic. A node could still use a transmission range larger than TD MAX if necessary.

To reduce wireless channel interference, it is better to choose an adjacent node as the relay node [29]. Because the transmission power of cluster heads is adjustable, hop

count is improper to be used is a poor method of defining neighboring relations. In this paper, the multihop forwarding algorithm considers nodes on the cluster head backbone in the forward direction (i.e., closer to the base station) only.

The neighboring node set *RCH* of cluster head *si* is defined as

$$s_i.RCH = \{s\,j \mid d(s_i, s_j) \le xR_i, d(s_j, BS) < d(s_i, BS)\} \ldots\ldots (4.4)$$

x is the minimum integer that lets *si .RCH* contain at least one item (if there doesn't exist such an *x*, define *si .RCH* as a null set, and *si* will send its own data together with forwarding data directly to the base station).

Choosing the relay node with more residual energy could balance the energy consumption to extend the network lifetime. On the other hand, decreasing the energy cost per packet also contributes to the network lifetime. Here, a greedy geographic forwarding algorithm is proposed that aims to minimize the energy cost per packet.

Suppose *si* chooses *s j* as its relay node. For simplicity, we assume a free space propagation channel model. Because a localized algorithm is desirable, we assume there is a virtual hop between *s j* and the base station. To deliver an *l*-length packet to the base station, the total energy consumed by *si* and *s j* is

$$\begin{aligned} E_{2-hop}(s_i, s_j) &= E_{Tx}(l, d(si, s\,j)) + E_{Rx}(l) + E_{Tx}(l, d(s_j, BS)) \\ &= l(E_{\text{elec}} + \epsilon_{fs} d^2(s_i, s_j)) + lE_{\text{elec}} + l(E_{\text{elec}} + \epsilon_f s\, d^2(s_j, BS)) \\ &= 3lE_{\text{elec}} + l\epsilon_{fs}(d^2(si, s\,j) + d^2(s\,j, BS)) \ldots\ldots (4.5) \end{aligned}$$

according to Eqs. (4.1) and (4.2). Thus we define

$$E\text{relay}(si, s\,j) = d2(si, s\,j) + d2(s\,j, BS) \ldots\ldots (4.6)$$

as the energy cost of the path $si \Rightarrow s\,j \Rightarrow BS$. We use the distance between nodes rather than precise location information of $s\,j$ to define the energy cost of the relay path. The bigger the *E*relay is, the more energy will be consumed for transmitting packets on the path.

In the localized routing algorithm, *si* first chooses *k* eligible neighbor nodes from *si .RCH*, denoted as the set *S*eligible:

$$si.S_{\text{eligible}} = \{s_j \mid s_j \in si.R_{CH},\ E\text{relay}(s_i, s_j)\ \textit{is the k smallest}\} \quad \ldots\ldots (4.7)$$

The pseudocode for constructing $si.S_{\text{relay}}$ is given in Fig. 7. To reduce inefficiencies of energy consumption, a tradeoff should be made between the two criteria of residual energy and link cost E_{relay}. In our mechanism, *si* chooses as its relay node the neighbor in $si.S_{\text{eligible}}$ that has the biggest residual energy.

1: **while** $\forall sj \in RCH$ is not null **do**
2: compute *Erelay*(*si*, *sj*)
3: **if** |*Srelay*| < *k* **then**
4: put *sj* into *Seligible*
5: **else**
6: find out *sm* that satisfies
Erelay(*si*, *sm*) = *max*{*Erelay*(*si*, *sn*)}, $\forall sn \in Seligible$
7: **if** *Erelay*(*si*, *sj*) < *Erelay*(*si*, *sm*) **then**
8: replace *sm* with *sj* in *Seligible*
9: **end if**
10: **end if**
11: remove *sj* from *RCH*
12: **end while**

Figure 6 - Eligible neighbor nodes choosing algorithm

Besides the tradeoff between the two different goals, we propose another goal to balance the energy consumption. The nodes near the base station send the forwarding data directly to the base station, thus they may deplete their energy quickly if the base station is located far from the network field. In our solution, if cluster head *s j* 's distance to the base station is smaller than TD MAX, and cluster head *si* selects *s j* as its relay node according to the approach described before, and if the residual energy of *s j* is smaller than that of *si*, we let *si* communicate with the base station directly rather than aggravating the load of *s j* . In this way, energy of *s j* can be saved and the network lifetime is extended further.

After each cluster head has chosen a relay node or decided to transmit its data to the the base station directly, a tree rooted at the base station is constructed. A cluster head receives data packets from tree descendants and sends them with the cluster's own packets up to the root.

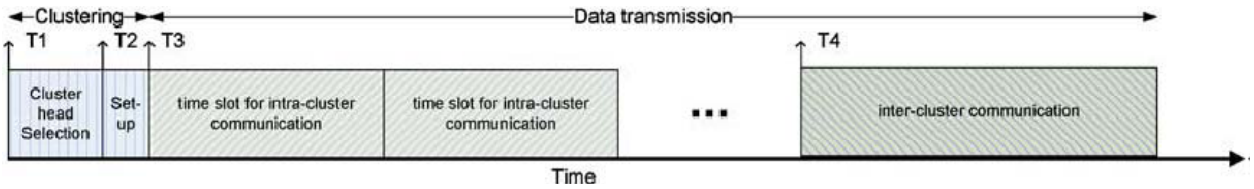


Figure 7 - Time-line showing UCR operation

Figure 17 illustrates the operation of UCR by the time line for one data gathering round. It begins with a clustering phase when cluster heads are selected and the intra-cluster TDMA schedule are set-up, followed by a data transmission phase where data are transferred from the nodes to the cluster head and on to the base station via a multi-hop path.

4.1.5 Balance of Energy Consumption with UCR

Due to smaller sizes of clusters in the hot spot, nodes are selected as cluster heads more frequently than these not in the hot spot. Energy holes may still appear though the intra-cluster load of cluster heads in hot spot could be reduced via unequal clustering. Our solution is that in each round, let heads in the hot spot consume less energy (intra- and inter-cluster cost in total) than these not in the hot spot, rather than consuming the same energy. This can be accomplished through decreasing the cluster size and inter-cluster transmission range of cluster heads in the hot spot simultaneously. As a rough example, suppose *s*1 is a node in the hot spot, and it is selected as a head every 10 rounds; *s*2 not in hot spot, as a head every 15 rounds. Suppose the energy consumption of *s*1 in a round is 0.1J if *s*1 is a head, and for *s*2 0.15J. We ignore the energy consumption in rounds when it is an ordinary node. In every 30 rounds, *s*1 and *s*2 consume the same amount of energy: for *s*1, it is 0.1J*3, and for *s*2 it is 0.15J*2. So the energy hole is avoided. Although the explanation is not formalized, it serves as a reasonable guideline to balancing the load across the network.

4.1.6 Conclusion

The main goal of Un-equal clustering is the need to resolve hot-spot problem which is mainly caused when we have big area of non-uniformly dispersed nodes communicating far from a gateway or Base Station (BS). This protocol achieves this by keeping the competitive range of cluster-heads closer to the BS or sink smaller than that of those far from the sink. This reduces intra-cluster load on the heads closer to the BS and they can

utilize more of their energy in inter-cluster communication. This results truly in uniform energy distribution across the network.

CHAPTER 5

5.1 HEED

Introduction

Many Protocols have been proposed that just depend on a single metric to either enhance average network life-time or reduce the communication cost to reach to the sink. Hybrid Energy Efficient Distributed (HEED) tries to achieve both the reduction in energy dissipation, thereby, increasing the network lifetime and the minimum communication cost incurred on any node to reach to the sink by employing the distance to the sink as the factor affecting the communication cost. This results in efficient communication particularly for Multi-media Wireless Sensor Networks (MMWSN) where communication cost reduction becomes as much as the minimum energy consumption, as to achieve the over-all balanced routing across heterogeneous nodes in these kind of networks.

5.1.1 NETWORK MODEL

Consider a set of sensors dispersed in a field. We assume the following properties about the sensor network:

1) The sensor nodes are quasi-stationary. This is typical for sensor network applications.
2) Links are symmetric, i.e., two nodes $v1$ and $v2$ can communicate using the same transmission power level.
3) The network serves multiple mobile/stationary observers, which implies that energy consumption is not uniform for all nodes.
4) Nodes are location-unaware, i.e. not equipped with GPS-capable antennae. This justifies why some techniques, such as [10], [23] are inapplicable.

5) All nodes have similar capabilities (processing/communication), and equal significance. This motivates the need for extending the lifetime of every sensor.
6) Nodes are left unattended after deployment. Therefore, battery re-charge is not possible. Efficient, energy-aware sensor network protocols are thus required for energy conservation.
7) Each node has a fixed number of transmission power levels. An example of such sensor nodes are Berkeley Motes.
8) A simplified energy for the communication energy dissipation, has been deployed. Both the free space ($d2$ power loss) and the multi-path fading ($d4$ power loss) channel models are used, depending on the distance between the transmitter and receiver. The energy spent for transmission of an l-bit packet over distance d is:

$$ET\,x(l,\,d) = lE_{\text{elec}} + l\epsilon d^{\alpha} = \begin{cases} lE\text{elec} + l\epsilon\, f\, sd2\,, d < do \\ lE\text{elec} + l\epsilon mpd4, \quad d \geq do \end{cases} \quad \ldots\ldots (5.1)$$

The electronics energy, E_{elec}, depends on factors such as the digital coding, and modulation, whereas the amplifier energy, $\epsilon_{fs}d^2$ or $\epsilon_{mp}d^4$, depends on the transmission distance and the acceptable bit-error rate. To receive this message, the radio expends energy:

$$E_{Rx}\,(l) = lE_{\text{elec}} \ldots\ldots (5.2)$$

Let the clustering process duration, *TCP*, be the time interval taken by the clustering protocol to cluster the network. Let the *network operation interval*, *TNO*, be the time between the end of a *TCP* interval and the start of the subsequent *TCP* interval. We

must ensure that *TNO* ≫ *TCP* to reduce overhead. Although we assume that nodes are not mobile, clustering can still be performed if nodes that announce their willingness to be cluster heads are quasi-stationary during the *TCP* interval in which they are selected, and the ensuing *TNO* interval. Nodes that travel rapidly in the network may degrade the cluster quality, because they alter the node distribution in their cluster.

It is important to note that in our model, *no* assumptions are made about (1) homogeneity of node dispersion in the field, (2) network density or diameter, (3) distribution of energy consumption among sensor nodes, (4) proximity of querying observers, or (5) node synchronization.

5.1.2 The Clustering Problem

Assume that *n* nodes are dispersed in a field and the above assumptions hold. Our goal is to identify a set of cluster heads which cover the entire field. Each node *vi*, where $1 \leq I \leq n$, must be mapped to exactly one cluster c_j , where $1 \leq j \leq nc$, and n_c is the number of clusters ($n_c \leq n$). A node must be able to directly communicate with its cluster head (via a single hop). Cluster heads use a routing protocol to compute inter-cluster paths for multi-hop communication to the observer(s).

The following requirements must be met:

1) Clustering is completely distributed. Each node independently makes its decisions based only on local information.

2) Clustering terminates within a fixed number of iterations (regardless of network diameter).

3) At the end of each *TCP*, each node is either a cluster head, or not a cluster head (which we refer to as a regular node) that belongs to exactly one cluster.

4) Clustering should be efficient in terms of processing complexity and message exchange.

5) Cluster heads are well-distributed over the sensor field, and have relatively high average residual energy compared to regular nodes.

5.1.3 CLUSTER FORMATION

As clustering is triggered every *TCP* +*TNO* seconds to select new cluster heads. At each node, the clustering process requires a number of iterations, which we refer to as N_{iter}. Every step takes time *tc*, which should be long enough to receive messages from any neighbor within the cluster range. We set an initial percentage of cluster heads among all *n* nodes, *Cprob* (say 5%), assuming that

an optimal percentage cannot be computed a priori. *Cprob* is only used to limit the initial cluster head announcements, and has no direct impact on the final clusters. Before a node starts executing HEED, it sets its probability of becoming a cluster head, *CHprob*, as follows:

$$CHprob = CH_{prob} \times \frac{Eresidual}{Emax} \quad \ldots\ldots \quad (5.3)$$

where *Eresidual* is the estimated current residual energy in the node, and *Emax* is a reference maximum energy (corresponding to a fully charged battery), which is typically identical for all nodes. The *CHprob* value of a node, however, is not allowed to fall below a certain threshold *pmin* (e.g., 10−4), that is selected to be inversely proportional to *Emax*. This restriction is essential for terminating the algorithm in *Niter* = *O*(1) iterations,

as we will show later. Observe that our clustering approach is capable of handling heterogeneous node batteries. In this case, every node will have its own *Emax* value.

In order for the clustering to be done in TCP, the topology grid is divided into smaller regions so as to obtain uniform distribution of nodes in the network. The Eresidual of each node in this smaller region is determined and compared and the node having the highest energy wins the election. This node broadcasts CH_MSG. Note that in case two or more nodes have the same energy, the node with the lowest id becomes the cluster-head in a specific region. The nodes in the cluster-range not elected as Cluster-head receive CH_MSG from all the heads in their range and decide to join a specific head depending on the lowest communication cost. The communication is the energy cost per packet.

Suppose *si* chooses *s j* as its relay node. For simplicity, we assume a free space propagation channel model. Because a localized algorithm is desirable, we assume there is a single hop between *si* and the Cluster Head CH_i. To deliver an *l*-length packet to the cluster-head, the total energy consumed by s_i is

$$E_{si\text{-}CHi}(s_i, s_j) = E_{Tx}(l, d(si, s\,j)) + E_{Rx}(l)$$

$$= l(E_{\text{elec}} + \epsilon_{fs}\, d^2(s_i, s_j)) + lE_{\text{elec}}$$

$$= 2lE_{\text{elec}} + l\epsilon_{fs}(d^2(si, s\,j) \quad \ldots\ldots (5.4)$$

according to Eqs. (5.1) and (5.2). Thus we define

$$E_{si\text{-}CHi}(si, s\,j) = d^2(si, s\,j) \quad \ldots\ldots (5.5)$$

Thus, distance to the head becomes the over-riding factor to decide the communication cost. Hence, each node uses its distance to a given cluster-head and joins to the closest head at the end of every T_{CP} period.

In order to elect the next head after $T_{CP}+T_{NO}$, the head elected after TCP gets the metric of each of its neighbors using a METRIC_MSG. This message is sent periodically. The next head is selected as soon as the energy of the present head reaches certain threshold called E_{thres}.

Neighbor discovery is not necessary every time clustering is triggered. This is because in a stationary network, where nodes do not die unexpectedly, the neighbor set of every node does not change very frequently. In addition, HEED distribution of energy consumption extends the lifetime of all the nodes in the network, which adds to the stability of the neighbor set. Nodes also automatically update their neighbor sets in multi-hop networks by periodically sending and receiving heartbeat messages.

A node considers itself “covered” if it has heard CH_MSG. If a node completes HEED execution without selecting a cluster head that is Cluster-Head, it considers itself uncovered, and announces itself to be a cluster head by broadcasting CH_MSG. Note that a node can elect to become a cluster head at consecutive clustering intervals if it has high residual energy and low cost.

1: Divide the Network Topography into region of equal and appropriate size

2: If ($s_i.E_{residual} > s_k.E_{residual}$) in ($X_i$,$Y_i$) region, s_i broadcasts CH_MSG

3: If (sk is not CH) then

Recv CH_MSG

Store d(s_k, s_i)

4: If (now_time == $2T_{CP}+T_{NO}$)

Set s_i as head such that d(s_k,s_i) = Min(d(s_k,s_i))

5: S_k sends JOIN_MSG to S_i

6: S_k send METRIC_MSG (s_k.metric) to S_i after ever T_m

7: S_i Recv METRIC_MSG and makes a set S_m.

8: After every $T_{CP}+T_{NO}$, S_i sends NEXT_HEAD_MSG to S_k with S_k.metric = MIN (S_k.metric $\in$ S_m)

9: S_k receives NEXT_HEAD_MSG and broadcasts CH_MSG and steps 3-8 repeat again

Figure 8 - HEED Clustering Algorithm

5.1.4 INTER-CLUSTER COMMUNICATION

After the network is clustered, inter-cluster organization depends on the network application. For example, cluster heads can communicate with each other to aggregate their information via multiple Hops. For multi-hop communication among cluster heads, the selected transmission range among cluster heads may vary to ensure a certain degree of connectivity and to control interference. For example, in [26], the authors assume that the nodes are uniformly distributed in the network field and that each cell of size $c \times c$ in the network contains at least one node. In this case, the network is guaranteed to be connected if the inter-cluster transmission range

$$Rt = (1+\sqrt{5})c \quad \ldots\ldots (5.6)$$

is satisfied. A cell in this context is defined as an area in the 2-dimensional space in which every node can communicate with every other node residing in every neighboring

cell. In a clustered network, a cell can be defined as an area where every node can reach every other node residing in the same cell. The cell side length is therefore $\leq R_c/\sqrt{2}$ where *Rc* is the cluster range. Thus, we can conduct a similar analysis to [26], [27] to select *Rt*. In [3], the authors suggest using the minimum possible power level to reach a destination, in order to reduce interference.

In [4], the authors propose a technique to select the minimum power level to use across the entire network in order to keep it connected, assuming uniform node dispersion. Any of these techniques can be adopted in to guarantee a connected inter-cluster overlay graph.

For inter-cluster communication, the definition of connectivity depends on its multi-hop organization and the relationship between the inter-cluster transmission range, *Rt*, and the intra-cluster transmission range, *Rc*. The following lemmas and theorem define the required density model and provide the necessary conditions for asymptotically almost surely (a.a.s.) multi-hop network connectivity.

Lemma 1: Assume that *n* nodes are uniformly and independently dispersed at random in an area $R = [0, L]^2$. Also assume that the area is divided into *N* square cells of size $Rc/\sqrt{2} \times Rc/\sqrt{2}$. If $R^2_c\, n = aL^2 \ln L$, for some $a > 0$, then

$$\lim_{n,N\to\infty} E[\eta(n,N)] = 1 \quad \ldots\ldots (5.7)$$

where $\eta(n,N)$ random variable that denotes the minimum number of nodes in a cell (i.e., each cell contains at least one node a.a.s., or the expected number of empty cells is zero a.a.s.).

Lemma 2: There exists at least one cluster head in any $(2 + 1/\sqrt{2})Rc \times (2 + 1/\sqrt{2})Rc$ area a.a.s.

Proof. This lemma is proved by contradiction. Assume that Lemma 1 holds, and that there there does not exist any cluster heads in an $(2 + 1/\sqrt{2})Rc \times (2 + 1/\sqrt{2})Rc$ area *A*. This implies that every node *v* within this area *A* is connected to a cluster head that lies outside *A*. Even if cluster heads outside *A* are on the borders of *A*, then there is at least an area $B = Rc/\sqrt{2} \times Rc/\sqrt{2}$ inside *A* which cannot be covered by cluster heads outside *A* (as depicted in Fig. 3(a)). But area *B* contains at least one node a.a.s. according to Lemma 2 and this node is connected to a cluster head within *A*. This contradicts the initial assumption, and therefore there exists at least one cluster head within *A* a.a.s.

Lemma 3: For any two cluster heads *v*1 and *v*2 in two neighboring areas *A* and *B* of size $(2 + 1/\sqrt{2})Rc \times (2 + 1/\sqrt{2})Rc$, *v*1 and *v*2 can communicate if $Rt \geq 6Rc$.

Proof. Fig. 3(b) shows an organization where a $(2 + 1/\sqrt{2})Rc \times (2 + 1/\sqrt{2})Rc$ area *A* contains one cluster head *v*1 in the bottom left corner. A cluster head *v*2 is the farthest from *v*1 when it resides in the top right corner of the closest $(2 + 1/\sqrt{2})Rc \times (2 + 1/\sqrt{2})Rc$ area *B*. Using Euclidean geometry, the distance between *v*1 and *v*2 $\approx 6Rc$, which is the minimum transmission range *Rt* for *v*1 to reach *v*2.

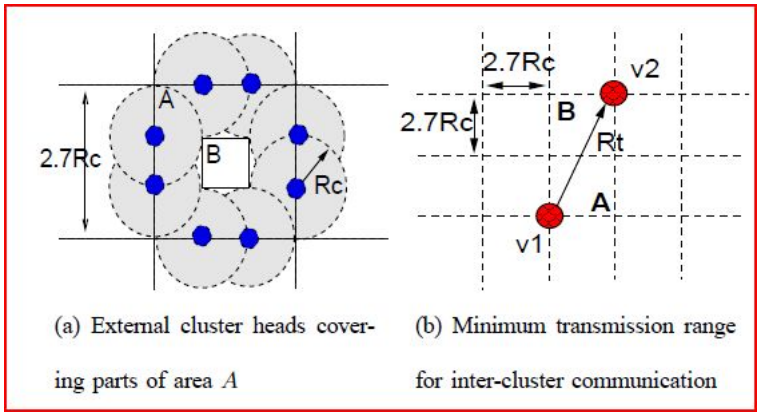


Figure 9 - CONDITIONS ON TRANSMISSION RANGE FOR NEWORK CONNECTIVITY

Theorem 1: HEED produces a connected multi-hop cluster head graph (overlay) a.a.s.

Proof. Assume that the conditions in the previous 3 lemmas hold. We prove this theorem by contradiction.

Assume that HEED produces two connected components (graphs) of cluster heads $G1 = (V1, E1)$ and $G2 = (V2, E2)$, such that any $v1 \in V1$ cannot communicate with any $v2 \in V2$. Without loss of generality, assume that $V2$ lies on the right of $V1$, and that a cluster head $v1 \in V1$ lies on the rightmost border of $V1$. $v1$ is able to communicate with a cluster head $v2$ on its right side, since the condition in Lemma 3 holds. $v2$ must reside inside $V2$, which contradicts with the assumption that a cluster head in one component cannot communicate with one in the other component. Thus, $V1$ and $V2$ are connected a.a.s. *2*

It is considered that clustering and data aggregation in a dense network conserve energy. To evaluate this conjecture, we conduct a very simple worst-case analysis on an operational scenario. The goal of this analysis is to quantify the required node density to achieve such energy conservation for that scenario.

Assume that transmission proceeds from all nodes in the top left cell to an observer in the bottom right cell. Define the energy gain, *Eg*, as the difference between the energy consumed for transmitting 1 bit of data by all the nodes in the top left cell without clustering *Eo*, and the energy consumed for transmitting 1 bit of data by all the nodes in the top left cell using clustering and data aggregation *Ec*. Therefore,

$$Eg = Eo - Ec \quad \ldots\ldots (5.8)$$

Now assume that: (1) nodes are dispersed uniformly at random in a field, (2) one unit of energy is consumed for transmitting 1 bit of data per one unit of distance, and (3) every cell (as defined above) has one cluster head. We now show that $Eg > 0$, if $n > 2\sqrt{2}\,(L/Rc)^2$

Since $L >> Rc$, the optimal path length from the source nodes to the observer $=\sqrt{2}L$. To compute the suboptimal path length (in the clustered network), consider each 2×2 cell. The clustered network path at most deviates by a factor of $\sqrt{2}$ from the optimal $2 \times Rc$ path. Therefore, the suboptimal path length $= 2 \times\sqrt{2} \times Rc \times\sqrt{2}L/(2 \times Rc) = 2L$. The average number of nodes per cell (assuming uniform distribution) $= n \times Rc^2/(2 \times L^2)$.

Eo = energy consumed by all the nodes in the cell to send 1 bit along the $\sqrt{2}L$ path to the observer $= n \times Rc^2/(2 \times L^2) \times\sqrt{2}L = n \times Rc^2/\sqrt{2}L$.

Ec = energy consumed by all the nodes in the cell to send 1 bit to their cluster head at range *Rc*

+ the energy consumed by the cluster head to transmit on the suboptimal path to the destination $= [n \times Rc^2/(2 \times L^2) - 1] \times Rc + 2L$. Therefore, $Ec \approx n \times Rc^3/2L^2 + 2L$.

$Eg = Eo - Ec = n \times Rc^2/\sqrt{2}L - n \times Rc^3/2L^2 - 2L > 0$

$$\Rightarrow\ n[(Rc^2/\sqrt{2}L) - (Rc^3/2L^2)] > 2L$$

$\Rightarrow\ n > 4L^3Rc^2\ (\sqrt{2}\mathrm{L} - Rc)$

Since $L >> Rc$, therefore, $\sqrt{2}\mathrm{L} >> Rc$, and $n > 2\sqrt{2}\ (L/Rc)^2$.

5.1.5 INTER-CLUSTER ROUTING

In the description of HEED operation, we assumed single-hop communication among cluster heads and their registered cluster members. This is desirable in source-driven networks, where reports are periodically transmitted by the sensor nodes. In this case, a TDM frame may be constructed at each cluster head to eliminate interference within a cluster. Clearly, constructing TDM frames requires node synchronization, and in lightly-loaded networks, using TDM frames may waste resources. A better approach in this case is to allow channel contention. Multi-hop routing to the cluster head can increase network capacity in this case.

Cluster head overlay (i.e., inter-cluster) routes are used to communicate among clusters, or between clusters and the observer(s). In this case, an ad-hoc routing protocol, such as Directed Diffusion [5] or Dynamic Source Routing (DSR) [31], can be employed for data forwarding among cluster heads. TinyOS beaconing is the approach currently specified for sensors running TinyOS. This constructs a breadth-first spanning tree rooted at the base station. In a clustered network, the beaconing approach can be applied to only the cluster head overlay, instead of the entire network.

If two regular nodes from different clusters attempt to communicate, communication through their cluster heads is sub-optimal if the two regular nodes can directly communicate via a shorter path. This, however, is not the typical communication pattern for sensor network applications, where data is transmitted to an observer which

is not close to the target source of data, and data may be aggregated by cluster heads. In addition, since the cluster range is typically limited (compared to the network size), the network can be approximately viewed as a grid-like area, where optimal routes along the grid are computed using routing tables or through reactive routing techniques.

Conclusion

CHAPTER 6

6.1 CAODV

Introduction

Ad-hoc On Demand Distance Vector (AODV) has the tremendous applications in multi-purpose wireless networks, particularly in MANETS. This is mainly due to its reactive nature and feature of reliable data delivery. But, in Wireless Multi-media sensor networks, the over-head of AODV can become un-tolerable. Our goal is to reduce the network over-head of AODV by not changing it entirely rather by applying a mechanism whereby the effect of intense overhead caused by multiple flooding of control packets in AODV can be reduced. In order to accomplish that, we have employed clustering approach. First, we will discuss the basic AODV algorithm, its inherent features including advantages and disadvantages and then our clustering approach. At the end of this chapter, we will show the robustness of our mechanism to achieve low over-head and greater lifetime of Wireless Sensor Network (WSN).

6.1.1 AD-HOC ON DEMAND DISTANCE VECTOR (AODV)

The AODV routing protocol is a pure on-demand routing algorithm. Consequently, AODV discovers and maintains routes between two nodes, when these two nodes need to communicate with each other. A sequence number is used by AODV to ensure the maintenance of the fresh routes and the utilization of the most recent routing

information. AODV has two main objectives: broadcasting discovery packets and distributing information about changes in location. AODV is different from other routing protocols; as it uses the sequence number of the source and the destination in addition to flooding the route request (RREQ) packet across the network in order to find the route to the intended destination. When a source node needs to communicate with another node, for which it has no route between them in its routing table, a path discovery process is commenced by broadcasting an RREQ packet from the source node to its neighbors. The RREQ packet includes the source node's address, the destination node's address, the current sequence number of the source, the broadcast ID, the most obtained sequence number for the destination, and a hop counter. Both of the source address and the broadcast ID are used to identify each RREQ. The broadcast ID will be increased one increment, if the source node sends a new RREQ. If any node has received this RREQ before this time, it would drop the redundant one, and it would not rebroadcast it again. Otherwise, if the receiver node knew the route to the destination, it would send back a route reply (RREP) packet, which contains information about how to reach the destination node. This information includes the most recent sequence of numbers from the destination and the number of hops to reach the destination. If the receiver node does not know the route to the destination, it would rebroadcast RREQ to its neighbors and would increment the hop counter. This process of sending an RREQ is shown in Fig. 11.

In AODV, the validity of a route at the intermediate node is confirmed, if the corresponding destination sequence number is greater than or equal to the destination

sequence number, which is contained in the broadcasting RREQ packet. Any node, which sends an RREP, places the current sequence number of the destination and its distance in hops to the destination into the RREP. Subsequently, an RREP is sent back to the source along the path followed by RREQ. Once the source node receives the RREP packet, it transmits the data packets to the destination. If the route discovery timer expires and the source node does not receive any RREP, it rebroadcasts the RREQ. This discovery at- tempt is repeated for a predetermined maximum number of times. If no route is discovered after the maximum number of attempts, the session will be aborted.

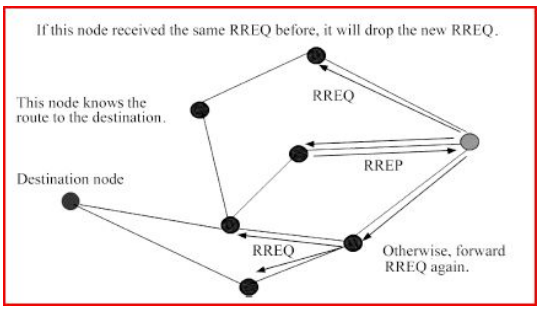


Figure 10 - AODV sending RREQ packet process

If a link failure occurs, while the route is active and any neighbour of the upstream node is using that link, the node creates and propagates a route error (RERR) packet to the source node informing it of the unreachable destinations. The RERR packet contains all Internet protocol (IP) addresses of unreachable destination nodes (due to the link break) and their sequence numbers, which are incremented by one. Subsequently, the node broadcasts the packet and invalidates those routes in its route table. If the source node still needs the route to the destination node, it can reinitiate the route discovery process again after receiving the RERR.

6.1.1.1 Features of AODV

The main advantage of this protocol is that it uses an on- demand approach for establishing the routes between two nodes. On the other hand, the serious drawbacks of this protocol are the multiple RREP packets generated in response to a single RREQ packet. This can lead to extra control overhead due to the flooding of the RREQ packet across the network. This overhead is very costly and may result in serious redundancy, contention, and collision. During the route discovery process, an RREQ packet is sent out to all neighbors. Each neighbour node in turn forwards the RREQ packet to its neighbour nodes without taking into account whether or not the neighbour node is going to be out of the coverage range. In time, this forwarding

could result in a failure to send the data along a discovered route, due to a possible link break occurring by one neighbour moving outside the range of the previous node in the path. The other drawback of the AODV is that it keeps using the same path between the source node and the destination node until any of the intermediate nodes die.

6.1.2 CLUSTERING IN AODV

The process of clustering starts with the election of the volunteer nodes (VNs). The VNs can be elected randomly or by picking nodes of higher energy. Since, volunteer nodes have to act as volunteer just at the beginning, the overhead in deciding a particular node as volunteer node can be far more than the effect of selecting nodes of lower energy as volunteer, on the total life-time of the network. Thus, random selection of volunteer nodes is preferred. One thing to note is that the volunteer nodes are to be well-distributed in the whole area so as to cover all the nodes in the network. The function of a VN is to divide the total network area into multiple zones. It then

broadcasts ZONE_INF message to the other nodes which consist of the coordinates deciding a particular zone. For example, any node lie under (50,50) belong to zone 1 and any other lying between (50,50) and (100,50) is in zone 2 and so on.

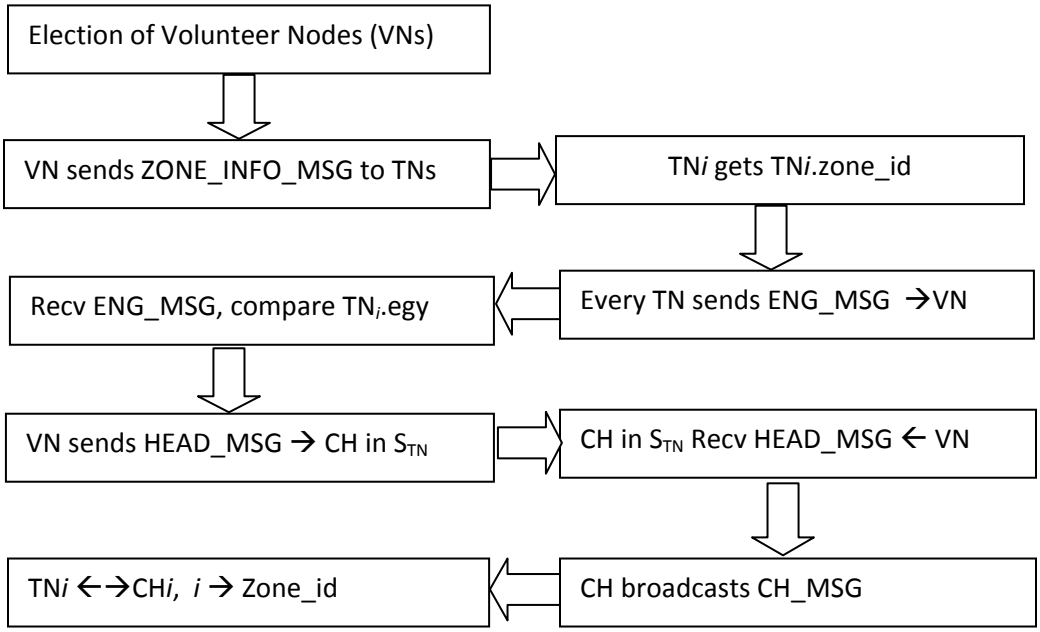
The Typical Nodes (TN) receive the ZONE_INFO_MSG and decide about their zones depending upon their location. The location can be found out using the ways discussed in chapter 3. During the same time interval, TNs find out their residual energy designated as Eresidual and broadcast ENG_MSG to VN.

The VN receives the ENG_MSG and after an interval of T_{ch} , designates a node with the highest energy to be a cluster-head in a particular zone. It sends a unicast HEAD_MSG (vn.id, ch.id) to the designated Cluter-head. After receiving the HEAD_MSG, the selected cluster-head broadcasts CH_MSG (ch.id , ch.zone). All the TNs receiving this message associate with their respective cluster-heads based on their zone ids by sending JOIN_MSG and setting respective CHs as the next hop towards the sink. The main thing to be noted is that whenever TNs have to communicate with the sink, they do not generate RREQ packets, rather send their data directly to the sink. Only, the heads run the AODV algorithm, and this reduces communication cost and enhances the over-all network life-time.

The over-all process of cluster-formation is shown in fig 21. In order to make the clustering more robust, the rotation among cluster-heads in a specific zone is done to ensure uniform energy distribution in the network. This is done by choosing one of the nodes except the present head as the volunteer node (VN) in a zone. Thus, rotation process is local to a particular zone. Afterwards, the over-all process is done again as in

the beginning. Thus, all the steps are done in the same manner except the formation of zones and sending of ZONE_INFO_MSG because the zones are already there.

Figure 12 – FLOW DIAGRAM FOR THE FORMATION OF CLUSTERS AND CLUSTER-HEAD



6.1.3 ROBUSTNESS ACHIEVED WITH CLUSTERING IN AODV

As the major drawback of AODV are the multiple RREP packets generated in response to a single RREQ packet. This can lead to extra control overhead due to the flooding of the RREQ packet across the network. This overhead is very costly and may result in serious redundancy, contention, and collision.

With clustering, only the cluster-head generate RREQ packets and hence, RREP that reduces the control over-head to a great extent. Although, the clustering introduces its own over-head but this over-head is not effective compared to the reduction in over-head achieved by allowing only cluster-heads to run AODV algorithm. This immense

reduction in over-head, marks great perspective for CAODV to be used as a protocol of choice in WSNs.

6.1.4 CONCLUSION

AODV run on all the nodes in a WSN not only consumes memory and power resources but also generates too much over-head. Our implied clustering technique counters it quite efficiently by allowing only cluster-heads to do routing using AODV. This not only reduces control over-head but also results in low-latency communication and over-all greater network life-time.

CHAPTER 7

7.1 COMPARISON PARAMETERS

Introduction

Understanding the fundamental performance limits of wireless sensor networks is critical towards their appropriate deployment strategies. Both the data transmission rate, the lifetime of sensor networks and end-to-end delay can be constrained, due to interference among the transmissions and the limited energy source of the sensor. In addition to presenting the general results with respect to the maximum sustainable throughput of wireless sensor networks, this chapter focuses on the discussion of the energy-constrained fundamental limits with respect to the network throughput, latency and Network lifetime.

7.1.1 THROUGHPUT:

The throughput E is defined as $E = N\tau / \tau$, where $N\tau$ is the number of packets received by the sink within the time period τ [20]. So, the throughput is the successful packets transmitted per second. Some authors take the throughput in Kb/s.

The performance limit of the network throughput is defined as the *maximum stable throughput* (MST) of the network [12]. The maximum stable throughput is the maximum amount of traffic per unit time (usually measured in bits/sec) that can be injected into the network from all the sources while the size of the queue at any network node is

bounded. Usually, it is assumed that all nodes generate equal amount of network traffic. In this case, the maximum stable throughput *per node* can be similarly defined.

7.1.1.1 Relationship between Throughput and Capacity

In most of the literature on performance limits with respect to network throughput, the term *capacity* is used to refer to the maximum network throughput achievable.

The interference-constrained capacity is caused by two factors. First, when more than one nearby network nodes transmit at the same time, the signals at the receiving node can be corrupted. Therefore, some nodes in the network may not be able to transmit at the same time. Consequently, the network throughput is reduced. The aforementioned phenomenon is referred to as the *spatial concurrency constraint* by Gupta and Kumar [19]. Second, in multi-hop wireless networks such as wireless ad hoc and sensor networks, the amount of network traffic produced by each transmitting node is proportional to the number of hops taken from the source node to the destination node. The reason is that every intermediate relay node must retransmit the same data in order to forward the data to the next relay node. This phenomenon is referred to as the *multi-hop traffic constraint*.

Suppose that the mean distance between the source and the destination is L and the transmission range of the transmission is r. The number of hops traversed by the packet is at least L/r . Assume that the per-node throughput is λ, then each node will generate no less than $L\lambda/r$ bits/sec of network traffic, which will be served by other nodes in the network. The total amount of traffic generated by all n nodes in the network is thus at

least $Ln\lambda/r$ bits/sec. To keep the queue lengths of the network nodes bounded, the total amount of traffic cannot exceed nW, where W is the maximum throughput that can be achieved by each network node. Consequently, the following inequality must be satisfied:

$$\lambda \leq Wr/L \ldots. (7.1)$$

Eq. (8.1) shows how the multi-hop traffic constraint affects the per-node throughput λ. More hops between the source node and the destination node means a smaller value of r/L which in turn leads to smaller per-node throughput λ. Eq. (8.2) may suggest that increasing r can improve the per-node throughput λ. However, due to the spatial concurrency constraint, increasing r will prevent more nodes from transmitting at the same time. The loss of λ from increased r is quadratic due to the fact that the spatial concurrency constraint affects all the nodes in the neighboring area of the transmitting node. Consequently, it is more desirable to reduce the transmission range r as much as possible. However, reducing r may cause the wireless ad hoc network to lose connectivity. Gupta and Kumar [18] prove that r needs to be at least $\Theta(\sqrt{logn/\pi n}$)in order to keep the network connected. Therefore, the per-node throughput ¸ diminishes with the increasing number of nodes n.

7.1.1.2 Relationship Throughput and Routing over-head

The routing over-head is the number of routing (control) packets transmitted in comparison to the total successful data transmissions. Although, generally the capacity constraints affect the over-all throughput but the under-lying routing protocol also has its effects on the over throughput achieved in WSNs if they are simulated in the same

scenario because of over-head packets and synchronization time they take. Thus, the throughput the protocols discussed in chapter 4-7 have been compared and simulated. The process of calculation of throughput in NS-2 has been explained in Annex.

7.1.2 LATENCY

Latency is the time difference between the time a packet is sent by a source and that at which it is received at the sink. Thus, it refers to end to end delay. It includes transmission, processing and propagation delays. Moreover, it is also affected by the communication model. Since, the communication model in case of Wireless Sensor Networks is not uniform, so the latency degradation is a big issue, particularly for bursty Multi-media data.

We have calculated and compared the latency of the protocols discussed in Chapter 4-7 in order to determine the effectiveness of each of these protocols in Multi-media communication Wireless Sensor Networks. The method of latency calculation in NS-2 is given in Annex.

7.1.3 NETWORK ENERGY AND NETWORK LIFE-TIME:

The time before the death of the first node determines the life-time of the network

7.1.4 CONCLUSION

Simulation and results

Simulation Parameters:

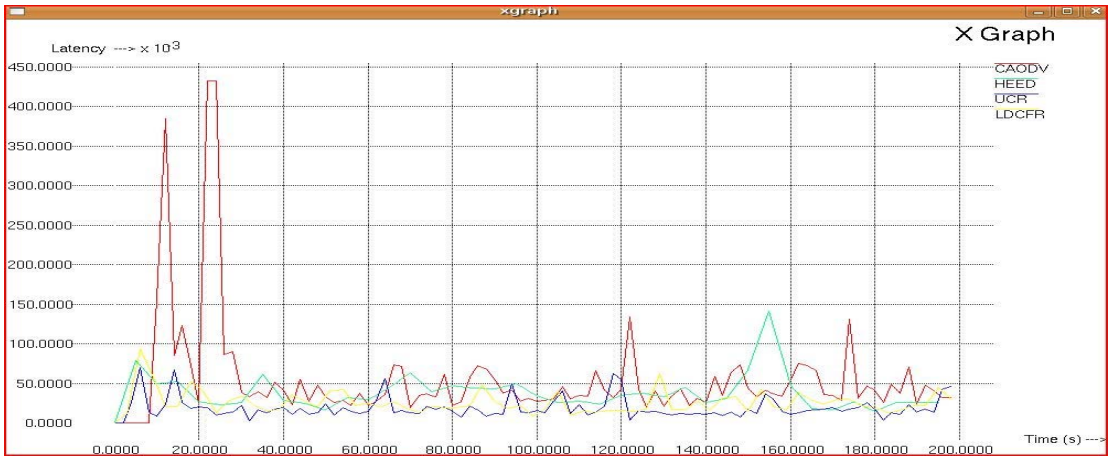


Figure 11 - Latency of different protocols with 100 Nodes

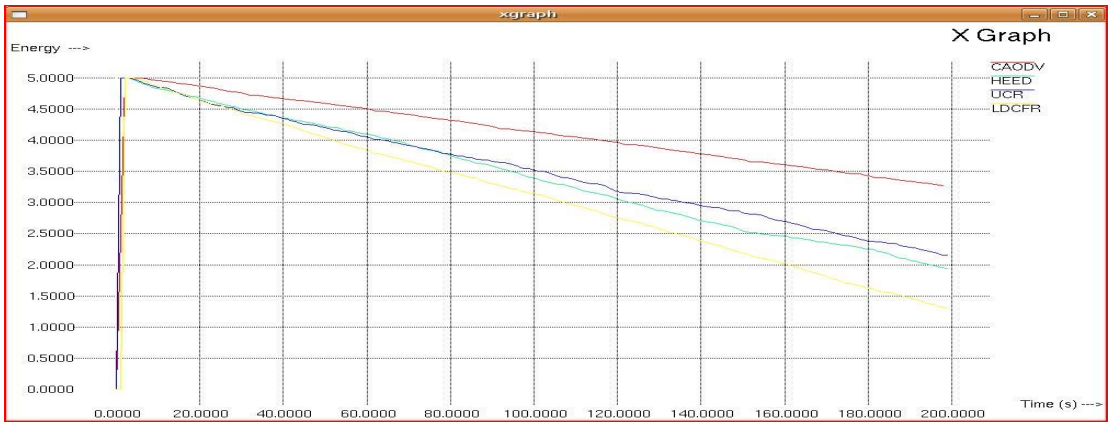


Figure 12 - Average network energy of different protocols with 100 nodes

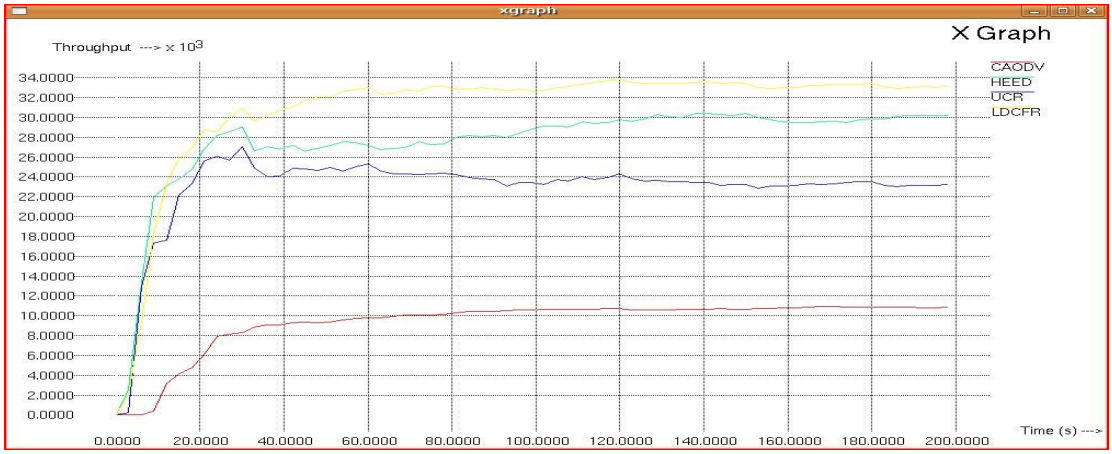


Figure 13 - Throughput of different protocols with 100 nodes

Discussion

Results

Conclusion

Throughput

Latency

Average Network Energy

APPENDIX I

Registration of a Routing Protocol in NS-2

Note: The registration of one of the protocols is being discussed. Only the name of the protocol has to be replaced to register another protocol.

Create directory named as **ldcfr** as

```
mkdir ldcfr
```

In the directory we create three files : ldcfr.cc, ldcfr.h, ldcfr_packet.h.

Now, modify following files. Therefore it is better you backup these files before you start adding new protocol, so that you can easily go back.

- $NS_ROOT/Makefile
- $NS_ROOT/queue/priqueue.cc
- $NS_ROOT/common/packet.h
- $NS_ROOT/trace/cmu-trace.h
- $NS_ROOT/trace/cmu-trace.cc
- $NS_ROOT/tcl/lib/ns-packet.tcl
- $NS_ROOT/tcl/lib/ns-lib.tcl
- $NS_ROOT/tcl/lib/ns-agent.tcl
- $NS_ROOT/tcl/lib/ns-mobilenode.tcl

Let's start with ~/ns-allinone-2.34/ns-2.34/**Makefile** just add following lien at 269.

```
ldcfr/ldcfr.o \
```

Add following lines to ~/ns-allinone-2.34/ns-2.34/queue/**priqueue.cc** from line 93.

```
// LDCFR patch
case PT_LDCFR:
```

To define new routing protocol packet type, modify ~/ns-allinone-2.34/ns-2.34/common/**packet.h** file. Change PT_NTYPE to 63, and for our protocol PT_LDCFR = 62. If one has already installed another routing protocol. Just make sure PT_NTYPE is last, and protocol number is ordered sequentially. From line 85 changes would be :

```
// LDCFR packet
static const packet_t PT_LDCFR = 62;

// insert new packet types here
static packet_t       PT_NTYPE = 63; // This MUST be the LAST one
```

Make following code change at line 254 of ~/ns-allinone-2.34/ns-2.34/common/packet.h. The code is used to indicate that the packet is routing protocol packet and has high priority.

```
type == PT_AODV ||
type == PT_LDCFR)
```

And at line 390 of the same file

```
// LDCFR patch
name_[PT_LDCFR] = "LDCFR";
```

Make NS2 trace our simulation and write it to *something*.tr, in order to do that modify cmu-trace.h and cmu-trace.cc.

To add trace function we add following line to ~/ns-allinone-2.34/ns-2.34/trace/**cmu-trace.h** at line 163:

```
void    format_ldcfr(Packet *p, int offset);
```

~/ns-allinone-2.34/ns-2.34/trace/**cmu-trace.cc** must be added following code at line 1071

```
// LDCFR patch
void
CMUTrace::format_ldcfr(Packet *p, int offset)
{
        struct hdr_ldcfr *wh = HDR_LDCFR(p);
        struct hdr_ldcfr_beacon *wb = HDR_LDCFR_BEACON(p);
        struct hdr_ldcfr_error  *we = HDR_LDCFR_ERROR(p);
```

```
        switch(wh->pkt_type) {
                case LDCFR_BEACON:

                        if (pt_->tagged()) {
                                sprintf(pt_->buffer() + offset,
                                                "-ldcfr:t %x -ldcfr:h %d
-ldcfr:b %d -ldcfr:s %d "
                                                "-ldcfr:px %d -ldcfr:py
%d -ldcfr:ts %f "
                                                "-ldcfr:c BEACON ",
                                                wb->pkt_type,
                                                wb->beacon_hops,
                                                wb->beacon_id,
                                                wb->beacon_src,
                                                wb->beacon_posx,
                                                wb->beacon_posy,
                                                wb->timestamp);
                        } else if (newtrace_) {

                                sprintf(pt_->buffer() + offset,
                                                "-P ldcfr -Pt 0x%x -Ph
%d -Pb %d -Ps %d -Ppx %d -Ppy %d -Pts %f -Pc BEACON ",
                                                wb->pkt_type,
                                                wb->beacon_hops,
                                                wb->beacon_id,
                                                wb->beacon_src,
                                                wb->beacon_posx,
                                                wb->beacon_posy,
                                                wb->timestamp);

                        } else {

                                sprintf(pt_->buffer() + offset,
                                                "[0x%x %d %d [%d %d] [%d
%f]] (BEACON)",
                                                wb->pkt_type,
                                                wb->beacon_hops,
                                                wb->beacon_id,
                                                wb->beacon_src,
                                                wb->beacon_posx,
                                                wb->beacon_posy,
                                                wb->timestamp);
                        }
                        break;

                case LDCFR_ERROR:
                        // TODO: need to add code
                        break;

                default:
#ifdef WIN32
                        fprintf(stderr,
                                        "CMUTrace::format_ldcfr:
invalid LDCFR packet type\n");
#else
                        fprintf(stderr,
```

```
                                        "%s: invalid LDCFR packet
type\n", __FUNCTION__);
#endif
                        abort();
        }
}
```

Modify tcl files to create routing agent. First we define protocol name to use in tcl file. It would done by modifying ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-packet.tcl** at line 172

```
  # LDCFR patch
  LDCFR
```

Set routing agent by modifying ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-lib.tcl** at line 633

```
  LDCFR {
        set ragent [$self create-ldcfr-agent $node]
  }
```

From line 860 of the same file following code should be added.

```
 Simulator instproc create-ldcfr-agent { node } {
        #  Create LDCFR routing agent
        set ragent [new Agent/LDCFR [$node node-addr]]
        $self at 0.0 "$ragent start"
        $node set ragent_ $ragent
        return $ragent
 }
```

Now, set port numbers of routing agent. sport is source port, dport is destination port. Modify ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-agent.tcl** line 202

```
  Agent/LDCFR instproc init args {
        $self next $args
  }

  Agent/LDCFR set sport_   0
  Agent/LDCFR set dport_   0
```

Add following in the file: ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-mobilenode.tcl** line 201

```
  # Special processing for LDCFR
  set ldcfronly [string first "LDCFR" [$agent info class]]
  if {$ldcfronly != -1 } {
        $agent if-queue [$self set ifq_(0)]   ;# ifq between LL and MAC
  }
```

Go to ~/ns-allinone-2.34/ns-2.34/ directory and do

```
make clean
make
```

APPENDIX II

Working on NS-2

NS-2 consists of two languages Otcl and C++. Tclcl is used to create linkage between these. So, working on NS-2 deals with the understanding of Otcl and C++. For event simulation, Otcl is used while development work like the logic of a protocol is simulated in C++. (We have worked on both)

First, the registration of a protocol is done as shown in appendix I. After registration, we generally have three or four files to work with in NS-2, these are:

protoname/protoname pkt.h
protoname/protoname.h
protoname/protoname.cc
protoname/protoname rtable.h

General Format of protoname/protoname pkt.h

protoname/protoname_pkt.h
```
1: #ifndef __protoname_pkt_h__
2: #define __protoname_pkt_h__
3:
4: #include <packet.h>
5:
6: #define HDR_PROTONAME_PKT(p) hdr_protoname_pkt::access(p)
5
7:
8: struct hdr_protoname_pkt {
9:
10: nsaddr_t pkt_src_; // Node which originated this packet
11: u_int16_t pkt_len_; // Packet length (in bytes)
12: u_int8_t pkt_seq_num_; // Packet sequence number
13:
14: inline nsaddr_t& pkt_src() { return pkt_src_; }
15: inline u_int16_t& pkt_len() { return pkt_len_; }
16: inline u_int8_t& pkt_seq_num() { return pkt_seq_num_; }
17:
18: static int offset_;
19: inline static int& offset() { return offset_; }
20: inline static hdr_protoname_pkt* access(const Packet* p) {
21: return (hdr_protoname_pkt*)p->access(offset_);
22: }
23:
24: };
25:

26: #endif
```

General Format of protoname/protoname.cc

protoname/protoname.h

```
1: #ifndef __protoname_h__
2: #define __protoname_h__
3:
4: #include "protoname_pkt.h"
5: #include <agent.h>
6: #include <packet.h>
7: #include <trace.h>
8: #include <timer-handler.h>
9: #include <random.h>
10: #include <classifier-port.h>
11:
12: #define CURRENT_TIME Scheduler::instance().clock()
13: #define JITTER (Random::uniform()*0.5)
14:
15: class Protoname; // forward declaration
16:
17: /* Timers */
18:
19: class Protoname_PktTimer : public TimerHandler {
20: public:
21: Protoname_PktTimer(Protoname* agent) : TimerHandler() {
22: agent_ = agent;
23: }
24: protected:
25: Protoname* agent_;

26: virtual void expire(Event* e);
27: };
28:
29: /* Agent */
30:
31: class Protoname : public Agent {
32:
33: /* Friends */
34: friend class Protoname_PktTimer;
35:
36: /* Private members */
37: nsaddr_t ra_addr_;
38: protoname_state state_;
39: protoname_rtable rtable_;
40: int accesible_var_;
41: u_int8_t seq_num_;
42:
43: protected:
44:
45: PortClassifier* dmux_; // For passing packets up to agents.
46: Trace* logtarget_; // For logging.
47: Protoname_PktTimer pkt_timer_; // Timer for sending packets.
48:
49: inline nsaddr_t& ra_addr() { return ra_addr_; }
50: inline protoname_state& state() { return state_; }
51: inline int& accessible_var() { return accessible_var_; }
52:
53: void forward_data(Packet*);
```

```
54: void recv_protoname_pkt(Packet*);
55: void send_protoname_pkt();
56:
57: void reset_protoname_pkt_timer();
58:
59: public:
60:
61: Protoname(nsaddr_t);
62: int command(int, const char*const*);
63: void recv(Packet*, Handler*);
64:
65: };
66:
67: #endif
```

Tcl hooks

protoname/protoname.cc

```
1: static class ProtonameClass : public TclClass {
2: public:
3: ProtonameClass() : TclClass("Agent/Protoname") {}
4: TclObject* create(int argc, const char*const* argv) {
5: assert(argc == 5);
6: return (new Protoname((nsaddr_t)Address::instance().str2addr(argv[4])));
7: }
8: } class_rtProtoProtoname;
```

General Format of protoname/protoname rtable.h

protoname/protoname_rtable.h

```
1: #ifndef __protoname_rtable_h__
2: #define __protoname_rtable_h__
3:
4: #include <trace.h>
5: #include <map>
6:
7: typedef std::map<nsaddr_t, nsaddr_t> rtable_t;
8:
9: class protoname_rtable {
10:
11: rtable_t rt_;
12:
13: public:
14:
15: protoname_rtable();
16: void print(Trace*);
17: void clear();
18: void rm_entry(nsaddr_t);
19: void add_entry(nsaddr_t, nsaddr_t);
20: nsaddr_t lookup(nsaddr_t);
18
21: u_int32_t size();
22: };
23:
24: #endif
```

These portions of these files are used as such by just changing ‘protoname’ with the protocol name we want to work on. All the logic can be made independently using the common as well as exclusive made routines.

The linkage between tcl and C++ is generally done with the bind function whose syntax is shown below

```
1: Protoname::Protoname(nsaddr_t id) : Agent(PT_PROTONAME), pkt_timer_(this) {
2: bind_bool("accessible_var_", &accessible_var_); // binding is being done
3: ra_addr_ = id;
4: }
```

Accessing from Tcl scripts is fairly simple. The next example sets the value of *accesible var* to *true*.

simulation.tcl

```
1: Agent/Protoname set accesible_var_ true
```

Calculation of Throughput in NS-2

1: Attach the Loss Monitor with the sink as shown below

set sink [new agent/loss monitor]

2: Now, take the values of bytes_ field in Loss Monitor

3: Convert bytes into bits and divide by time period in which these bits are received at the BS, this gives the throughput in b/s.

Calculation of Latency in NS-2

1: Modify the packet.h file

2: Add a time-stamp for the packet which stores the time at which a packet is sent by a node

3: At the sink, change the ‘LossMonitor’ to determine the time at a given packet is received

4: The difference between instant at which a packet is received and its time-stamp, gives us latency

Calculation of Average Network Energy in NS-2

1: At a certain instant of time, we get the energy of all the nodes in our network in C++ in ns2

2: This aggregate energy of all the nodes is then averaged by dividing the number of nodes in the network

3: We do TCL and C++ linkage to get the average energy in TCL

Thus, after a certain time-interval, we take the values of average energy and plot the result with Xgraph

BIBLIOGRAPHY

Citations :

[1](throughput-8) C. Peraki, S. D. Servetto. \On the Maximum Stable Throughput Problem in Random Networks with Directional Antennas," in Proc. of the 4th ACM MobiHoc, June 2003.

[2](throughput -8) Gupta and P. R. Kumar, \The Capacity of Wireless Networks", in IEEE Transactions on Information Theory, 46(2):388-404, March 2000.

[2] (throughput-8) Chen Wang and Li Xiao, \Improving Event-to-Sink Throughput in Wireless Sensor Networks, 2007

[2] (9-ucr) O. Younis and S. Fahmy, HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks, IEEE Transactions on Mobile Computing 3(4) (2004) 660–669

[2] (27-ucr) O. Younis and S. Fahmy, An experimental study of routing and data aggregation in sensor networks, in: Proceedings of the International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (November 2005) pp. 50–57

[2] (29-ucr) V. Kawadia and P.R. Kumar, Power control and clustering in ad hoc networks, in: Proceedings of IEEE INFOCOM (April 2003) pp. 459–469

[2] (26-heed) F. Ye, G. Zhong, S. Lu, and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks," in International Conferenece on Distributed Computing Systems (ICDCS), 2003.

[2] (23-heed)F. Ye, H. Luo, J. Chung, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Protocol for Large-Scale Wireless Sensor Networks," in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), Atlanta, Georgia, September 2002.

[2] (4-heed)S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar, "Power Control in Ad-Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW protocol," in Proceedings of European Wireless 2002. Next Generation Wireless Networks: Technologies, Protocols, Services and Applications, February 2002, pp. 156–162.

[2] (5-heed) C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), 2000.

[2] (27-heed)D. M. Blough and P. Santi, "Investigating Upper Bounds on Network Lifetime Extension for Cell-Based Energy Conservation Techniques in Stationary Ad Hoc Networks," in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), 2002

General References

[1] Laiali Almazaydeh, Eman Abdelfattah, Manal Al- Bzoor, and Amer Al- Rahayfeh performance evaluation of routing protocols in wireless sensor networks , April 2010

[2] Sotiris Nikoletseas and Paul G. Spirakis Probabilistic Distributed Algorithms for Energy Efficient Routing and Tracking inWireless Sensor Networks, February 2009

[3] Zhang Jin, Yu Jian-Ping, Zhou Si-Wang, Lin Ya-Ping and Li Guang A Survey on Position-Based Routing Algorithms in Wireless Sensor Networks, February 2009

[4] Guihai Chen · Chengfa Li · Mao Ye · JieWu 'An unequal cluster-based routing protocol in wireless sensor networks' 2007

*[5] Ian F. Akyildiz *, Tommaso Melodia, Kaushik R. Chowdhury, A survey on wireless multimedia sensor networks, November 2006*

[6] Zhihua Hu, Baochun Li /fundamental performance limits Of wireless sensor networks, 2005

[7] Francisco J. Ros Pedro M. RuizImplementing a New Manet Unicast Routing Protocol in NS2, December, 2004