1-1-2007

REEP : a data-centric, reliable and energy-efficient routing protocol for wireless sensor networks

Farhana Zabin
Ryerson University

Recommended Citation

Zabin, Farhana, "REEP : a data-centric, reliable and energy-efficient routing protocol for wireless sensor networks" (2007). *Theses and dissertations.* Paper 340.

REEP: A DATA-CENTRIC, RELIABLE AND ENERGY-EFFICIENT ROUITNG PROTOCOL FOR WIRELESS SENSOR NETWORKS

routing

by

FARHANA ZABIN

A Thesis
Submitted to the Faculty of Graduate Studies
through Computer Network
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
Ryerson University

Toronto, Ontario, Canada, 2007

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

FARHANA ZABIN

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

FARHANA ZABIN

September 13, 2007

ABSTRACT

Recent advances in sensor technology and wireless communications have led to many new data dissemination routing protocols, especially designed for wireless sensor networks, where energy awareness is the most important consideration. The focus of this thesis is on the area of routing protocols for wireless sensor networks, especially for those applications where, the whole sensor field need to be taken under observation to detect available different types of moving objects. Besides the efficient use of limited energy, reliability is another important issue in sensor communication, where the network is susceptible to environmental factors. In this thesis, the design of a new energy-efficient data-centric routing protocol, named *Reliable and Energy-Efficient Protocol* (REEP), is proposed. We have used MATLAB 7.4 for our implementation. The performance of REEP has been compared with Directed Diffusion (DD) for the aforementioned sensor network application. Our simulations and experimental results show that REEP performs better than DD.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my thesis supervisors, Dr. Isaac Woungang, Department of Computer Science and Dr. Sudip Misra, Department of Computer Science for their caring, guidance and cooperation throughout my graduate studies. This work could not have been achieved success without their help and support.

My heartiest gratitude goes to the Examiners, Dr. Muhammad Jaseemuddin, Department of Electrical and Computer Engineering and Dr. Ivan Lee, Department of Electrical and Computer Engineering for being in the committee and spending their valuable time.

I also would like to express my sincere gratitude to Dr. Bobby Ma, Department of Electrical and Computer Engineering for serving as the chair of the defense.

I extend my deepest appreciation to Mohammad Ahsan Ali for his valuable suggestions and tremendous help throughout my thesis work.

DEDICATION

To my parents,
who have been always great inspiration to me
and my dear husband,
who always stays beside me in all my needs

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Advanced development in wireless technology is creating a great impact on our daily life style. Technologies we are using more often are becoming smarter in a sense that they are getting smaller and more intelligent (i.e., capable of doing complex computations). More over, many of them are getting wireless. For example, devices such as cell phones, PDA, laptops, Palmtops, are all capable of wireless communication. It is not unreasonable to expect that over the years, the world may be covered with sensor networks with access to them via the Internet backbone. Wireless communication is one of the biggest inventions of the twenty first century. Researchers are not stubbed in one area of wireless communication. As long as their research fields are exploring and developing, our day-to-day demands and needs are also expanding. The use of wireless sensor devices in numerous application areas (such as medical, civil, military, etc.) is accelerating the development of technology.

Any wireless sensor network is a real-time distributed system. In general, distributed systems are equipped with unlimited power, wired, location independent, not real-time, having a fixed set of resources and user interfaces such as screen and mice [AS02][AK04][AY05]. In contrast, the wireless sensor network systems are wireless, having scarce power, are real-time, uses sensors or actuators as interfaces, having dynamic resources and location [AS02][AK04][AY05]. This sensor network system assures the accuracy and the availability of information that can significantly improve the quality of information as well as the way of gathering them [CK03][CE04]. Unlike the other network systems, sensor network can help to get the real time information with the reduced cost and higher fidelity, which is hard to get.

1.1 Introduction to Wireless Sensor Networks

A wireless sensor network typically consists of a large number of sensor nodes, densely deployed over an area. Generally, the algorithm for WSN is implicitly a distributed algorithm. Sensor nodes are capable of collaborating with each other and measuring the condition of their surrounding environments (i.e. light, temperature, sound, vibration). The sensed measurements are then transformed into digital signals and processed to reveal some properties of the phenomena around sensors. Those collected information are routed back to the user through the multi-hop infrastructure-less architecture of WSN. In such network, each sensor node plays the dual role of a data originator and a router to route the data along specified path. Due to the fact that the sensor nodes in WSN have short radio transmission range, intermediate nodes act as relay nodes to transmit data towards the sink node using a multi-hop path.

Wireless sensor nodes are the fundamental elements to construct a wireless sensor network. The basic structure and functionalities of typical sensor nodes are described in the next section. There are some important design issues, which need to be taken into account while designing any sensor network routing protocol. A wide range of different application areas (such as military, medical, environmental, space, etc.) of wireless sensor networks are highlighted in section 1.1.3. Based on these different application areas, the way of deployment is also different in each case.

1.1.1 Sensor Nodes

Sensor nodes are tiny in size compared to the traditional nodes (fully functional desktop Personal Computer (PC) or Laptop). The functionalities of sensor devices are almost similar to those of the traditional devices and the role of a sensor node in a network is to work as a router. They have operating system like desktops or Laptops known as TinyOS, a microprocessor for processing limited instructions and power supplied by battery only. Therefore these tiny nodes have less longevity than traditional nodes.

Wireless sensor nodes are composed of sensing technology, data processing and wireless communication infrastructure. Advanced development in MEMS (Micro Electro Mechanical System) and RF (Radio Frequency) design enables multifunctional sensor nodes to become economically and technically feasible, which is cheaper and smaller in size (see figure 1.1). These multifunctional nodes have short transmission range because of their limited radio capabilities. The numbers of sensor nodes are used depending on the application type. For example, this number may be different when dealing with agricultural applications such as soil test, or with environmental science applications such as humidity or temperature test.

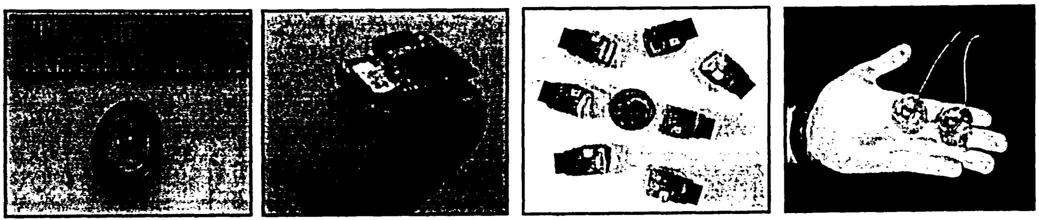

(a) (b) (c) (d)

Figure 1.1: Sensor nodes of different sizes
(a) Daft Dust [BRK1] (b) Eco [CHOU] (c) Eco [MOTE] (d) dots [BRK2]

The architecture of a sensor node consists of four main components. They are: *sensing unit*, *processing unit*, *transceiver unit* and *power unit*, illustrated in figure 1.2. In addition to them, sensor nodes can have *location finding system* to acquire the knowledge of node's location with high accuracy, which is mostly required in sensing task and routing techniques. Some other application dependent subunits can be installed with the sensor nodes, such as *power generator* and *mobilizer unit* that is required when an application needs a node to carry out a specific task. The sensing unit is made up of *Sensor* and *ADC* (Analog to Digital Converter). The analog signals are captured by the sensors from the environment and converted into digital signals by ADC and inserted into the processing unit for further processing. The processing unit includes a processor attached with a small storage unit that carries out all procedures and information needed to co-operate with the other sensor nodes. The transceiver unit of a sensor node enables the node to participate in the network. The power unit is one of the most important units in terms of the source of energy.

All of these units are require fitting into a small sized box, even smaller than a cubic centimeter [PK00]. Apart the size, sensor nodes have other constraints, for example, they must consume extremely low power, operate on an unattended manner in a highly dense area, they should have low production cost and be dispensable, autonomous and adaptive to the environment [KK99].

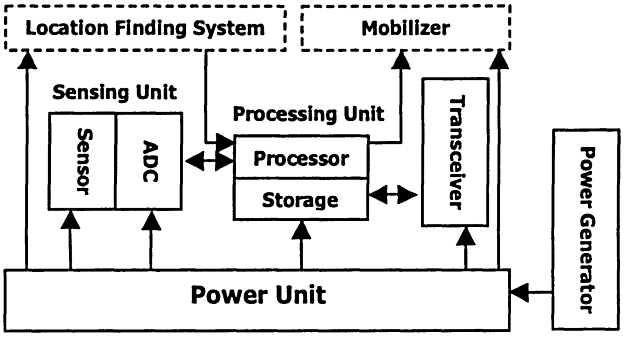


Figure 1.2: Basic components of a Sensor node; redrawn from [AS02]

1.1.2 Design issues

The design of a wireless sensor network is influenced by some important design factors which serve as guidelines for designing a protocol or an algorithm for any type of sensor network. These factors are as follows: scalability, production costs, operating environment, sensor network topology, hardware constrains, transmission media, fault tolerance and power consumption [AS02]. Among these, the factors fault tolerance and power consumption are the most important in any design of energy efficient protocols for WSN.

During data transmission, any node can fail due to lack of power or can be blocked by any obstacle or it can get undesirable physical or environmental damage to the node. In such situations, the ability to sustain by avoiding any interruption of network functionality assures more reliable transmission of information, which is known as fault tolerance issue. This fault tolerance can be modeled [HS00] to find out the probability of not having a failure within a specified time interval. Sensor nodes are often energy constrained because they are battery-powered device. The main tasks of a sensor node that involve power consumption are: sensing, data processing, and communication. Therefore, application specific protocols can be designed by trading off other performance metrics with the energy consumption. Two main points should be taken into account when designing a protocol for WSN. The first concern is to maintain the level of power consumption at each stage of functionalities. the second one is to achieve fault tolerance in case of any type of failure.

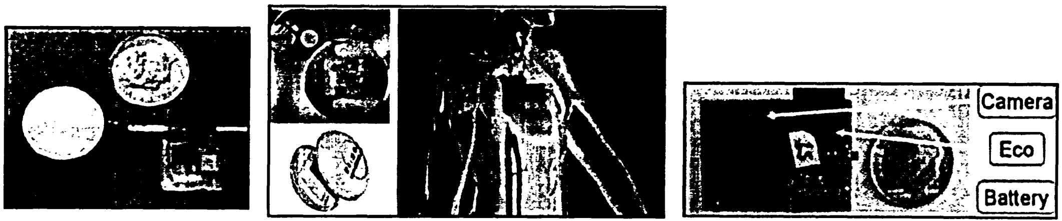
1.1.3 Applications

There exist a large variety of applications of wireless sensor networks. The distributed wireless micro-sensor systems enable the reliable monitoring for commercial, industrial, civil and military applications. Different types of sensor nodes are used in different applications according to the types of applications and their requirements. Such type of applications include environment monitoring, home automation, traffic control, fire detection, object tracking, nuclear reactor controlling, chemical processing, space exploration, disaster relief, habitat monitoring [MP02] and healthcare applications etc.
Some typical environment monitoring applications are as follows [AS02][AK04][AY05]:

- Temperature measurement
- Humidity level
- Lighting condition
- Air pressure
- Soil makeup
- Noise level
- Vibration

There are many advanced medical applications are available nowadays that use wireless sensors to monitor the condition of patients. A sensor node, called Eco [CHOU] has been designed to be worn on the limbs of pre-term infants to monitor their movement in response to the assisted exercises (figure 1.1 (b)). A wireless ultra-wearable low power ECG monitoring system [PCB06] has been developed to monitor the heartbeat of a patient to check the physiological condition (figure 1.3 (a)). HBS (Handheld Breast Scanner) [NX06] has been developed for non-invasive breast cancer detection that works based on the principle of frequency domain photon migration spectroscopy. The goal of BiosensorNet [DSON] project is to create an intelligent, self-managing, context-aware biosensing networking to improve the patient care (figure 1.3 (b)). Cardiac status of a patient is monitored via implanted sensors that relay information to the on-body wireless node. The node then collects information from multiple sensors and sends it to the patient's phone or PDA. These are then forwarded to a remote medical monitoring system.

The eCAM [PC06] sensor node is an ultra compact, high data rate node with a miniature camera (figure 1.3 (c)). This node has been constructed with a VGA quality digital video camera with the Eco node. It outputs real-time reliable streaming of VGA quality video, which can be used in highly secured zones for intrusion detection.



(a) ECG sensor and Eco with a dime coin [PCB06]

(b) Cardiac monitoring via implanted sensor node [DSON]

(c) eCAM with a dime coin [PC06]

Figure 1.3: Wireless sensor nodes developed for different applications

The science and engineering goal of the Smart Dust project [KK99][PSTR] is to build a self-contained complete and complex system in a tiny volume using state-of-the-art technologies for a massively distributed sensor network. The Piconet project [BC97] is mainly focused on office and home information discovery and automation. The WINS project [PK0] has come up with distinctive progress in radio designs for low power environmental sensing and also focused on low level network synchronization.

1.1.4 Deployment

Deployment of sensor nodes, over the area of interest, mainly depends on the type of application, because most of the sensor network protocols are application specific. Therefore, the node deployment within the sensor field can be either grid based with pre-planning engineering deployment or scattered deployment without any pre-organization. In this way, hundreds to thousands of sensor nodes can be deployed statically all over the targeted area.

There are three deployment phases in wireless sensor networks [AS02]. During the first phase, the *pre-deployment* or *deployment phase*, a sheer number of nodes are required in each application and are deployed by throwing off from aircraft or ship-board, delivering from an artillery shell, rocket or missile or placing them one by one either by human or robot. In the second phase, the *post-deployment phase*, topologies are prone to frequent changes due to requirements of different tasks, mobility of node's position, malfunctioning, availability of less energy etc. Therefore, any type of failure is the most common and expected event in this phase. During the last phase, the *re-deployment of additional nodes phase*, additional nodes are replaced with the malfunctioning nodes or extra nodes are re-deployed in need of task dynamics.

The self-organizing and collaborative characteristics of sensor nodes help them to find their neighbors within their vicinity. Based on different types of applications, sensor nodes can be stationary or mobile in a network. Mobile nodes are more intelligent compared to stationary nodes because their self-organizing characteristics lead them to track over their changing locations during random changes of the topology.

1.2 Introduction to Energy-Efficient Protocols for WSN

Rather than concentrating on the general purpose protocols for WSN, the work in this thesis has been originated from the study of challenges encountered by the energy-efficient family of protocols for wireless sensor networks. The next section will explain the importance of energy-efficient protocols in the WSN context as well as the aforementioned challenges and finally we will describe the problem statement and proposed solution.

1.2.1 Importance of Energy-Efficient Routing Protocols

In order to meet the requirements of different types of application, the correct processing of collected information and their appropriate routing are the important considerations in a routing protocol. These steps can be accomplished by introducing an efficient use of energy in sensor nodes. Each sensor node in the network is battery-equipped, and is therefore, limited in terms of the energy. Consequently the network lifetime in sensor networks becomes limited. In most of the sensor network applications replenishment of power resources is nearly impossible. Another important factor that influences the consumption of more power in sensor networks is that each sensor node consumes power not only for sensing, but also for processing the sensed data and transmitting/receiving them to/from neighbors. These explain why the efficient use of power is the primary and most important consideration for designing a sensor network protocol.

1.2.2 Challenges of Energy-Efficient Routing Protocols

As mentioned earlier, power is the scarcest resource in wireless sensor networking. Therefore, designing an energy-efficient routing protocol for WSN is one of the most challenging tasks for researchers nowadays. Some important factors should be taken into account when designing an energy-efficient routing protocol for WSN. These factors lead to the savings of energy, such as the avoidance of unnecessary flooding of control information, and the minimization of complex computation for data processing and controlling over the sensing tasks. To meet the challenges for energy efficiency, there are some other issues such as quantifying the data loss ratio, finding a reliable path for data transmission, and handling fault tolerance. These issues should be addressed when designing a new protocol for WSN.

1.3 Problem Statement and Proposed Solution

The detailed studies of different energy-efficiency based protocols give us the realization of some drawbacks of some of their functionalities. Given below are the highlights of those shortcomings:

- Nodes are not aware of the available energy resources, thus, nodes may instantaneously fail when they are out of energy during the data transmission process.

- Queries in sensor field are sometimes specific to some tasks, which do not provide an overall summary of the detected events in the network. So, user needs to send different queries for different tasks.

- In case of path failure, for alternate path discovery, the use of periodic and repeatedly low rate flooding of control information consumes more power.

These drawbacks have motivated us towards the design of a new routing protocol for WSN that will adopt some mechanisms in order to achieve a better energy efficiency and reliability. In this thesis, we are going to design such a protocol called *Reliable and Energy-Efficient Protocol - REEP* (see details in the Section entitled "Design of REEP" in Chapter 4).

1.4 Objective of this Thesis

Designing an energy efficient protocol for WSN is becoming a challenging research area nowadays. Apart from the military applications, lots of civil, environmental and commercial applications are also giving attention to the use of sensor networks. To meet the various types of demands in different kinds of applications, lots of research works are evolving that need cheap, long lasting and relatively small size of sensor nodes with more computational capability.

The existing WSN routing protocols can be classified into two main categories: *data centric* and *hierarchical*. Hierarchical protocols need more complex computation for electing cluster heads and maintaining clusters in addition of routing. Therefore, our work is mainly focused on data centric category, where all communication is done based on Meta-data (data are named by their attributes). Some early works [IG03][KR99][HS01] have shown the use of meta-data defined by different naming schemes. Our main concern in this thesis is to design a *reliable* and *energy-efficient* protocol, which can meet the demand of fault tolerance with increasing life time of the network.

1.5 Thesis Organization

In addition to this Chapter, this thesis has 5 other chapters.

Chapter 1 (*Introduction*): This chapter is a general introduction to WSN. It includes basics structure of sensor nodes, different applications of WSN in various fields, WSN deployment phases, and some design factors governing the communication architecture of sensor networking. The importance and challenges of energy-efficient protocols are highlighted, along with the problem statement and our proposed solution.

Chapter 2 (*Background Work*): In this chapter, many existing mobile ad-hoc networking protocol names are mentioned, and the different types of sensor network protocols encountered per OSI layers are described briefly.

Chapter 3 (*Non-Functional and Functional Requirements of REEP*): In this chapter, the non-functional requirements of REEP such as reliability, efficiency, usability and functional requirements of REEP such as task description, resource discovery, are discussed.

Chapter 4 *(Design of REEP)*: In this chapter, our proposed REEP protocol is described, along with its features. These include: the way REEP handles energy usage, fault tolerance and usability. Some advantages and disadvantages of REEP are also highlighted.

Chapter 5 (*Evaluation and Validation of Design*): In this chapter, we have compared the performance of REEP with DD for aforementioned application type. The implementation steps of both protocols are also described here followed by a comparison of the REEP and DD protocols in terms of some predefined performance metrics. These results are shown to satisfy both our predefined functional and non-functional requirements of REEP.

Chapter 6 (*Conclusion and Future Works*): This chapter concludes the thesis work, including the thesis contributions, discussions and directions for future works.

Chapter 2

BACKGROUND WORK

Recent advanced development of MEMS (Micro Electro Mechanical System) technology has opened a large research area for many researchers. Different protocols and algorithms for use in sensor networks are being proposed for different types of applications in both civilian and military domains. Our work relates to civil area applications of sensor networks and is inspired by many other previous research results [IG03][KR99]. The most crucial and sensitive resource in a sensor network is the available energy of each sensor node. So, efficient consumption of energy should be the most important consideration while designing any new scheme for wireless sensor network.

2.1 Traditional Wireless Ad-hoc Network Protocols

The unique characteristics of mobile networks and advances of wireless communication technology have attracted significant interest in the last few decades. Compared to wired networks, wireless MANET (Mobile Ad-hoc NETworks) is more flexible and inexpensive. Some of the important characteristics of wireless ad-hoc networking are: capable of handling dynamic changes in topology, infrastructureless, unstable link capacity, and limited transmission ranges. Considering these features, researchers have proposed many routing protocols in the last few decades for MANET. These routing protocols can be classified into the following categories [LK03]:

- ► *Pro-active Routing protocol (table driven),* such as: DSDV (Destination-Sequenced Distance Vector), WRP (Wireless Routing Protocol), FSR (Fisheye State Routing) [LK03], AWDS (Ad-hoc Wireless Distribution Service)-Layer 2 wireless mesh routing protocol, Babel - a variant of AODV with faster convergence, HSR (Hierarchical State Routing), TBRPF (Topology Broadcast based on Reverse-Path Forwarding routing protocol) [WIKI].

- *Reactive Routing Protocol (On-demand),* such as: DSR (Dynamic Source Routing), AODV (Ad-hoc Ondemand Distance Vector), TORA (Temporally-Ordered Routing Algorithm) [LK03], ARA (Ant-based Routing Alogorithm) [GS02], DYMO (DYnamic Manet On-demand), MOR (Multipath On-demand Routing) [WIKI].

- *Zone based hierarchical Routing Protocols* such as: ZRP (Zone Routing Protocol), HARP (Hybrid Ad hoc Routing Protocol), ZHLS (Zone-based Hierarchical Link State routing) [LK03].

- *Cluster based Routing Protocols* such as: CGSR (Cluster-head Gateway Switch Routing), HSR (Hierarchical State Routing), CBRP (Cluster Based Routing Protocol) [LK03], DART (Dynamic Address Routing), DDR (Distributed Dynamic Routing), GSR (Global State Routing) [WIKI].

- *Core node based Routing Protocols* such as: LANMAR (Landmark Ad hoc Routing), CEDAR (Core-Extraction Distributed Ad Hoc Routing), OLSR (Optimized Link State Routing) [LK03].

- *Location information based Routing Protocols* such as: LAR (Location Aided Routing), DREAM (Distance Routing Effect Algorithm for Mobility), GLS (Grid Location Service) [LK03], ALERM (Adaptive Location Aided Routing protocols-Mines), BGR (Blind Geographic Routing), GPSAL (GPS Ant-Like algorithm) [WIKI].

- *Link stability based Routing Protocols* such as: ABR (Associativity Based Routing), SSR (Signal Stability-Based Adaptive Routing) [LK03].

- *Multicast Routing Protocols* such as: AMRoute (Ad-hoc Multicast Routing), AMRIS (Ad hoc Multicast Routing protocol utilizing Increasing id-numberS), ODMRP (On-Demand Multicast Routing Protocol), CAMP (Core-Assisted Mesh protocol), MAODV (Multicast operation of Ad-hoc On-demand Distance Vector) [LK03], AQM (Ad Hoc QoS Multicast), CBM (Content Based Multicast), DCMP (Dynamic Core Based Multicast Routing Protocol), SMF (Simplified Multicast Forwarding), MZR (Multicast Zone Routing), LAM (Lightweight Adaptive Multicast) [WIKI].

- *Power Aware Routing protocol* [WIKI] such as: PARO (Power-Aware Routing Optimization), EADSR (Energy Aware Dynamic Source Routing), PAMAS (Power Aware Multi Access protocol with Signaling ad-hoc networks).

- *Geographical Multicast (Geocasting)* [WIKI] such as: LBM (Location Based Multicast), GeoGRID (Geographical GRID), GeoTORA (Geographical TORA), MOBICAST (Mobile Just-in-time Multicasting), Abiding Geocast / Stored Geocast (Time Stable Geocasting).

2.1.1 Why MANET Protocols are not Suitable for WSN

Although there are many proposed protocols available for traditional wireless ad-hoc networking (described in the previous section), they are not well suited for the unique features and application requirements of WSN. The main concern of wireless ad-hoc routing protocols is to achieve high quality of service (QoS), whereas that of sensor network protocols is to prolong the lifetime of the network.

Wireless sensor networks differ from traditional wireless ad-hoc network in many ways. They are as follows:

- *Deployment method:* In most of the wireless sensor network applications, all the sensor nodes are densely deployed over the deployment surface either deterministically or randomly.

- *Source of energy:* Sensor nodes are equipped with irreplaceable source of limited power (battery power) because of their tiny size. Wireless nodes in ad-hoc networks have replaceable source of powerful battery power.

- *Node identification:* Sensor nodes are not addressed with global identification (i.e. IP addresses) because of their sheer number of deployment and maintenance overhead due to frequent topology changes.

- *Size of the nodes:* Sensor nodes are generally small in size and limited in transmission range, memory and computational capabilities compared to powerful nodes in traditional wireless network. The size of nodes in sensor networks has a certain advantage when deploying the network.

- *Number of nodes in a network:* Compared to the other ad-hoc networks, the number of sensor nodes in a sensor network can be higher in a several orders of magnitude.

- *Mobility of nodes:* Depending on the types of application, sensor nodes can be either stationary or mobile. In most cases, they are subject to stationary. Topology changes in sensor network are very frequent in some applications like battlefield. Hence, sensor nodes are prone to failure.

- *Communication type:* Sensor nodes do not follow the point-to-point communication as traditional nodes in other ad-hoc networking. Rather, they follow broadcasting techniques. In senor network all the sensor nodes work in a group to gather information, whereas in ad-hoc networks each node may represent a different user.

2.2 Wireless Sensor Network Protocols

Wireless sensor networking is a distributed communication system where sensor nodes are densely scattered throughout the sensor field. They communicate with each other in an ad-hoc style. The main task of sensor nodes in a sensor field is to detect events, perform data processing based on local information and then route the data along specified path. Data are routed back to sink by a multi-hop infrastructureless architecture. These sensor nodes are typically smaller in size, less costly, low powered and capable of communicating within short distance and performing a limited local computation. Hence, network protocols and algorithms must possess self-organizing capability without an administration node and capable of working with highly dynamic topology.

In the next few sections, we will briefly discuss about the protocol stack with different layers and their corresponding available WSN protocols.

2.2.1 Protocol Stack for WSN

The protocol stack of WSN is used by every sensor node in the network and consists of five layers, illustrated in figure 2.1. They are as follows: *Physical layer*, *Data link layer*, *Network layer*, *Transport layer* and *Application layer.* All these layers are associated with three management planes. They are *Power management plane*, *Mobility management plane* and *Task management plane.* These three management planes help sensor nodes to coordinate the sensing task and control the mobility and overall power consumption in each layer. It should be noted that, in this thesis we focus only on the Network layer, which is responsible for appropriate routing of the data supplied by Transport layer.

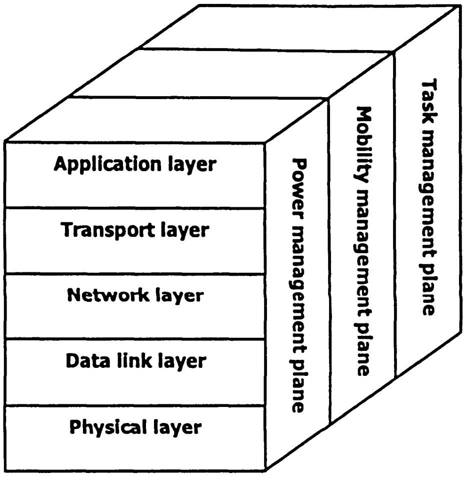


Figure 2.1: Protocol Stack for Sensor Network; redrawn from [AS02]

All sensor nodes in the network use the above protocol stack, illustrated in figure 2.1. Some of the running projects [KK99][PK00] are using this protocol stack with different combination of layers. For instance, in smart dust project [KK99][PSTR], motes (sensor nodes) use the protocol stack that includes application layer, MAC layer and physical layer only. Protocol stack in WINS project [PK00] includes application layer, network layer, MAC layer and physical layer.

2.2.2 Different Layer Protocols for WSN

Several types of protocols have been designed by many protocol designers for different layers of the protocol stack (Figure 2.1). Our thesis work is mainly focused on the network layer protocols and is related to some of the existing protocols (see related works in Section 2.3). Protocols for each layer and their categories are described below in brief.

(I) Application Layer

Protocols for WSN at the application layer are still unexplored and they remain as open research issues. In [AS02], three possible application layer protocols for WSN are highlighted, which are as follows: Sensor Management Protocol (SMP), Task

Assignment and Data Advertisement Protocol (TADAP), and Sensor Query and Data Dissemination Protocol (SQDDP). The Sensor Query and Tasking Language (SQTL) is also proposed in [SSJ01] as a sub-category of SQDDP. As mentioned earlier, sensor networks are infrastructureless, therefore, to manage sensor nodes we may need the SMP because SMP allows software to perform administrative tasks such as turning sensor nodes on and off, moving sensor nodes, synchronizing the time of nodes and reconfiguring the sensor network, if necessary [AS02].

(II) Transport Layer

Currently, there is no transport layer protocols proposed for sensor networks so far [AS02]. A special transport layer protocol will be needed when communication within sensor network needs access to the Internet or other networks. TCP ensures end-to-end communication based on global addressing in traditional network. As a matter of fact, sensor nodes are not globally identified by any IP address, so transport layer protocol for sensor network should consider the attribute based naming scheme to indicate the destination of data packets [AS02]. This attribute based naming scheme is described in detail in Chapter 3. Designing a transport layer protocol for sensor networks is currently a challenging research area nowadays. The reason behind this is that sensor nodes are limited in memory and power, thus, cannot store large amount of data like the servers in the Internet [AS02]. In addition, global ID for sensor nodes and acknowledgement techniques are too costly for any sensor network application. Therefore, authors in [AS02] have mentioned that a new scheme (that split the end-to-end communication at the sink nodes) might be needed when UDP-type protocols are used in the sensor networks and traditional TCP/UDP protocols are used in the Internet or Satellite networks.

(III) Network Layer

Most of the existing protocols that have been proposed so far for WSN pertain to the network layer of the protocol stack (Figure 2.1). These network layer protocols have been designed based on some basic principles [AS02]:

- Power consumption is the most important issue when designing a protocol for sensor network.
- Data aggregation is necessary, because sensor network is mostly data-centric where data redundancy is a usual phenomenon.
- Attribute based addressing and location-awareness are two features of an ideal sensor network routing protocol.

Based on power efficiency issue, authors in [AS02] have offered some design choices to protocol designers for selecting an energy efficient route. They are:

- *Maximum available power (PA) route:* the route that has maximum total available power is preferred.
- *Minimum energy (ME) route:* the route that consumes minimum energy to transmit the data packets between the sink and the sensor nodes.
- *Minimum hop (MH) route:* the route that takes the minimum number of hops to reach the sink node is preferred.
- *Maximum minimum PA node route:* the route along which the minimum PA is the largest of the minimum PAs of the other routes is preferred.

Sensor network protocols at the network layer of the protocol stack can be classified into three categories [AK04]: *Data centric (or Flat) routing*, *Hierarchical routing* and *Location-based routing.*

Data centric (or Flat) Routing: Data-centric routing refers to the fact that all queries result in sampled data and are named by some of their attributes, and are routed based on those attribute values. This is also known as flat routing because all the nodes in the sensor network perform a distributed multi-hop routing and play the same role. In data-centric routing, the *interest* is disseminated throughout the network to perform sensing tasks. So far, there are two approaches that have been used for interest dissemination in data-centric routing [AS02]. The first one is where sinks broadcast the interest about a specific area [IG03] and the second one is where sensor nodes broadcast the advertisement of the sensed data and wait for request from interested nodes [KR99]. Data aggregation techniques have been considered as an important issue in data centric routing. They help in solving the implosion and overlap problems [KR99]. Both the Sensor Protocol for Information via Negotiation (SPIN) [KR99] and Directed Diffusion (DD) [IG03] (see Section 2.3.2) protocols have shown energy savings through data negotiation and elimination of redundant data. Other than the DD and SPIN protocols, there exists some more data-centric protocols [AK04], such as Rumor Routing, Minimum Cost Forwarding Algorithm (MCFA), Gradient Based Routing (GBR), Information Driven Sensor Querying (IDSQ), Constrained Anisotropic Diffusion Routing (CADR), Energy Aware Routing and Routing protocol with random walks [SB02].

Hierarchical routing: Hierarchical or cluster-based routing technique has its origin in traditional wireless networking. It is currently used in WSN for energy-efficient routing. The main advantage of this technique is that it ensures high scalability and efficient communication. Hierarchical routing performs in two phases. In the first phase, the cluster formation and cluster head selection tasks are mainly performed and in the second phase, data sensing, processing and routing sensed data are performed. In hierarchical routing, nodes can be homogeneous, i.e., only higher energy available nodes are selected as cluster heads) or heterogeneous, i.e., cluster heads are specialized nodes that are less energy constrained). The main functions of a cluster head are complex computation, data aggregation and communication with other cluster heads or sinks. Other nodes are responsible for sensing and relaying data to their corresponding cluster heads. Many hierarchical routing protocols are available: . Some of them are: Low Energy Adaptive Clustering Hierarchy (LEACH), Power Efficient GAthering in Sensor Information System (PEGASIS), Threshold-sensitive Energy Efficient sensor network protocol

(TEEN), Adaptive Periodic TEEN (APTEEN), Minimum Energy Communication Network (MECN) , Small MECN (SMECN), Self Organizing Protocol (SOP), Sensor Aggregates Routing (SAR), Virtual Grid Architecture (VGA), Hierarchical Power Aware Routing (HPAR), Two Tier Data Dissemination (TTDD) [AK04].

Location based routing: Location based routing refers to those protocols that use node's location information for routing. In this technique, nodes are addressed by a relative location information, which is measured by the estimated distance between two nodes on the basis of incoming signal strength [AK04]. An alternate way of finding the location of a node is using a satellite GPS system, which requires that nodes be equipped with small and low powered GPS receiver [XH01]. Other than data-centric and hierarchical routing protocols, there are some other proposed protocols which are location based routing protocols. They are Geographic Adaptive Fidelity (GAF), Geographic and Energy Aware Routing (GEAR), Most Forward within Radius (MFR), GEographic DIstance Routing (GEDIR), the Greedy Other Adaptive Face Routing (GOAFR) [XH01] and SPAN [AK04].

(IV) Data Link Layer (MAC layer)

There are some special kinds of protocols proposed for the MAC layer. The existing traditional MAC protocols of MANETs cannot be used for sensor network because of their frequent topology changes, resource constraints and application requirements. MAC protocols for a sensor network must have built-in power conservation, mobility management and failure recovery strategies [AK04]. MAC scheme requires a network infrastructure to establish communication links for the data transmission. On the contrary, most sensor networks are infrastructure-less in which most of of sensor nodes are scattered in the sensor field. The proposed MAC layer protocols for sensor network are Self-organizing MAC for Sensor network (SMACS), Eavesdrop-And-Register (EAR), Hybrid TDMA/FDMA-based and CSMA-based protocols [AS02].

(V) Physical Layer

Physical layer schemes mainly deal with some activities such as data encryption and decryption, modulation and frequency hopping, etc. The development of different schemes for physical layer in wireless sensor network is not a very wide research area so far. Ultra-wide-band (UWB) is a popular physical layer scheme. It is an attractive candidate for sensor network because of its simple transceiver circuitry and low transmission power [AS02]. It is well known that long distance wireless communication is expensive, both in terms of energy and implementation complexity [AS02]. Multi-hop communication in a sensor network can effectively overcome this problem. It can also overcome other problems like shadowing, fading, path-loss effects and diffraction [AS02].

2.3 Related Works

SPIN (Sensor Protocol for Information via Negotiation) [KR99] is a family of adaptive protocols, which includes SPIN-1 and SPIN-2. It is the first data-centric routing protocol [AY05], which uses the data negotiation mechanism to eliminate the redundant data transmission. Consequently, data delivery is not guaranteed in SPIN. Another data-centric routing protocol, named Direct Diffusion (DD) [IG03], has been developed using a single path for data transmission. Both SPIN and DD are energy-efficient protocols. Later, based on this single path routing in DD, a highly resilient and energy-efficient multi-path routing scheme has been developed in [GG01]. Then many other protocols have been designed and proposed based on or following the similar concept of DD [MA01][YG02][BE02][SS01][CH02][SR02][SK03].

In this Thesis, we have proposed a new routing protocol for WSN, called *Reliable and Energy Efficient Protocol (REEP)*. REEP is a network layer data-centric routing protocol whose design has been motivated by the existing design features of DD [IG03]. In addition, the energy conservative heuristic of SPIN-2 has motivated us to maintain an energy threshold value in each REEP node in order to make the sensor nodes energy aware. In the SPIN family of protocols [KR99], SPIN-2 works the same way as SPIN-1 when all the nodes are full of energy; but when the energy of a SPIN-2 node approaches a low-energy threshold, it reduces its participation in the protocol.

In Section 2.3.1, we emphasize on why this data-centric approach is energy-efficient, then we provide in Section 2.3.2 a brief description of DD

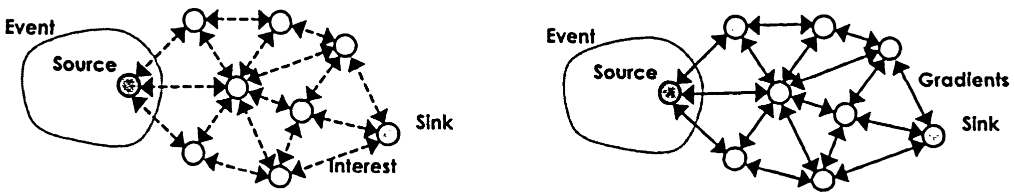
2.3.1 Why The Data-Centric Approach is Energy Efficient

In traditional address-based routing schemes, routes are created between addressable nodes and are managed at the network layer [AY05]. The data-centric routing scheme is different from the traditional routing approach. Here, data are named by their attributes and the names specify the properties of the data (More detail explanation are given in Section 4.3.1). It is not feasible to assign a global identification to each sensor node in the sensor field due to the random deployment of sheer number of nodes in the network. Moreover, associating the Global Positioning System (GPS) with each node is very expensive for any sensor network application that requires a large number of nodes and random topology changes. It is very hard to query on a specific set of sensor nodes without having such global identification. As a result, data redundancy causes more power consumption in the network that is inefficient in terms of energy usage. Consequently, most of the network layer protocols for sensor networks use the data-centric approach [AY05], which helps to eliminate significant data redundancy by using data aggregation, thus saves a significant amount of energy. The protocol proposed in this thesis also follows the same approach.

2.3.2 Directed Diffusion (DD)

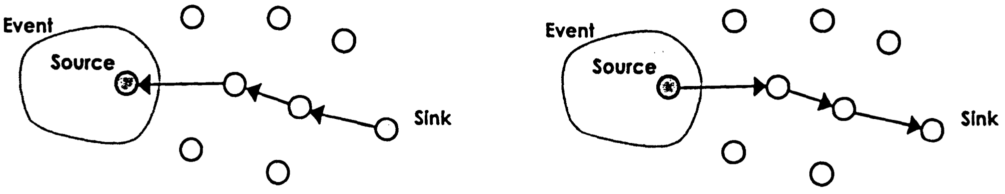
As mentioned earlier, DD uses a data-centric scheme [IG03]. Data generated at sensor nodes are named by attribute-value pairs. All nodes in a directed-diffusion based network are application-aware, meaning that all communications are done based on named-data. Tracking the location of any moving object, such as a vehicle, a human or an animal in a specified region, is an example of the usage of the DD protocol in a simple remote surveillance sensor network. To have a clue about how DD works, let's consider a simple example where a user raises the following interest: "How many pedestrians do you observe in the geographical region X?" or "Is there any four-legged animal in area Y?" or "Tell me in what direction vehicles are moving in area Z?" In such cases, the user addresses his query to the sink node of the network. This node then triggers the collection of information using sensor nodes within that specified region. In this processing, the query is transformed into an *interest*, i.e., actually a task description. In the DD protocol, each task is described by a list of attribute-value pairs [IG03]. The sink node initially and repeatedly diffuses (i.e. broadcasts) the interest as a low-rate *event notification* throughout the network (see Figure 2.1 (a)). The authors in [IG03] referred to these notifications as *exploratory events*, which are useful for the path setup and repair. A two-way gradient establishment is done in every pair of neighboring nodes during this interest propagation phase (see Figure 2.1 (b)). A *gradient* is used that specifies both the data rate requested by a specified neighbor and the direction in which the events shall be sending. These gradients that are setup for exploratory events are referred to in [IG03] as *exploratory gradients* because of their low data rate.

Once a node of a specified region detects any matching target, it reports the exploratory events back to the user by using the reverse path of interest propagation. When a sink node receives the reported exploratory events, it *reinforces* one particular neighbor to draw down the real data (see Figure 2.1 (c)). This *reinforcement event* is the same as the interest event, except that it contains a higher data rate. Gradients that are setup for this event are referred to as *data gradients* because of their high data rate. The sink node reinforces that neighbor from whom it first received the reported event, and the intermediate nodes do the same tasks as the sink. An empirically low delay path is then established from the source node to the sink node for data transmission through this sequence of local interactions. Figure 2.1 (d) shows an example of data transmission through a reinforced path in a simplified data-centric routing paradigm.



(a) Interest Propagation

(b) Gradient establishment



(c) Reinforcement sending for path setup

(d) Data transmission through reinforced path

Figure 2.2: A Brief Explanation of the Directed Diffusion Functionalities

2.4 Summary

In this Chapter, we briefly discussed the different layer protocols for WSN network in the protocol stack. Other than the network layer, protocols from other layers are beyond the scope of this thesis.

Among network layer protocols for WSN, it was found that location-based protocols lead to expensive applications because they use the GPS system in sensor nodes. They may not be feasible for most applications that require small size nodes. It was also found that the first phase of hierarchical protocols consumes more power during complex computation of cluster formation and cluster head selection. These cluster-based protocols require complex algorithms for their first phase' activities. Moreover, they require more powerful nodes as cluster heads in some applications. Finally, in data-centric routing, a Base Station (BS) (or a sink node) sends its query to a specific region. This routing technique requires the introduction of an attribute-based naming scheme for query generation purpose. It was found that such query may return effective results when using a data aggregation technique.

In the next Chapter (Chapter 3), we describe the functional and non-functional requirements of the REEP design. We also discuss how different events are generated and handled by REEP.

Chapter 3

NON-FUNCTIONAL AND FUNCTIONAL REQUIREMENTS OF REEP

Compared to the wired communication, wireless communication is more unreliable. Wireless sensor communication is even more unreliable. This is because unlike traditional nodes, sensor nodes are constrained in terms of source of energy, size, and computational capability. Moreover, there are lots of environmental factors which interrupt such wireless communication. So, the requirement of efficiency and reliability in sensor communication should be the most important concern when designing sensor network protocols. Our REEP protocol design must fulfill these requirements.

The requirements of REEP can be described in two categories: non-functional and functional requirements.

3.1 Non-Functional Requirements of REEP

By *non-functional* requirements of REEP, we mean those requirements which will lead REEP to perform better compared to other protocols of the same family. The non-functional requirements include reliability, efficiency, and the usability of REEP. The most important feature of REEP is that each node of a network maintains an energy threshold value. This threshold value makes REEP'nodes energy-aware. This energy awareness of each node ensures reliability and efficiency. The following sections describe how these non-functional requirements are accomplished in REEP.

3.1.1 Reliability

In a network, when a data packet travels through a dedicated path from source to sink, any node on the path may fail to transmit the packet because of insufficient or no energy remaining in the node. Consequently, in case of a node failure, some data packets may get lost in the failed node. Keeping this problem in mind, we are planning to design REEP in such a way that during path setup and data transmission, the residual energy will be checked with the energy threshold value in each node. As a result, before any node gets totally dysfunctional, it passes over the current data item (to avoid data loss) and informs its neighbors about its status by sending negative events to deny participating in further path setup. This mechanism makes the path more reliable for data transmission, because nodes with adequate energy take part in path setup and data transmission and nodes with energy below the threshold value are only capable of sensing and sending control information to their neighbors. This asserts our definition of "reliability" in the context of our proposed protocol. This particular non-functional requirement is validated through some simulations provided in Chapter 5 (Section 5.6.6).

3.1.2 Energy Efficiency

The use of the data-centric approach for routing, as well as the energy threshold value and a FIFO queue, introduces an efficient use of energy in REEP nodes. The data-centric approach is energy-efficient (see Section 2.3.2). Since every node maintains a FIFO queue for each task in the network, this queue helps to reduce the unnecessary flooding and ensures more energy savings. In case of any path failure, REEP needs not invoke a network wide flooding for alternate path setup. Instead, any node can select its next best neighbor from that queue (i.e., the one with highest priority) in order to request for alternate path setup. Therefore, energy-efficiency can be achieved in REEP. This non-functional requirement of REEP has been validated using some simulation scenarios presented in Chapter 5.

3.1.3 Usability

The design of our proposed protocol allows it to work with different scenarios (described later in Section 4.4), which clarify the "usability" of REEP in our design context. The first phase (*sense event* propagation) of REEP scans the whole network and allows a user to choose a single or multiple tasks from a list of available detected objects within the topology area. Based on this choice, the sink will send a request for real data, where each task is distinguishable by both the detected object type and its source node's location information. This design mechanism makes REEP usable in most scenarios.

The requirement for a user is optional in REEP. Tasks can be chose by either any user or any application according to the requirements of this application. In this work, we are not too concerned about how these tasks are chosen.

3.2 Functional Requirements of REEP

In the context of this thesis, the term "functional requirements" refer to the internal workings of the software artifacts sustaining the design of REEP, i.e., the technical details, the data manipulation and processing, the task description, and the resource discovery. However, since most of these requirements are already encapsulated in the internals of MATLAB 7.4, the software package that we have used for our implementation of REEP (see detail in Chapter 5), we will only define how the task description and resource discovery are handled by REEP.

3.2.1 Task Description

In DD, a sensing task works as an *interest* that contains the description of this sensing task. This *interest* message is a kind of query which indicates what a user would like to know about some resources available in the network. Consider the following sensing task as an example of an *interest* in DD [IG03].

```
type = wheeled vehicle      // detect vehicle location
rect = [-100,100,200,400]   //from sensors within rectangle
interval = 20ms             // send events every 20 ms
duration = 10 seconds       // for the next 10 seconds
```

REEP follows the similar task description rule, but the contents are different. In REEP, a sensing task works as a *sense event* that contains the description of this sensing task. Unlike the *interest* of DD, the *sense event* of REEP is not specific to the object type and the rectangular area. This *sense event* is a kind of general query which scans the whole network. A *sense event* in REEP can be described as:

```
sink =  [120, 230]       // node location
timestamp = 11:20:40     // event generation time
duration = 10 sec        // for the next 10 seconds
```

3.2.2 Resource Discovery

REEP follows some of the resource discovery mechanisms of DD (Section 2.3.2). These are naming scheme, event generation, and event management. There are other resources, such as data processing, path setup processing and neighbor relationships, which we have redesigned in order for REEP to achieve a better performance (Section 4.3.3-4.3.5) compared to DD, for specified scenarios and application types (see simulation results presented in Section 5.4 of Chapter 5)..

3.3 Summary

In this Chapter, we have discussed the non-functional and functional requirements of REEP, as well as how these requirements can be accomplished in the context of REEP.

Next, based on the above requirements the next chapter (Chapter 4) focuses on describing the design steps of REEP.

Chapter 4

DESIGN OF REEP

Our proposed protocol for wireless sensor network has been designed motivated by the design structure of the DD protocol. As such, the data-centric routing technique used in DD has been re-applied in the REEP design context. The purpose of this chapter is to provide a view of the REEP design architecture and processing, which include: the design specifications, the design elements, the design steps, and finally the design usability scenarios.

4.1 Design Specifications

Before describing the design steps, it is necessary to first introduce the design choices of REEP. There are five important design elements (see detail in Section 4.2) that are used in different propagation phases of the REEP protocol. The energy threshold value is one of these elements and the selection of this value varies depending on the requirements of different applications. For the path setup in the REEP protocol, three propagation phases are needed. In addition, any data-centric routing scheme requires the use of a naming scheme. For designing the REEP protocol, we have selected the same naming scheme used in DD (see Section 4.3). Finally, in the REEP design, the use of a human operator is optional or application-dependent.

In order to achieve energy savings using REEP, one of the design constraints is that all the sensor nodes in the sensor network will not sense all the time; rather, they will sense only when they receive new sensing events. Based on the application type, a prescheduled sensing task can be designed for robustness. For example, one event per second. This sensing task is then periodically generated by the sink node with an increased timestamp. This is necessary because sometimes, events are not transmitted throughout the sensor network because of the weak quality of the sensor network links.

4.2 Design Elements

REEP has five important elements. These are: (1) the *sense event*, (2) the *info event,* (3) the *request event,* (4) the *energy threshold value* and (5) the *FIFO queue.* A *sense event* is a kind of query that pushes each node to start sensing. This sense event is generated at the sink node, and is supported by the sensor network for acquiring information. The response of this query is the *info event,* which is generated at the source node. It specifies the detected object type and the location information of the source node. After receiving this information, *request events* are generated at the sink node and are used for path setup to retrieve the real data. Real data in any sensor network is the collected or processed information regarding any physical phenomenon. Each node in REEP uses an *energy threshold value* by checking whether a node agrees or denies participating in any further activities. This will result in a more reliable transmission of any event information or real data. The FIFO queue is used at each node to track the sequence of *info event* received from its different neighbors. Each node selects the first neighbor from that queue to send the request for path setup. The above described elements are for a particular category of sensor network applications, such as applications which require the use of location-tracking tasks for moving objects. Figure 4.1 illustrates a simplified view of how REEP works.

4.3 Design Steps

The steps of the REEP design include the design of a naming scheme and the design of different events propagation. These events propagation represent the workflow of the main functionalities of REEP. They can be divided into three steps as shown in Table 4.1. An explanation of how the events are generated and processed in the REEP protocol will be given later in this section.

Table 4.1: REEP functional steps

Steps	Name
1st	*Sense event* propagation (Section 4.3.3)
2nd	*Info event* propagation (Section 4.3.4)
3rd	*Request event* propagation (Section 4.3.5)

4.3.1 Naming Scheme

Designing a naming scheme is the first step towards designing a diffusion-based routing protocol for sensor network. For the design of REEP, we have chosen the simple *attribute-value* based naming scheme for all tasks, used in the directed-diffusion protocol context. Other choices of naming schemes exists as well [WS99], but in our work, we are using the aforementioned attribute-value based naming scheme -- in which each task includes a list of attribute-value pairs. As an example, a sensing task can be described as follows:

{Attributes} {Values}
↓ ↓

```
sink =  [120, 230]     // sink location that generated this
                       // sense event
timestamp = 11:20:40   // event generation time
duration = 1 sec    // time each node activates its sensing
                    //device when no object is detected
```

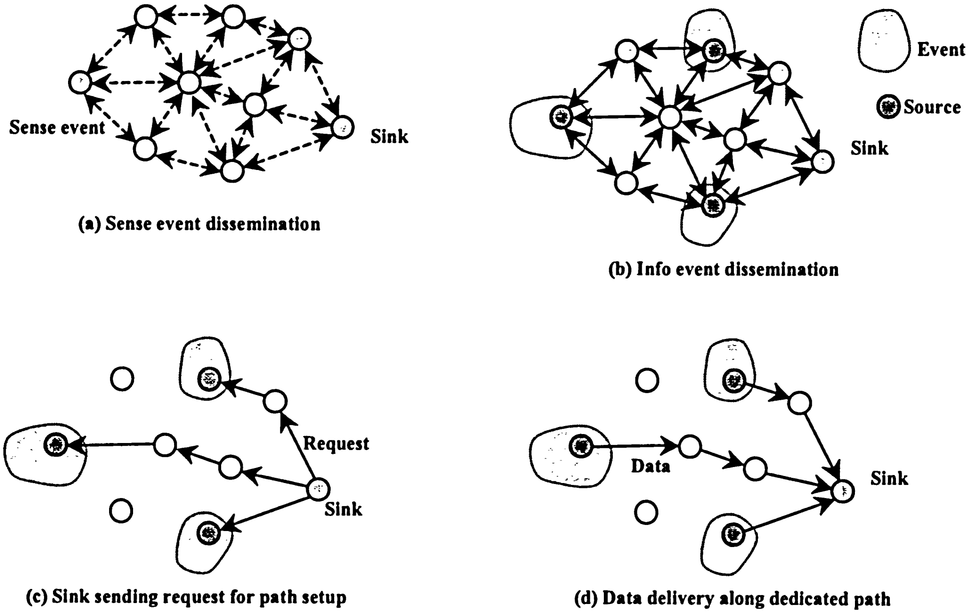
It should be noticed that a node activates its sensor device to sense a physical phenomena if and only if it receives a sense event from different sink nodes or with different timestamps. Thus, if the sensor nodes detect any object, they will generate a report using a similar naming scheme. This report is referred to as an *info event.* Such event can be described as follows:

```
type = human        // detected object type
location = [10,15] // source node location
intensity = 0.84    // signal amplitude measure(useful info
                    // for the application)
confidence =  0.78  // confidence in the match
timestamp = 11:22:30 // event generation time
```

In the above description, each attribute has an associated predefined value range. For example, the domain of the *type* field is the set of code book values representing mobile objects (e.g. vehicles, animals, humans) [IG03]. The value of the attribute can be any element of its domain. For example, the value of the *type* attribute is human. There can be other design choices for the attribute-value domains. In this thesis, we are not concerned about these predefined domain values or any target recognition algorithm in detail because this is beyond the scope of our interest. Briefly said, these algorithms use a library of pre-sampled stored waveforms to match with the collected sample waveforms. Different objects have different acoustic or seismic signals. In general, these algorithms associate a degree of confidence to the match, in order to indicate the relative type of the object. They also associate the intensity of the waveform to the match, in order to indicate the relative distance to the originated signal.

4.3.2 Simplified Schematic of REEP

To have an overall view of how REEP works, let us consider the following simplified form of a WSN application. Let us assume that a large geographic area has a large number of sensor nodes deployed for security purpose. As shown in Figure 4.1 (a), let us assume that a user "injects" queries into the sink node through the task management node. An example of such query is: "What kind of moving objects can be found within the entire secured zone?" This query leads all the sensor nodes to sense for a limited time to collect information from the environment. Nodes that sense any moving object, such as a human or a vehicle or an animal, send this information (not the real data) back to the user. If multiple sources detect objects from different locations (see Figure 4.1 (b)), the user receives a list of detected objects and the corresponding detector source node's location information. The user can, then, choose one or multiple sources from the list for having the detailed data of the detected object. The user then sends requests through the sink node to retrieve the real data. The sink node forwards these requests to the particular neighbors in order to setup the paths for the real data transmission (see Figure 4.1 (c)). Intermediate nodes do the same tasks as the sink node does for path setup. Once the sources receive their corresponding requests for real data, they complete the path and start to send the data along the dedicated paths (see Figure 4.1 (d)). Thus, the user starts receiving the real data through a single path for each task.



(a) Sense event dissemination

(b) Info event dissemination

(c) Sink sending request for path setup

(d) Data delivery along dedicated path

Figure 4.1: A Brief explanation of the REEP functionalities

4.3.3 Sense Event Propagation

A sensing task is initiated at a sink node in the network. A sink node can be any node within the network topology, from which the user wants to acquire all the information s/he needs. Now, let's describe how this sense event is diffused throughout the sensor network. As an example, the following sense event is instantiated in a particular sink node, with the corresponding values:

```
sink =  [120, 230]
timestamp = 11:20:40
duration = 1 sec
```

This is an exploratory event which is flooded all over the network, and which is intended for path setup and repair. Here, the value [120, 230] of the *sink* attribute indicates the location of the sink node that generated this sense event. The value of the *timestamp* attribute specifies at what time the event was generated and the *duration* attribute specifies the time each node will keep activating its sensing device in case no object is found. This value is used as an indicator of the energy saving and to increase the node's lifetime.

Let us consider the example of sense event propagation shown in Figure 4.2. In this example, Node A is the sink node which generates the sense event. Every node maintains a cache to store all necessary information for the routing of data. Node A stores all the necessary information from this event and broadcasts this sense event to all of its neighbors just to activate their sensing devices to sense environmental phenomenon around them. Every time a node receives a sense event, it checks only the sink and timestamp info of the received event with the saved sink and timestamp info of the node's cache. It should be noted that each node always overwrites the latest sense event info and is not required to store a list of previous sense events, because this event is used only for sensing purpose, not for routing. When node B receives the sense event from Node A for the first time, it checks its cache and finds that this is a new sense event. Then, Node B saves the sink and timestamp info from that event and sends the event to Nodes C, D, E and A. Node B also activates its sensing device to collect samples within its sensing range. Nodes C, D and E repeat the same processing as the intermediate Node B did. If any node receives the same sense event from its neighbor (which means that the sense event has already been sensed by that neighbor), then it simply discard the event after checking its cache. Here, the same event refers to the event that was generated by the same sink node with the same timestamp. For example, when Node D sends this event to all of its neighbor nodes B, C and E, these nodes receive a copy of the same event from D again, and then drop it.

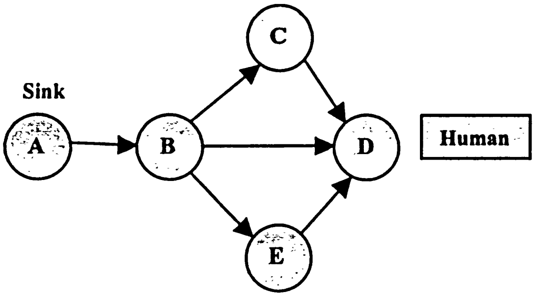


Figure 4.2: Propagation of *sense event*

Depending on the requirements of different applications, the sink node will periodically broadcast this sense event with an increasing timestamp. The sink node will first overwrite the information of the sink location (sink that generates the sense event) and the timestamp (event generation time) in its cache. Then, the sink will activate its sensing device if the received sense event has been generated by some other sink node in the network. The sink node will also send this new sense event to all its neighbors.

From the above example, it is observed that every time a node receives a sense event, it will start to sense again if that event comes from the same sink node with a different timestamp or from a different sink node. These new sense event generations occur based on the sensing schedule or on how frequent a user pose a new query into a sink node, when needed. For example, in the case any information is lost at a node.

The node that detects any moving object within the duration period indicated in the sense event generates the *info* event and sends it to all its neighbors.

4.3.4 Info Event Propagation

A node that detects any object in wireless sensor network is termed as the *source node.* During the sensing period, a source node collects samples and matches them with predefined library values to generate an *info* event. Let's consider the following example of an *info event* generated by an arbitrary source node:

```
type = human
location = [10, 15]
intensity = 0.84
timestamp = 11:22:30
```

Source node includes the value of the object `type` that it has detected and the node's `location` information into the *info* event. These are used to indicate which source node has detected which object. The `intensity` of the sampled waveform roughly indicates the distance between the source node and the detected object. This value helps the user to choose the best source node in case multiple sources detect the same object with different intensity values. The `timestamp` field specifies the time when this info event was generated.

Info event propagation is similar to the sense event propagation, except that during sense event propagation, no entry is created, only the sink and timestamp info are updated each time by overwriting them. But, during the info event propagation, different entries are created for different tasks at each node.

The source node saves all the information from the generated info event by creating an entry in its cache, and then sends this event to all its neighbors. In each node, different entries are created for different tasks based on two attribute values, referred to as the `type` of the object and the `location` of the source node. In our example, the two entries are distinct if the detected object `type` is different or the `location` info of its detector source node is different. At every time, the above processing works when the aforementioned two attribute values match for each entry.

Each entry in the cache has several attribute fields. The 'Type' and 'Source' fields contain respectively the `type` of the object and the `location` of the source node that detected this object. The 'InfoTime' field contains the info event generation time obtained from the `timestamp` attribute in the info event. The 'FIFOqueue' field maintains a FIFO queue. Each time a node receives an info event, it saves the sender node's ID in this queue for the corresponding matching entry. This value is needed later on when the sink node sends a request event to ask for a path setup. An intermediate node chooses the first node from this queue to send the request event.

In Figure 4.3, the info event propagation is illustrated. Node D sends the info event to all its neighbor nodes B, C and E. Node B receives this new info event for the first time from node D, creates a new entry, saves all the necessary information from the event for future routing, and sends it to all of its neighbors C, D, E and A. Node B saves the node's ID of Node D in its 'FIFOqueue'. If Node B receives a copy of the same info event from its neighbor, it just searches for the matching entry and saves the sender's node ID into its 'FIFOqueue'. Here, two matching (based on type and location) info events are distinct if the info event generation time differs between them, which implies that both events were generated with different timestamps.

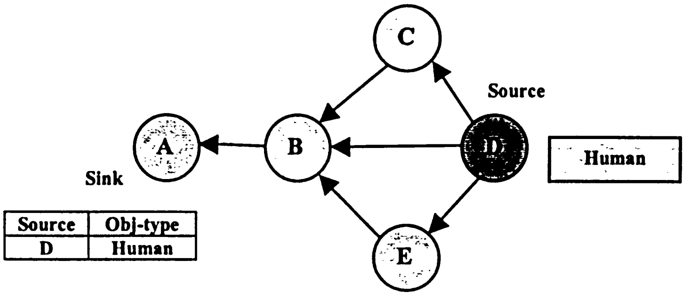


Figure 4.3: Propagation of *info event*

If a node receives a new info event with different event generation times and the node finds a matching entry for that event, which was for some previous task, it does not create a new entry for this new event. Rather, this node simply resets all the values with the updated information, including the 'FIFOqueue' for that corresponding entry. A node does not keep all the previous records for all entries because it leads to more complex searching that requires more energy.

During the info event propagation, 'FIFOqueue' is created based on the sequence in which a node receives the same info event from all its neighbors. This implies that the first node of this queue has the highest priority to be selected for path setup. When the sink node receives the same info event from its other neighbors, it does not request all its neighbors at the same time. Rather, it saves the neighbor' IDs in that queue for future use, in case an alternate path setup is required. Intermediate nodes do the same processing as the sink node. In Figure 4.3, since Node B receives the info event from Node D first, and saves it into its FIFO queue, Node B will send the event request received from Node A to Node D only. When Node B receives that same info event from Nodes C and E, it will subsequently save their node's IDs in its 'FIFOqueue' for future use purpose.

4.3.5 Request Event Propagation

One of the important features of REEP is the maintenance of an energy threshold value at each node. This threshold value is checked at each time, in each node, only when a node receives a request event or a real data to be transmitted. The design of this energy threshold value is application dependent. When a node's energy goes below the prescribed threshold value, this feature allows each node to avoid being involved in the path setup and data transmission, and to transmit control information only. Nodes with less amount of energy can be replaced before they become dead.

In some situations such as node or link failure, nodes need to search for an alternative path setup. In such situations, REEP nodes do not need to perform network wide search or any periodic flooding to find out the next best path to be used for data transmission, or REEP nodes do not need to maintain any alternate multi-paths [GG01]. Rather, they can choose the next best neighbor from their queues for alternate path setup. Therefore, the entire request event is saved in each node for future use.

As explained earlier, when all info events are received at the sink node, the user receives a list containing the information regarding the detected object type and associated source node location information (see Figure 4.3). The user can choose a single or multiple tasks to retrieve the real data. Based on the user's selection, the sink node generates request events for path setup purpose.

Once a sink node receives the info event for the first time, it does the same processing as an intermediate node does. In addition, the sink node creates a request event and sends it to that specific node from which it has received the corresponding info event first. To do this, the sink node simply chooses the first neighbor from its 'FIFOqueue' to send the request event for path setup.

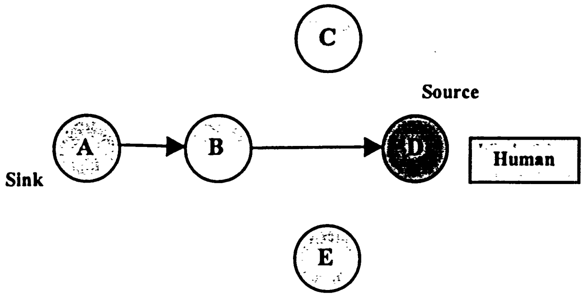


Figure 4.4: Propagation of *request event*

An intermediate node does not have the entire path information for a specific task. All it has is the knowledge about its neighbors from/to whom it receives/sends events. So, when an intermediate node, such as Node B in Figure 4.4, receives a request event from Node A, it searches for any matching entry and picks the first neighbor from the FIFO queue of that entry to send this request event. Each node also saves the ID of the neighbor from whom it has received and accepted the request for data transmission corresponding to the matching entry.

When a source node, for example Node D in Figure 4.4, receives a request event, it completes the path and starts sending the real data at a higher rate along the path corresponding to the specified request. After completing this sequence of local interactions, a path is constructed for data transmission from the source node to the sink node.

4.4 Usability of the REEP Protocol

The way REEP protocol has been designed allows it to work in different scenarios. Such scenarios are, for instance, when a single source detects single/multiple object(s) or when multiple sources detect single/multiple object(s). These scenarios are described below.

4.4.1 Single Source detects Single Object

When only a single source node in the whole sensor network detects a single object, it creates an entry for that object and floods this information throughout the network. If the user posts the request for this object, then, the sink node will send a request for path setup. Once the path has been established, the source node will start sending the real data through this path. This scenario is illustrated in Figure 4.5.

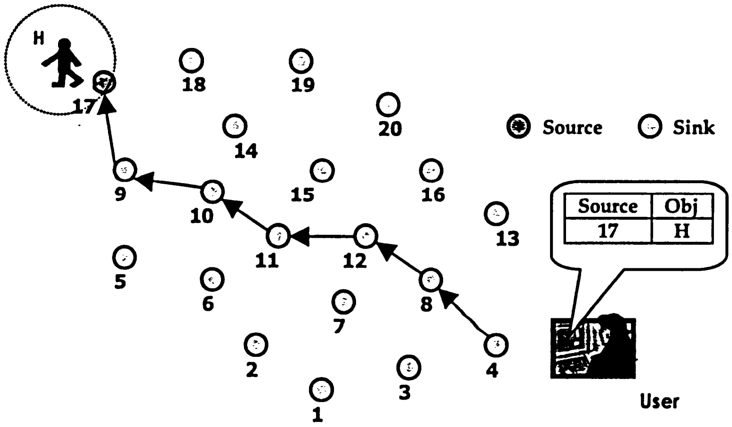


Figure 4.5: Single Source detects Single Object

4.4.2 Single Source detects Multiple Objects

If only one source node in the whole sensor network detects multiple types of objects around it, the source node then creates multiple entries for each of these objects, and floods that information throughout the network. Once the sink node receives all the information, it sends them to the user. The user sees a list consisting of different detected objects, but the source node location is the same for all objects. The user then chooses a single or multiple objects for obtaining the detailed information. Based on the user's request, the source node sends the data along the dedicated path for the specified object type. This scenario is illustrated in Figure 4.6.

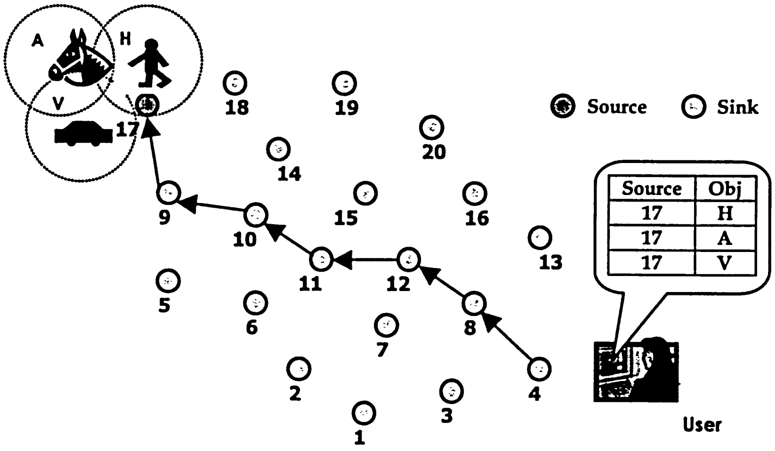
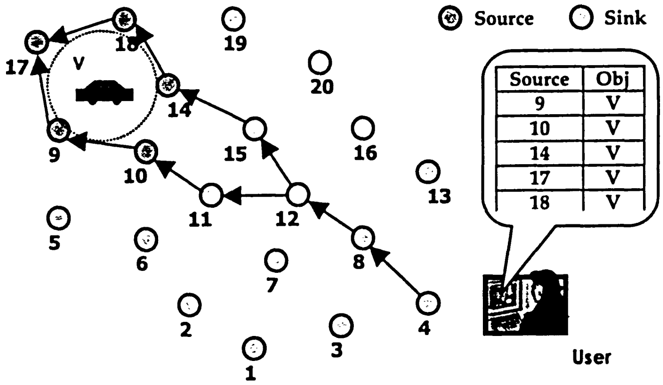


Figure 4.6: Single Source detects Multiple Objects

4.4.3 Multiple Sources detect Single Object

This is a quite common scenario in wireless communication, in which a couple of sensor nodes detect the same object (see Figure 4.7). This happens when an object belongs to the area where sensing occurs over the overlapping regions of multiple nodes. In such a situation, different source nodes create different entries for the same object and flood this information. When that information is exchanged among multiple source nodes, they create individual entries for the same object with different source information. This instance creates a kind of data redundancy in the nodes cache, but does not affect the overall performance of the network. The reason is that when the user observes that a list showing multiple sources have detected the same object, and the location of those source nodes belong to a small area, the user can quickly realize that the object is a single object, and not multiple objects of the same type. S/he can then choose any one of the source nodes to request the data based on the additional intensity information of each task. Once the source node receives this request, it starts sending the data along the path towards the sink node. Figure 4.7 illustrates this scenario.

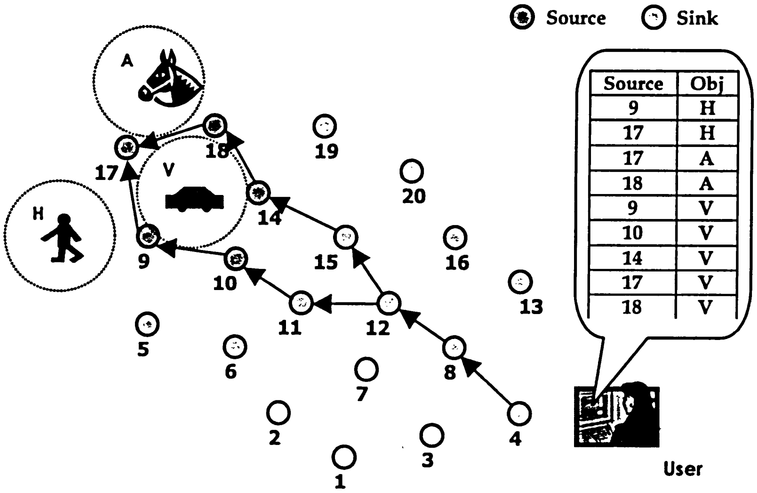


Source	Obj
9	V
10	V
14	V
17	V
18	V

Figure 4.7: Multiple Sources detect Single Object

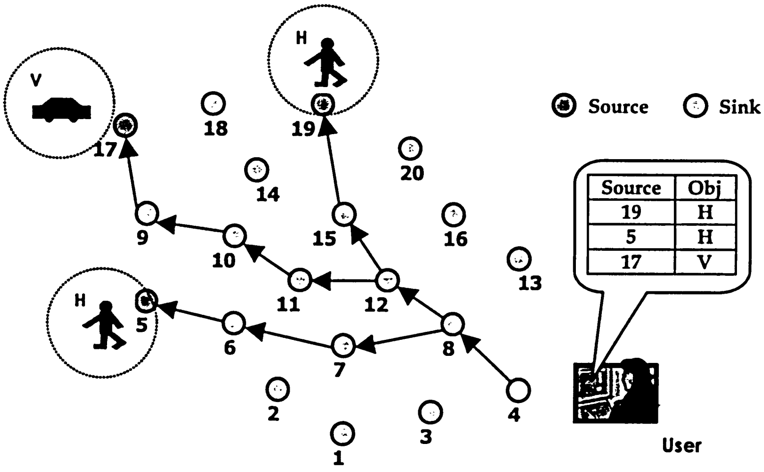
4.4.4 Multiple Sources detect Multiple Objects

This scenario can have two different aspects. First, multiple sources of a small geographic area detects the same multiple objects within that area (see Figure 4.8 (a)). Second, multiple sources from different areas detect different multiple objects within their vicinity (see Figure 4.8 (b)). Both situations result in a list of detected objects associated with their source node's location information. In the first case, the user can choose the appropriate source nodes (based on the `intensity` value) and in the second case, the user can choose any source node for any object. Paths are then constructed according to the user's request and the source nodes then send the data along these paths as shown in Figure 4.8.



Source	Obj
9	H
17	H
17	A
18	A
9	V
10	V
14	V
17	V
18	V

(a) Same set of Multiple Objects



Source	Obj
19	H
5	H
17	V

(b) Different Multiple Objects

Figure 4.8: Multiple Sources detect Multiple Objects

4.4.5 Selection of a Specific Object by a User

In REEP, besides the object type and the source node's location information parameters contained in the *info event*, there are some additional parameters that can be used to help the user select a specific object from the list of available objects for real data retrieval purpose. These parameters are the *Intensity* and *Confidence* of the detected objects (See Table 5.4). Consider the situation (Figure 4.7) where a user cannot recognize an object within the list of available objects. In such a case, the location information of each detected object could be introduced as an additional parameter within the *info event*. This is classified as part of future works.

4.5 Fault Tolerance in REEP

Wireless sensor communication is likely to be vulnerable to node or link failure because of limited and irreplaceable source of energy of sensor nodes. This type of failure causes disconnected path and breaks the communication among sensor nodes. There are some other types of failures that can be experienced by unexpected environmental factors on the communication path. Examples of path failures include thunder storm, strong wind blow or any obstacle on the path. Repair or reconstruction of the failed path is needed to resolve the problem.

Network protocols are designed in such a way that they deal with these types of failure and keep the communication to continue among sensor nodes in spite of the adversities. This issue is known as *fault tolerance* in networking. REEP has been designed while keeping these issue in mind. The following section elaborates on the fault tolerance issue and how REEP handles it.

4.5.1 Repair of Failed Path using Negative Events and FIFO queue

As mentioned earlier, in the REEP protocol, each node maintains an energy threshold value and paths are constructed after checking the energy level of each node with this threshold value. This implies that, paths in REEP protocol are more reliable for data transmission. But still it is expected that after a certain period of time, nodes on the path can fail because of excessive data transmission required for performing several tasks. Also, node failure causes path failure.

In REEP, each time after receiving real data, a node checks its remaining energy level and compares it with the threshold value. Now, let us consider a situation (refer to Node B in Figure 4.9) where a node's energy level was above the threshold value before receiving the data, but goes down the threshold value right after receiving the data. This node finds low energy in it. Since each node maintains an energy threshold value, low energy does not mean that a node is dead and not capable of doing any processing. A node's energy level below the threshold value simply means that the node is capable of doing limited processing.

Node B (see Figure 4.9) then picks up the corresponding request event from its cache, which was saved during request propagation step, changes this event's status to negative and sends this negative request event towards the neighbor Node A on the path from whom it received the data packet. In addition, Node B also changes the data packet's status to negative and sends this negative data packet to the next Node C on the path (see Figure 4.9). Consequently, data do not get lost in the middle of the path for having low energy in a node. Thus, this low energy Node B lets its both-side neighbors A and C to know about its own status that it has low energy and capable of sensing and exchanging control information only, not for path setup or data transmission.

It should be recalled from the request event propagation step that each node also saves the neighbor's information from which it accepted the request for path setup from the sink node to the source node. When a node receives data, it picks up that neighbor's information from its cache and sends the data towards that neighbor.

Node A receives and forwards the negative request (Figure 4.9 (a)) to node F towards the source node in order to close the path. This node does not send any more data to the neighbor node B. Only exploratory events are exchanged between them.

On the other side, Node C receives the negative data packet (Figure 4.9 (a)), changes the packet's status to positive and sends it to the next node D on the path. Additionally, Node C also removes the low energy node B's ID from its 'FIFOqueue' and picks up the next Node E from the queue to request and sends the previously saved request event to Node E for alternate path setup (Figure 4.9 (a)). Node E sends its best neighbor G and so on. Thus, an empirically alternate path is established.

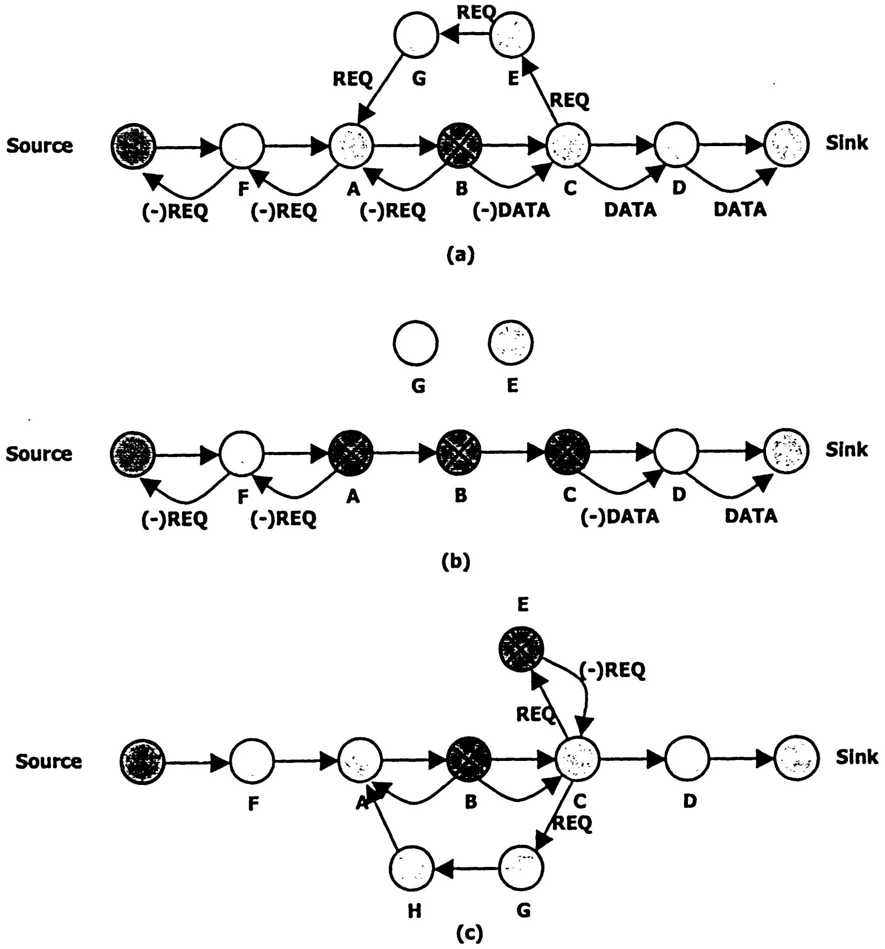


Figure 4.9: Node failure and Alternate Path Construction

Now, what happens if either Node A or Node C finds that its energy is low when it receives the negative event ((-)REQ or (-)DATA) from node B? They simply pass the negative events to their corresponding neighbor nodes without changing the status of the event (see Figure 4.9 (b)). Node A sends the negative event to Node F and Node C sends it to Node D.

What happens if Node E finds that its energy level goes below the threshold value as soon as it receives request from Node C (see Figure 4.9 (c))? Node E changes the status of the received request event from positive to negative and sends back to Node C indicating that Node E is not interested in path setup because of low energy. Node C does the same as before, removes this Node E's ID from 'FIFOqueue' and picks up the next node from the queue to send a request for alternate path setup (see Figure 4.9 (c)).

4.6 Advantages and Disadvantages of REEP

Based on the study of the REEP design carried in this chapter, we have identified some advantages and disadvantages of the REEP protocol.

4.6.1 Advantages

Some of the important features of the REEP protocol are the introduction of the *FIFO queue* and the *energy threshold value*. These features give a "*good shape*" to the REEP protocol and provide many advantages. The energy checking mechanism keeps control on the residual energy at each node, resulting to a completed path in the network which is reliable for data transmission. Additionally, maintaining the energy threshold value at each node allows that node to transmit the received data to the next node on the path in case any node finds low energy in it. Therefore, data loss is very low in REEP.

To handle alternate path setup in case of node or path failure in the network, REEP does not need to start flooding low rate events or to maintain proactively constructed alternate paths in advance, which cause more computation and power consumption in the network. In such a situation, nodes in REEP simply pick the next best neighbor from its FIFO queue to request for alternate path setup without causing any flooding.

REEP works well with multiple objects as well as with multiple sink and source nodes. The reason for this is that each task in REEP is distinguishable based on both the detected object type and its detector source node's location information.

In case of random objects moving towards a direction, REEP provides some information regarding this direction. How does it do proceed? Based on the application requirements, a prescheduled sensing task can be designed for REEP, in which for example, every second, the sink will broadcast a sense event throughout the network to scan all available moving objects within the topology area. When an object moves out of the sensing range of one source node and gets into the sensing range of another source node, the former source node has no more recent information about that object. The latter source generates some info events about the detected object and sends them to all nodes. A user can then have an idea about the direction of the moving object by observing the changes in the source nodes location information for a particular object type.

When a user needs to send some query into the network, s/he does not need to have an earlier knowledge of available objects and the area where they could be found. S/he can get a general idea of all detected objects and their relative location information by sending a sense event to the sink node to scan the whole network area.

In case of multiple objects or source nodes, REEP does not do all the works at once. Instead, it gives the option to the user to choose a task after completion of the info event propagation. The remaining activities, such as path setup and data transmission, are completed based on the user's choice. User chooses only those tasks from a list of tasks, s/he needs to know. This feature helps avoiding additional/unnecessary processing in the network.

4.6.2 Disadvantages

All data-centric protocols including REEP are most likely to be application-specific and their sensor nodes are subject to be stationary. This restriction imposes some limitations on the usage of this type of protocols. Therefore, REEP is not suitable for applications where dynamic changes of the network topology need to be propagated across the network.

In REEP, if the prescheduled sensing task or sensing task initiated by a user is very frequent, then there are higher chances for more node failure and data loss in the network because of the excessive energy consumption in the entire network. Without an appropriate design of prescheduled sensing tasks, REEP can perform poorly.

4.7 Summary

The REEP protocol has been designed by taking some important points into account. First, in diffusion-based networking, unlike the end-to-end communication in traditional networking, all the communication is neighbor-to-neighbor. Second, there are no specifically-designated routers to route the events in sensor networks. Each sensor node acts as a 'router' and an 'end' point for communication. Third, nodes in the sensor network take routing decisions based on the local information only, because each node can only have the knowledge about their neighbors within their vicinity. Nodes cannot have the global information about the whole network because they are not uniquely identified by global addresses. Fourth, wireless communication is basically accomplished by broadcasting where there is no acknowledgement system like in traditional networking. So, nodes cannot get their neighbor's current status. Finally, sensor nodes are capable of processing and aggregating different events, as well as performing distributed sensing.

Our next chapter (Chapter 5) discusses the evaluation and implementation of REEP.

Chapter 5

EVALUATION AND VALIDATION OF DESIGN

In this chapter, the main functional and non-functional requirements of our proposed protocol are validated. First, we provide qualitative comparisons of REEP and DD and then, we provide quantitative comparisons of both protocols in terms of various metrics including: computational complexity, average packet transmission, average dissipated energy, average delay, and average data loss ratio. Performance analysis is done both in grid and random arrangements of the sensor nodes in the sensor network.

The sample network topology that we have used in our simulations is composed of a large geographical area, occupied by a large number of sensor nodes scattered in a random fashion, and capable of performing cooperative tasks. The example application is such that, a whole sensor field needs to be scanned for available objects, because a user may not have a prior knowledge of all the available objects in the network. Then, based on the results of the query, the user can select any object or it's detector source node to retrieve the real data.

5.1 Qualitative Comparison of DD and REEP

The qualitative comparison of REEP and DD are based on some design steps features of these protocols, and their functionalities. The construction structure is similar in both REEP and DD; but the concepts behind them are different. The similarities and differences between these two protocols make the basis for our comparisons.

5.1.1 Similarities

The main similarities between DD and REEP protocols are described below.

- Both the DD and REEP protocols are diffusion-based, which means that the control information is diffused throughout the whole network for path setup before the real data transmission.
- Application-aware.
- Data-centric based, i.e. in such protocols, data are named by their attributes.
- Both use the same naming scheme.
- They work mainly in three phases for path setup, as shown in Table 5.1.
- Single path is established for data transmission based on the local interactions in both protocols.
- Both of them use negative events (termed as *negative reinforcement* in DD and *negative request* in REEP) in case of path failure.

Table 5.1: Names of the functional steps

Functional steps of DD	Functional steps of REEP
Interest propagation [exploratory events]	*Sense* propagation [exploratory events]
Data message (reply of the interest) propagation [exploratory events]	*Info* propagation [exploratory events]
Reinforcement propagation for path setup	*Request* propagation for path setup

5.1.2 Differences

REEP defers from DD based on the following features:

- REEP is an interactive protocol in the sense that paths are created after selecting a task. REEP is also energy efficient because in case a large number of source nodes have reported several events, the sink node does not send requests for all events. Instead, requests are sent only for those events that are requested by the application.

- The gradient setup technique has not been followed in REEP because this gradient setup causes more power consumption by exchanging control information multiple times (see details in Section 5.1.3).

- The FIFO queue of REEP ensures time and energy savings for alternate path discovery in case of nodes failure, which avoids invoking a periodic flooding of low rate exploratory events.

- In REEP, the maintenance of an energy threshold value at each node ensures more reliable path for data transmission. If a node's energy goes below the threshold value during the data transmission, the current received data do not get lost in it. Rather, the node sends the data to the next node in the path and denies transmitting any more data by sending *negative events*. A good choice of threshold value causes less data loss in the highly busy network.

- Each task is distinct in REEP based on both the object type and the source node's location information.

Figure 5.1 shows how the paths for data transmission are established using a simple example, where node 1 is the source and node 25 is the sink node. The whole network area is used as the rectangular area in DD and a single object detection is performed. A description of how DD and REEP works is also provided in Table 5.2, using this same example.

Table 5.2: Brief description of the DD and REEP protocols

DD	REEP
Sink (Node 25 in Figure 5.1) is flooding *interest* throughout the network.	Sink (Node 25 in Figure 5.1) is flooding *sense event* throughout the network.
Source (Node 1 in Figure 5.1) is sending back the *reply* throughout the network	Source (Node 1 in Figure 5.1) is sending back the *info event* throughout the network
Sink *reinforces* an intermediate node (Node 20), from which sink received the *reply* first, for path setup. All intermediate nodes follow the same rule for reinforcement.	Sink sends a *request event* to intermediate node (Node 19), from which sink received the *info* first, for path setup. All intermediate nodes follow the same rule for sending the *request event.*
A path has been established from sink to source (25→20→14→8→7→1) for data transmission	A path has been established from sink to source (25→19→14→13→7→6→1) for data transmission
If any node (Node 14) fails on the path, the next node (Node 20) is expecting the data but has not received it for a certain period of time or in a certain rate, then the node (Node 20) sends negative reinforce towards the failed node and reinforces another neighbor (Node 19). Node 20 invokes flooding to find out the next best neighbor to create an alternate path.	If any node (Node 7) experiences low energy on the path, it sends the data to the next node (Node 13) with negative status. Also it sends the request event to the previous node (Node 6) with negative status. Node 13 selects the next best neighbor (Node 12) from its FIFO queue to create an alternate path.
An alternate path has been established from source to sink (25→20→19→18→12→7→1) for data transmission.	An alternate path has been established from source to sink (25→19→14→13→12→6→1) for data transmission.

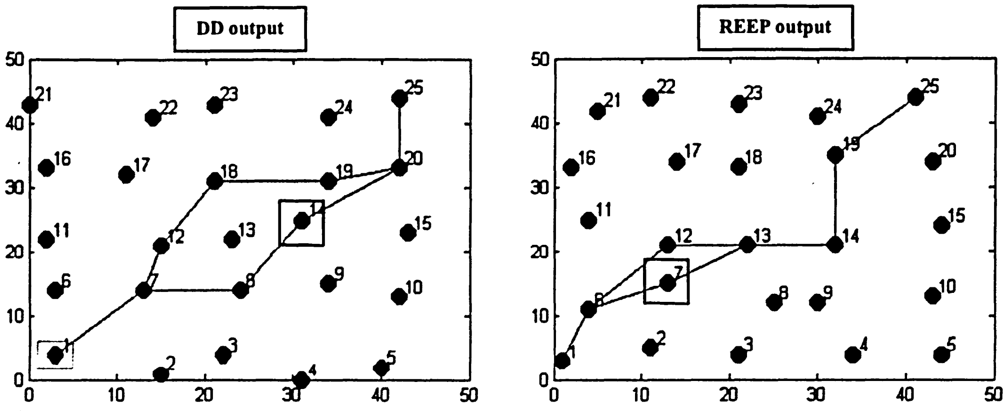
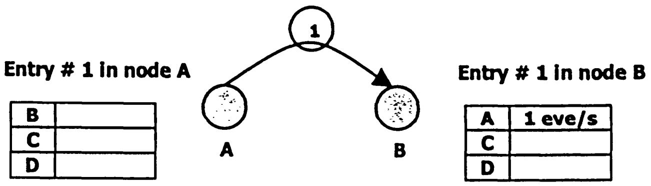


Figure 5.1: A simple example of DD and REEP

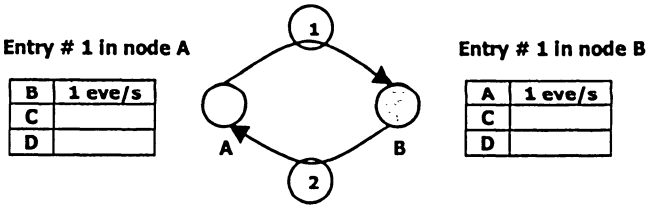
5.1.3 Comparison of Propagation Phases between REEP and DD

Compared to the propagation phases of DD, we have designed the propagation phases of REEP with some modifications. Unlike DD, the gradient setup technique has not been followed in REEP in the first *sense event* propagation phase. This reduces the flooding of information. In the *info event* propagation phase of REEP, all information is saved before broadcasting the event. The *request event* propagation phase maintains the energy consumption by each node. The We difference between the propagation phase as used in REEP and the propagation phase as used in DD is highlighted below.

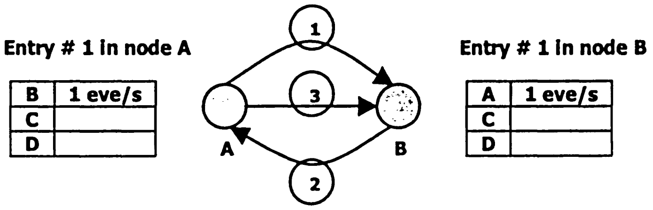
During the *interest* propagation phase (the first step) in DD, two ways gradient establishments are done between each pair of neighbor nodes. It takes 3 times exchanging of each new *interest* between the two neighbors (please refer to Figure 5.2) to accomplish these two ways gradient setup. Figure 5.2 shows that Node A does not set up a gradient before sending the *interest* to Node B (Figure 5.2 (a)) but only after receiving that *interest* from node B (Figure 5.2 (b)). When Node B receives the same *interest* again from Node A, it checks for the gradient setup done before for Node A and then stops sending that interest to Node A (Figure 5.2 (c)). Thus, an *interest* entry in each node contains several gradient fields, up to one per neighbor.



(a) Step 1



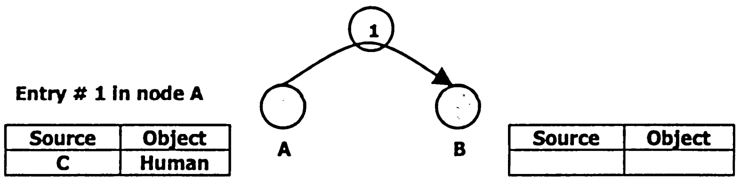
(b) Step 2



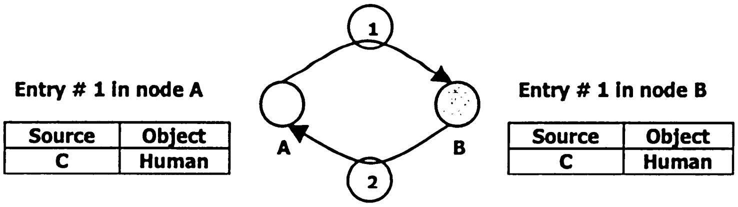
(c) Step 3

Figure 5.2: Two-way gradient setup in Directed Diffusion

In contrast, the first step in REEP (i.e. the *sense event* propagation) is very "light" in the sense that during this step, a node does not do any entry or gradient establishment in its cache. The only task a node does is that it starts sensing if and only if it receives a new *sense event*. In the next step of REEP (i.e. the *info event* propagation phase), a node that receives a new *info event,* saves all the necessary information first in its cache and then sends it to all its neighbors. Node A in Figure 5.3 (a) sends the *info event* to Node B after saving all the necessary information from this event and Node B does the same as Node A (Figure 5.3 (b)).



(a) Step 1



(b) Step 2

Figure 5.3: All the necessary information are saved before sending the *info event*

This feature leads REEP to have better performance in terms of total packet transmission and energy savings for each task. Detailed performance results justifying this claim are shown in Section 5.6 of this Chapter.

In DD, the source node's information is not associated with an event. In some cases, for example when two source nodes detect the same object, the sink might attempt to draw down same data from both sources. This situation causes data redundancy and energy inefficiency. In REEP, all the events are associated with their corresponding source node's location information. Therefore, at the sink node, every event is distinguishable by its object type and source node's location information.

5.1.4 Complexity Computation

Let us consider an example of a sensor network with square grid arrangement where the number of nodes in the network is N. We assume that the transmission range of the sensor nodes is such that each node can communicate with exactly eight neighbors surrounding it. We have assumed a sensor network with grid arrangement in topology for the purpose of simplifying the complexity computation. The complexity of a sensor network with grid arrangement also serves as an upper limit for the complexity of a sensor network with random arrangement. Because the number of links in a grid system is highest and fixed, and will always be the upper bound for both grid and random arrangements of same number of nodes in the network. In random arrangement, we have placed the sensor nodes not in a complete random order; rather, in a pseudo-random fashion (see Section 5.4.2). This way, some nodes lose connectivity with some of their neighbors and gain connectivity with few other neighbors, but the number of lost connectivity is higher than the number of gained connectivity. This phenomenon can be justified by moving one intermediate node in the sensor network as shown in Figure 5.4.

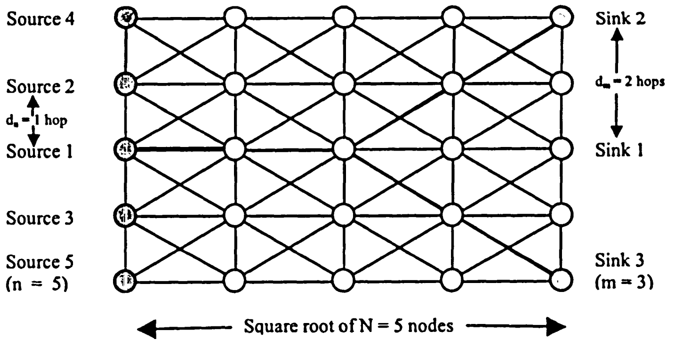


Figure 5.4: Square grid topology for complexity calculation, redrawn from [IG03]

Let us assume that we have 5 sources and 3 sinks. The number of sources is represented by n and the number of sinks is represented by m (i.e. n = 5, m = 3) (see Figure 5.4). All 'n' sources are placed vertically on the left edge and all 'm' sinks are placed vertically on the right edge of the grid. The first source is placed at the centre of the left border. The i^{th} source is $d_n \left\lfloor \frac{i}{2} \right\rfloor$ hops above the first source (if i is even) or below the first source (if i is odd). Sinks are placed using the same placement scheme, except that the distance between 2 adjacent sinks is d_m hops. In our measurement of costs, we assume that the total cost is the cost of transmission and reception of one event from each source to all the sinks. The cost is defined as one unit for the message transmission and one unit for the message reception [IG03].

Cost Calculation in DD

In DD, each source transmits its events along a shortest-path tree to all sinks. Here, the data delivery cost is determined based on the number of links on its source-specific shortest-path trees. There might be several shortest paths for each source-sink pair. The shortest path is determined using the deterministic rule in [IG03]. While building the path from the sink node to source node, we always prefer a diagonal link as the next hop as long as it leads to a shortest path. Otherwise, a horizontal link is selected [IG03]. We follow this rule repeatedly until we reach the source node. Thus, we end up with a shortest path that includes no vertical links. For example, if a shortest path tree rooted at source j is denoted by T_j, then the number of links on T_1 will have two components [IG03]: the number of horizontal links ($\sqrt{N} - 1$) and the number of diagonal links ($d_m \left\lfloor \frac{m}{2} \right\rfloor (\left\lfloor \frac{m}{2} \right\rfloor + 1) - d_m \left\lfloor \frac{m}{2} \right\rfloor ((m-1) \bmod 2)$). Other choices could result in a different cost because the number of shared links on the tree could be different.

The data delivery cost C_d of DD is equal to twice the number of links in the union of all shortest path trees rooted at the source. Here, $C(T_j)$ denotes the cost to transmit an event from source j. This cost can be expressed in terms of T_1 as follows [IG03]:

$$C(T_j) = C(T_1) + C(T_j - T_1) - C(T_1 - T_j)$$

In this expression, $C(T_j - T_k)$ is interpreted as the cost of transmission and reception along the tree formed by removing (from T_j) those links that are common to T_j and T_k. For the ease of computation, $C(T_j)$ can be expressed as the sum of two costs [IG03]: the cost of transmission and reception along the horizontal links $H(T_j)$, and the analogous cost along the diagonal links $D(T_j)$. Hence,

$$C_d = C(UT_{1 \to n})$$

$$= C(T_1) + \sum_{j=2}^{n} \left\{ H(T_j - UT_{1 \to (j-1)}) + D(T_j - UT_{1 \to (j-1)}) \right\} \qquad \text{[IG03] (1)}$$

where

$$H(T_j - UT_{1 \to (j-1)}) = H(T_j) \qquad \text{[IG03] (2)}$$

and

$$D(T_j - UT_{1\to(j-1)}) = 2\left\{ \left\lceil \frac{m + j \bmod 2}{2} \right\rceil d_n + \sum_{l=1}^{\min\left(\left\lfloor \left\lfloor \frac{j}{2} \right\rfloor \frac{d_n}{d_m} \right\rfloor, \left\lfloor \frac{m-(j \bmod 2)}{2} \right\rfloor\right)} \min(d_n, d_n \left\lfloor \frac{j}{2} \right\rfloor - l d_m) \right\}$$

[IG03] (3)

Asymptotically, C_d, the data delivery cost of DD is obtained as:
$O(n\sqrt{N})$ for $m << \sqrt{N}$ [IG03]

Cost Calculation in REEP

In REEP, each source transmits its events along a shortest-path tree to all sinks and follows the same path construction rule as in DD. The simulation output shows this phenomenon. Therefore, Cr the data delivery cost of REEP is the same as that of DD, i.e, $O(n\sqrt{N})$ for $m << \sqrt{N}$. The improved performance achieved in REEP is such that the flooding is reduced, because in DD the gradient is established by exchanging control information in *three* steps (Figure 5.2), whereas in REEP, control information is exchanged in *two* steps (Figure 5.3). This is validated through the simulation results (see Section 5.3).

Before introducing the quantitative comparison, we first describe the implementation steps of our REEP protocol.

5.2 Implementations

We have used MATLAB 7.4 for our implementation of DD and REEP. Each event propagation is simulated as an individual packet transmission, i.e., *interest*, *reply*, *reinforcement* and *data* packets are used as event propagation in DD and *sense*, *info*, *request* and *data* packets are used as event propagation in REEP.

5.2.1 Assumptions

In order to implement REEP, we have make some assumptions about communication ranges, technique used for node placement and different data structures details. All the nodes are assumed to be stationary in our sample network application, where they are not sensing all the time.

Communication range: In a grid system, we assume that the distance between two horizontal or vertical nodes is 10 meters and each center node (Node A in Figure 5.5) can have at most 8 neighbors surrounding it within its vicinity. Based on this assumption, each node's radio range is calculated to be 14.142 m (Figure 5.5). This value of radio range is constant in all of our experiments. Different communication ranges, such as sensing, reception and transmission range of each sensor node have been assumed equal in our thesis in order to observe the performance of both protocols in terms of energy efficiency.

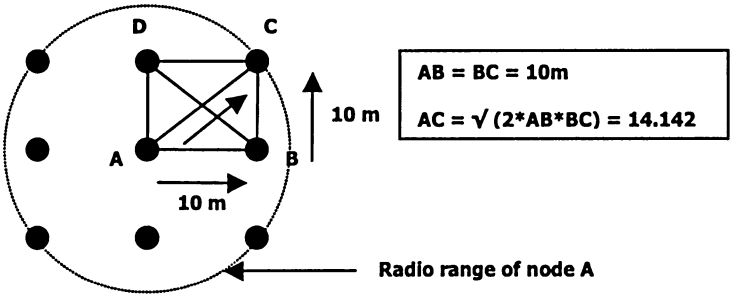


Figure 5.5: Radio Range of each Sensor Node

Node placement in topological area: We have placed each sensor node within a rectangular area in a pseudo-random fashion (Figure 5.6). This arrangement is referred to as "pseudo-random" because a node can be placed anywhere within a rectangular region or cell, but the location of that cell is fixed. The topology area (i.e., width and height) of the network is an input in our simulation. We have followed a grid topology to place the nodes in such a way that the distance between two nodes is not fixed; rather, it is a random number within a range. For example, if the average distance between the nodes is 10, then the distance between two nodes can vary from 5 to 15 (Figure 5.6) based on this calculation: (x or y value in grid system) + random (5). This topology allows one to avoid isolating any sensor node from all its neighbors. A totally random arrangement of sensor nodes in a topological area can result in increasing the congestion in some sub-areas or it can result in a sparse area with dispersed and isolated sensor nodes. This phenomenon can cause some sort of uncertainty in data transmission. If any intermediate node does not have any reachable neighbor, a path cannot be constructed from sink to source and the data can get lost. Our pseudo-random nature of node placement does not necessitate having a fixed number of neighbors for each node; rather it varies from node to node.

Sensor nodes can have different range limits for different activities depending on the requirements of any application. We have assumed that the sensing, transmitting and receiving range of every sensor node are equal. Thus, the neighbor list of each node includes all those neighbor nodes that belong to its radio range (see Figure 5.5).

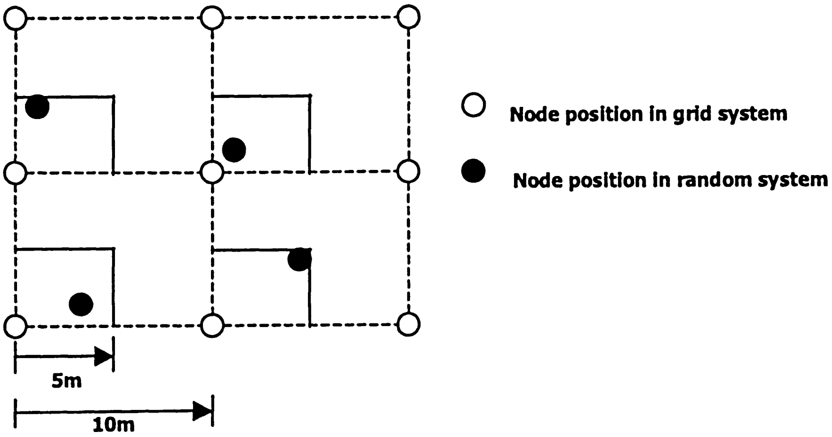
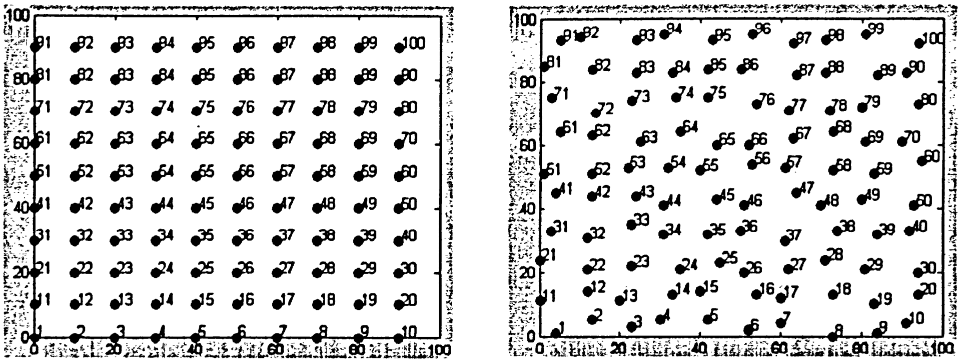

○ Node position in grid system
● Node position in random system
5m
10m

Figure 5.6: Node Placement in a Pseudo-random fashion

Based on the above placement technique, we have used a setup of 100 nodes in a 100 by 100 meter square area (see Figure 5.7). Node placement can either be in a grid arrangement (see Figure 5.7 (a)) or in a random arrangement (see Figure 5.7 (b)) system.



(a) Grid placement (b) Random placement

Figure 5.7: Placement of 100 nodes in a (100X100)m square area

Structure of a Sensor Node: Table 5.3 describes the data structure of each sensor node. The structure of a sensor node is at the top of hierarchy of all data structures used in our simulation to store the state information.

Table 5.3: Structure of a Sensor Node

SL no	Field Name	Description
1	ID	Unique ID of a sensor node
2	Location	Position of the sensor node in the network
3	NeighborList	The list of each node's neighbors
4	TotalObject	Total number of entries in each node's cache for different tasks
5	PacketList	Queue of pending packets to be processed
6	SenseSink	Node ID of the sink node that generated the *sense* event
7	SenseTime	*Sense* event generation time
8	Source	This field is true if the node is a source node
9	Energy	Lifetime of a sensor node
10	receivePkCount	Counter of total received packets
11	sendPkCount	Counter of total sent packets
12	Active	Flag to indicate an alive or a dead sensor node

13	ObjectType (Table 5.4)	All the information about the detected object type are stored in this field

Structure of an ObjectType: Table 5.4 illustrates the structure of each detected object. This data structure is used to store all the details information about the objects detected by a source node.

Table 5.4: Structure of an ObjectType

SL no	Field Name	Description
1	Type	Indicates the detected object type
2	Source	Source node's ID which has detected the object in REEP
3	InfoTime	*Info* event generation time in REEP
4	FIFOqueue	In REEP, Request for path setup is sent to the first node taken from this queue each time
5	RequestReceivedFrom	ID of the neighbor from which a node accepts the request in REEP
6	RequestPacket	Request packet is saved for future use incase of alternate path setup in REEP
7	DataPayload	Generated data are stored in this field in source node
8	PrevSender	Node ID of the neighbor who has sent request
9	RequestTime	*Request* event generation time in REEP
10	Rect	Specified rectangular area included in the *interest* event in Directed Diffusion
11	InterestTime	*Interest* event generation time
12	DataTime	Data generation time in Directed Diffusion
13	DataGenerated	Once data has been generated, this field becomes true
14	ReinforcementPacket	Reinforcement packet is saved for future use in case

		of alternate path setup in Directed Diffusion
15	ReinforcementTime	Reinforcement generation time in Directed Diffusion
16	ReplyTime	Reply generation time in Directed Diffusion
17	ReinforceReceivedFrom	ID of the neighbor from which a node accepts the reinforcement packet in Directed Diffusion
18	Intensity	Indicates a relative distance to the detected object
19	Confidence	Indicates a degree of match with the detected object
20	Gradient	This field contains the gradient value for each of the individual neighbors in Directed Diffusion

Structure of a Packet: Table 5.5 shows the structure of a packet that we have used in our implementation. Each event's propagation (see Section 4.3) has been simulated as a packet transmission.

Table 5.5: Structure of a Packet

SL no	Field Name	Description
1	Type	Indicates the packet type
2	ObjectType	Indicates the detected object type
3	TimeStamp	Packet generation time
4	Sink	Sink node's ID for this packet in REEP
5	Payload	Real data payload is stored in this field of a packet
6	SenderID	Sender ID of this packet
7	Source	Source node's ID for this packet in REEP
8	Last	This field is true if this is the last packet of a source
9	Deny	Low energy node assigns the value "true" to this field before sending a packet to indicate the status of the node in REEP
10	Rect	Specified rectangular area indicated in the *interest* event in Directed Diffusion
11	Interval	Specified time interval included in the *interest* event in DD

5.2.2 Implementation Steps

In the implementation phase, we have defined a network topology area where we have placed all the sensor nodes either in grid or random fashion. Based on the communication range, a neighbor's list is created in each sensor node. Before any transmission starts, we calculate the total available energy of the whole network. After receiving a packet from a neighbor, a node first checks whether it should act as a sink or source or intermediate node for this particular packet, then checks the packet type. According to the relevant node and packet type, each packet is then processed by the corresponding user defined function within the node. In order to maintain the sequence of packet processing, each node maintains a packet list. After processing a packet (according to the functionalities of the protocol), packets are then forwarded to the neighbors. Next, a path is established from the sink to source nodes for data transmission. The implementation steps are similar for REEP and DD.

5.3 Simulations

In order to analyze the performance of REEP and DD as functions of network size and energy, we have simulated a variety of different sized sensor fields with different setup for a variety of scenarios. These scenarios reflect the behavior of both protocols, and some of them highlight the advantages of REEP compared to DD for some specified scenarios and application types.

In all of our simulations, we considered the farthest distance between the source and the sink node such that the position of a sink node is the most upper right corner node (Node 100 in Figure 5.7) and the position of the source node is the lower left corner (Node 1 in Figure 5.7). We have placed the sink and source nodes in this way to observe the performance of both protocols in the worst-case scenario (i.e. when a path includes a maximum number of hops).

For our simulations, we have provided only snapshots of the output files, rather than the complete output files.

5.3.1 Simulation Inputs

The user inputs to our sensor network simulation are as follows:
TOTAL_SENSOR_NODES indicates the number of total sensor nodes used in the sensor network. ENERGY is the total initial energy available at each node. AREA_WIDTH is the width of a topological area. The RECT area and the OBJECT_TYPE are used to define *interest* in DD. The value of ENERGY_TRESHOLD is used in REEP to find reliable paths. MAX_DATA indicates the maximum number of data that are generated in the source node and MAX_OBJECT indicates the maximum number of detected objects. There are other specific inputs to the simulation, which are calculated based on the user' inputs. These are:

NODES_IN_ROW = ceil (sqrt (TOTAL_SENSOR_NODES)) (4)

SINK = TOTAL_SENSOR_NODES – mod (TOTAL_SENSOR_NODES, NODES_IN_ROW) (5)

NODE_HOR_VERT_DISTANCE = AREA_WIDTH / NODES_IN_ROW (6)

5.3.2 Performance Metrics

We use four performance metrics to analyze and compare the performance of DD and REEP for the aforementioned specified applications. These metrics are as follows:

Average packet transmission: This value measures the average number of packet transmissions per node, per task and is determined by the following equation:

$$\frac{(P_r + P_t)/2}{N \times T} \tag{7}$$

In the above equation, P_r denotes the total number of packet received and P_t denotes the total number of packet transmitted in the network. The sum of P_r and P_t has been divided by two, because one transmission includes the reception and the transmission of each packet. Here ***N*** indicates the total number of sensor nodes and ***T*** indicates the total number of tasks. The lower value of this metric indicates a lesser number of packet transmissions by each node as well as less energy consumption and better performance.

Average dissipated energy: It indicates the average amount of energy spent in each node for each individual task. An increased value of average dissipated energy indicates more power consumption by each node. This metric is computed with the help of the following equation:

$$\frac{\sum_{i=1}^{N} (IEi - REi)}{N \times T} \qquad (8)$$

In the above equation, ***N*** denoted the total number of sensor nodes and ***T*** denotes the total number of tasks. For each node ***i***, the used energy is the difference between the ***IE*** (the initial energy available in the node ***i***) and the ***RE*** (the remaining energy in node ***i*** after simulation).

Average data loss ratio: It indicates the average value of data loss ratio. The average data loss ratio is calculated by the following equation:

$$\frac{\sum_{i=1}^{T} \frac{MDi - RDi}{MDi}}{T} \qquad (9)$$

In the above equation, ***T*** denotes the total number of tasks and for each task ***i***, ***MD*** denotes the maximum data generated in the source node and ***RD*** indicates the total data received in the sink node. Data loss ratio for each task ***i*** has been computed by dividing the number of lost data by the maximum generated data. An increased value of average data loss ratio indicates more data loss.

Average delay: It indicates the time duration required for receiving the first data after the query generation at the sink node for a particular task. An increasing average delay indicates more time requirement. The average delay is computed with the help of the following equation:

$$\frac{\sum_{i=1}^{T} (tDi - tIi)}{T} \tag{10}$$

In the above equation, ***T*** is the total number of tasks and for each task ***i***, ***tD*** denotes the time when sink receives the first data for ***i***, and ***tI*** denotes the initial time when sink generated the query for ***i***.

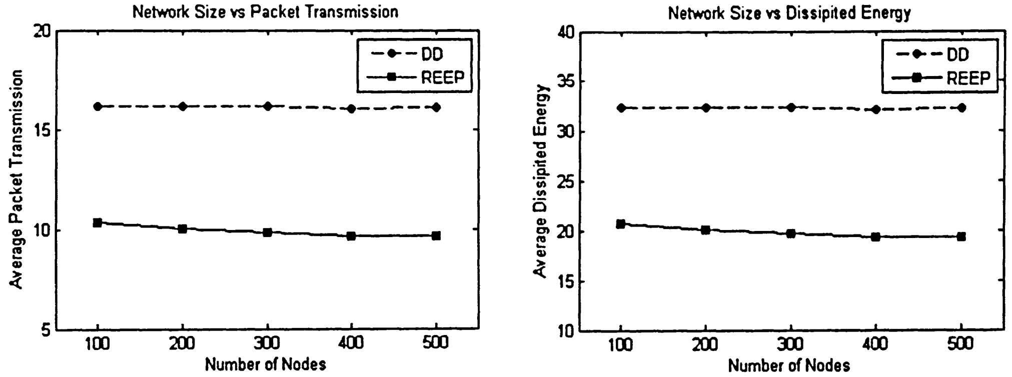
5.4 Simulation Results

Different simulation results are shown here for different cases. REEP performs comparatively better than DD in every experiment given in our work for the aforementioned type of applications. Since the gradient setup and low rate flooding of exploratory events (in case of alternate path discovery) are not followed in REEP, we can observe the better performance of REEP in terms of average packet transmission and average dissipated energy in all graphs. Flooding rate has been reduced in REEP, which contributes to the performance in terms of average delay in all graphs. The technique used to handle the fault tolerance issue in REEP, where data packets are not getting lost in the low energy' nodes (See Figure 4.9), contributes to the performance in terms of average data loss ratio in all graphs. These cases are explained in the following sections. Note that, all the value points in all graphs are the average of ten simulation runs.

5.4.1 Performance in Grid arrangement of nodes

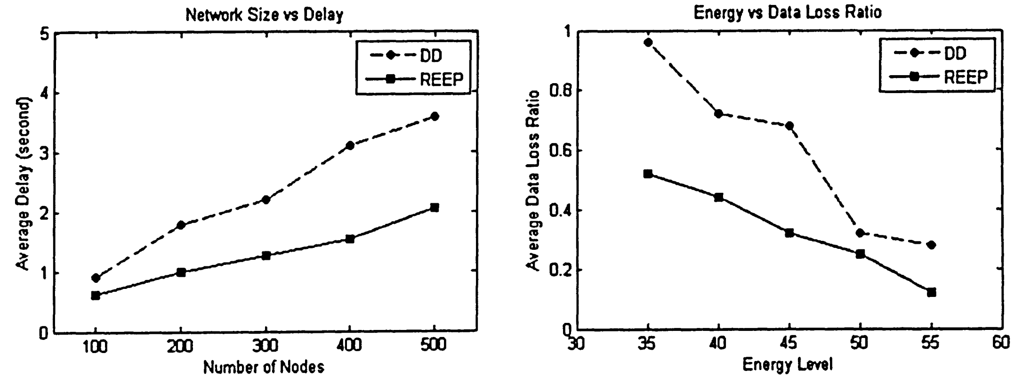
In our experiments, the performances of DD and REEP are compared in terms of the *average packet transmission*, *average dissipated energy*, *average delay*, and a*average data loss ratio*. We have simulated five different network sizes, with an increment of 100 nodes each time, ranging from 100 to 500 nodes. The sensor field has been generated by placing all the nodes in a grid fashion within a square area, and by scaling and keeping the communication range constant. If we do not keep the density constant, the performance due to increased network size will be affected by the effect of performance due to increased connectivity. The initial available energy at each node is 150 in cases of average packet transmission, average dissipated energy, average delay and 100 nodes have been used to in case of average data loss ratio.

In a grid system network, the values of average packet transmission (Figure 5.8-(a)) and the average dissipated energy (Figure 5.8- (b)) are not affected by the increase in the network size.



(a) Average Packet Transmission

(b) Average Dissipated Energy



(c) Average Delay (second)

(d) Average Data Loss Ratio

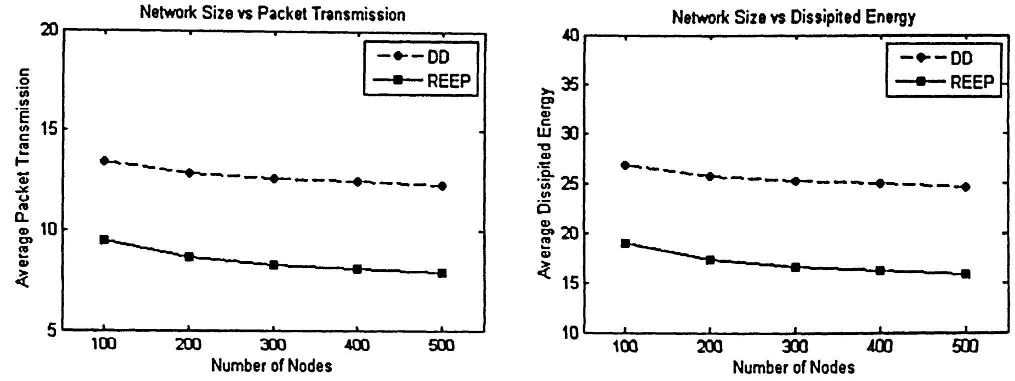
Figure 5.8: Performance in grid arrangement

Since we have kept the communication range constant for every node, the number of neighbors for each node is the same in each increased size of the network. Thus, the number of packet transmission remains almost the same at each node. Therefore, the dissipated energy also remains similar in every network.

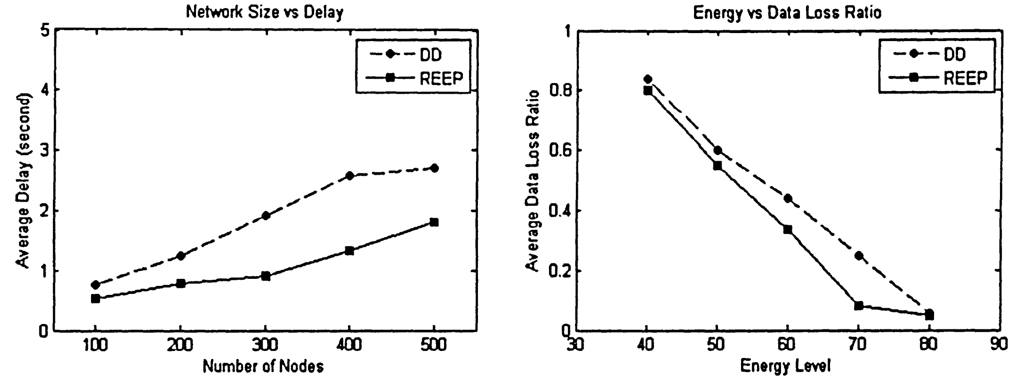
The average delay (Figure 5.8- (c)) increases with an increased network size because the hop count increases on the path. When there are 100 nodes in the network, the difference between DD and REEP is small in terms of average delay, but this difference increases with the increased number of nodes. This behavior is reflected inversely in terms of average data loss ratio (see Figure 5.8- (d)).

5.4.2 Performance in Random arrangement of nodes

The network setup is the same as in Section 5.6.1. Only the node placement has been done here in a random fashion. The results are shown in Figure 5.9. We can observe that the average packet transmission (Figure 5.9- (a)) and the average dissipated energy (Figure 5.9- (b)) decrease with the increase in the number of the nodes. Also, the data values are smaller in this random arrangement network nodes compared to the grid network arrangement case. Both protocols perform similarly in terms of average delay (Figure 5.9- (c)) in both the grid and the random systems with increased network size. The values of the average data loss ratio (Figure 5.9- (d)) are very close for both DD and REEP.



(a) Average Packet Transmission (b) Average Dissipated Energy

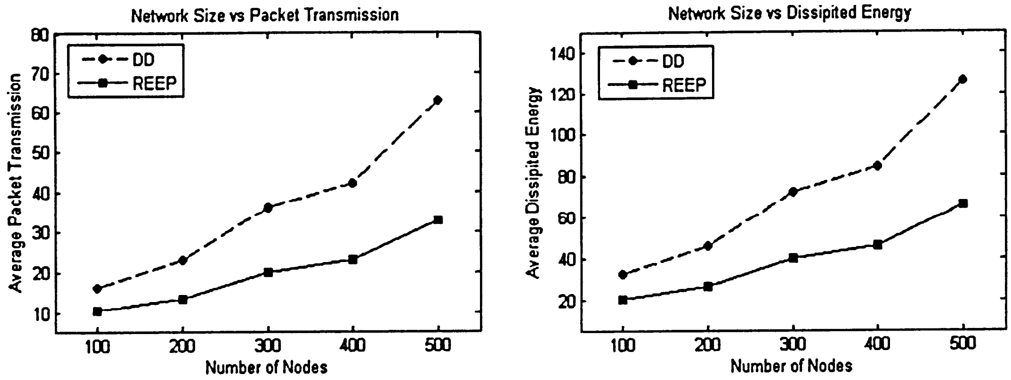


(c) Average Delay (second) (d) Average Data Loss Ratio

Figure 5.9: Performance in random arrangement

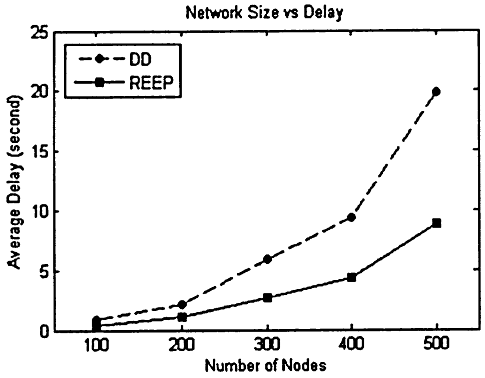
5.4.3 Performance based on Density in Grid arrangement

In previous experiments, we observed the performances of REEP and DD when increasing the network size but keeping the same network density in all cases. In this section, we report the results of experiments where we have set up the node's placement area at 100×100 meters-square. We can observe from Figure 5.10 that with the increment in the number of nodes in the network, unlike previous experiments, performance of each protocol varies in terms of average packet transmission (Figure 5. 10- (a)) and average dissipated energy (Figure 5. 10- (b)). It can be observed that when the density increases, the number of neighbors also increases for each node. Therefore, there is an increase of packet transmission, average dissipated energy and average delay (Figure 5. 10- (c)).



(a) Average Packet Transmission

(b) Average Dissipated Energy

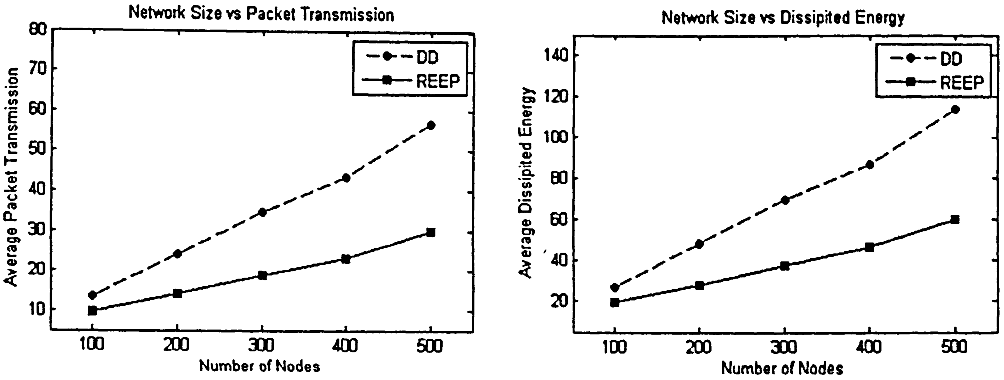


(c) Average Delay (second)

Figure 5.10: Performance according to Network Density (grid arrangement)

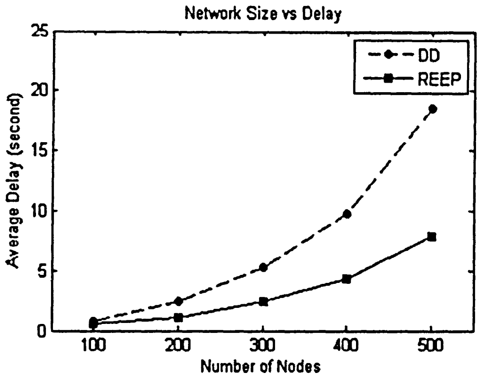
5.4.4 Performance based on Density in Random arrangement

When the density of the nodes increases in both the grid and random arrangements of nodes in the network, it can be observed that REEP and DD perform almost similarly.



(a) Average Packet Transmission

(b) Average Dissipated Energy



(c) Average Delay (second)

Figure 5.11: Performance according to Network Density (random arrangement)

5.4.5 Fault Tolerance in REEP

Figure 5.12 shows the performance of REEP with and without the fault tolerance. We have simulated REEP over 100 nodes to show how data loss ratio varies with the increased energy level. When a node fails and fault tolerance issue is absent, then alternate paths are not created and therefore data loss becomes higher. Data loss cannot be ignored in wireless communications, but can be reduced by implementing fault tolerance in routing protocols. Based on the facts illustrated in Figure 4.9, a snapshot on how low energy nodes deal with different packets is presented in Figure 5.14.

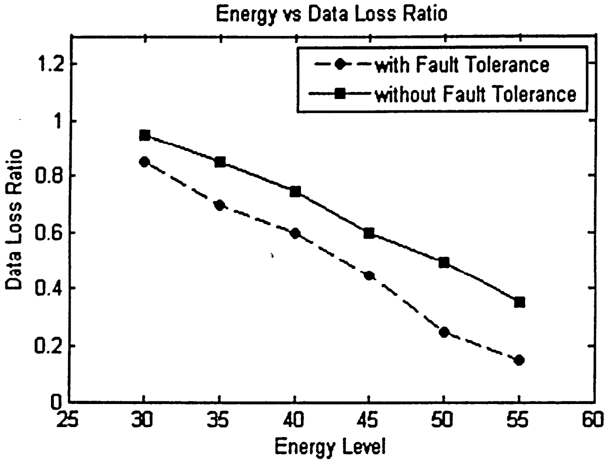
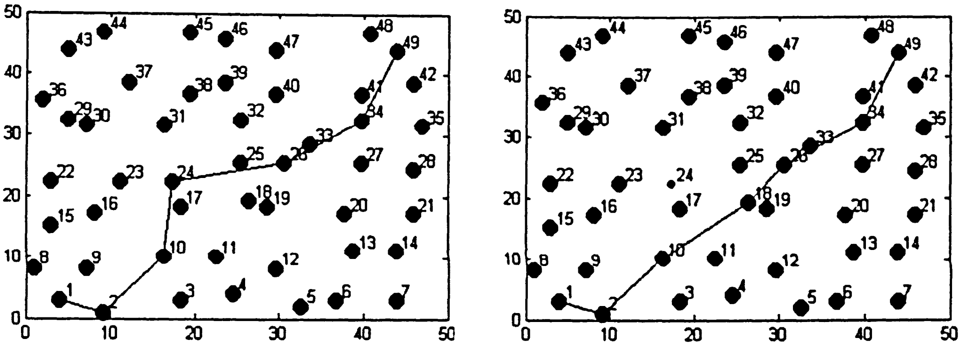


Figure 5.12: Fault Tolerance in REEP

5.4.6 Reliability in REEP

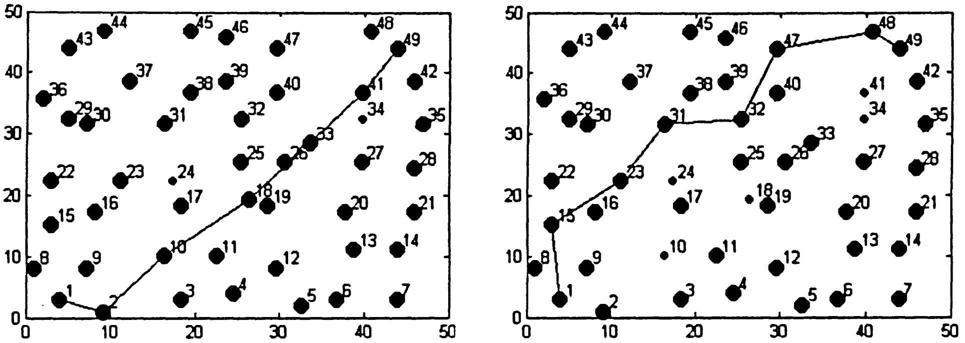
In this section, we show how the reliability issue is handled in REEP. In Figure 5.13-(a), it can be seen that a path has been established from source node 1 to sink node 49 for data transmission. In REEP, no node participates in the path setup if its energy level goes below the threshold value. Hence, we can infer that, reliable path is established in REEP for data transmission. The small red colored nodes in Figure 5. 13 indicates that their energy level is below the threshold value.

We can also observe the output of this scenario in Figure 5.14 (a snapshot from a log-file derived from our simulation). The highlighted lines show the events, where nodes with low energy have denied participating in the path construction, as well as events where nodes with high energy have participated in the path construction process.



(a) No node failure

(b) One node with energy below the threshold value



(c) Two nodes with energy below the threshold value

(d) Multiple nodes with energy below the threshold value

Figure 5.13: Reliable Path for Data Transmission

```
!!!~~~~~~~LOW ENERGY in node: 24 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 24--> sending (-)DATA pk to next node 26~~~~~~~~~~~~!!!
!!!~~~~~NODE 24--> sending (-)REQUEST pk to previous node 10~~~~~~~~~~~~!!!
~~~ NODE :26 requesting node 18 for new path construction~~~~
SINK node 49: received data for object type =1;____@ time--> H:M:S = 1:15:15
DATA = 4
PATH : 49  34  26  18  10   2   1
SOURCE NODE 1: creates packet, packet type, DATA = 4;____ @ time--> H:M:S = 1:15:16
SOURCE NODE :1-->sending data to -->2
SINK node 49: received data for object type =1;____@ time--> H:M:S = 1:15:16
DATA = 2
SOURCE NODE 1: creates packet, packet type, DATA = 4;____ @ time--> H:M:S = 1:15:16
!!!~~~~~~~LOW ENERGY in node: 34 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 34--> sending (-)DATA pk to next node 49~~~~~~~~~~~~!!!
!!!~~~~~NODE 34--> sending (-)REQUEST pk to previous node 26~~~~~~~~~~~~!!!
~~~ NODE :49 requesting node 41 for new path construction~~~~
.........
PATH : 49  41  33  26  18  10   2   1
.........
!!!~~~~~~~LOW ENERGY in node: 41 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 41--> sending (-)DATA pk to next node 49~~~~~~~~~~~~!!!
!!!~~~~~NODE 41--> sending (-)REQUEST pk to previous node 33~~~~~~~~~~~~!!!
~~~ NODE :49 requesting node 35 for new path construction
~~~~SINK node 49: received data for object type =1;____@ time--> H:M:S = 1:15:23
SOURCE NODE 1: creates packet, packet type, DATA = 4;____ @ time--> H:M:S = 1:15:23
!!!~~~~~~~LOW ENERGY in node: 41 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 41--> sending (-)DATA pk to next node 49~~~~~~~~~~~~!!!
!!!~~~~~NODE 41--> sending (-)REQUEST pk to previous node 33~~~~~~~~~~~~!!!
!!!~~~~~~~LOW ENERGY in node: 10 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 10--> sending (-)REQ pk to previous node 2~~~~~~~~~~~~!!!
~~~ NODE :49 requesting node 48 for new path construction~~~~
SINK node 49: received data for object type =1;____@ time--> H:M:S = 1:15:23
DATA = 18
SINK NODE 49: ALL DATA HAS BEEN RECEIVED FROM SOURCE:1
!!!~~~~~~~LOW ENERGY in node: 18 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 18--> sending (-)REQ pk to previous node 10~~~~~~~~~~~~!!!
!!!~~~~~~~LOW ENERGY in node: 10 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 10--> sending (-)REQ pk to previous node 2~~~~~~~~~~~~!!!
!!!~~~~~~~LOW ENERGY in node: 10 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 10--> sending (-)REQ pk to previous node 2~~~~~~~~~~~~!!!
!!!~~~~~~~LOW ENERGY in node: 24 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 24--> sending back REQ pk to sender 32~~~~~~~~~~~~!!!
!!!~~~~~~~LOW ENERGY in node: 18 ~~~~~~~~~~~~!!!
!!!~~~~~NODE 18--> sending back REQ pk to sender 32~~~~~~~~~~~~!!!
PATH : 49  48  47  32  31  23  15   1
SOURCE NODE 1: creates packet, packet type, DATA = 4;____ @ time--> H:M:S = 1:15:28
*** CURRENT ENERGY of WHOLE NETWORK TOPOLOGY is = 2451 ***
~~~Total received packet count : 981
~~~Total sent packet count : 321
~~~Total generated data packet at source  : 25
~~~Total received data packet at sink  : 25
~~~Data loss ratio : 0
~~~Average dissipated energy : 26.5714
~~~Average Packet transmission : 13.2857
```

Figure 5.14: REEP output shows different Path Construction

5.4.7 Different REEP Scenarios

We have demonstrated the usability of REEP in Section 4.4. In this section, we demonstrate the performance of REEP for those usability scenarios. Since our output figure of the network topology with path setup does not carry any information about the object types and the number of detected objects, we provide here some snapshots from the output log-files for each scenario.

Single Source detects Single/Multiple Object(s)

When a single source detects multiple objects, it creates the *info* events for each of them and sends to all the neighbors. Therefore, the average dissipated energy, the average packet transmission ratio and the average delay, increase with the increase in the number of detected object types (see Table 5.6). Figure 5.15 shows a snapshot of the output file, where we can observe that source 1 has detected three objects HUMAN, ANIMAL and VEHICLE.

```
~~~~~~~~~~~~~~START REEP~~~~~~~~~~~~~~~~~~~
*** CURRENT ENERGY of WHOLE NETWORK TOPOLOGY is = 4900 ***
SINK 49: creates SENSE packet. [time--> H:M:S = 21:29:58]
SOURCE 1: generates data for object type--> 1. [time--> H:M:S = 21:29:58]
 Generated data for HUMAN : 16,5,6,19,8,
SOURCE 1: creates INFO packet. [time--> H:M:S = 21:29:58]
SOURCE 1: generates data for object type--> 2. [time--> H:M:S = 21:29:58]
 Generated data for ANIMAL : 8,19,15,2,12,
SOURCE 1: creates INFO packet. [time--> H:M:S = 21:29:58]
SOURCE 1: generates data for object type--> 3. [time--> H:M:S = 21:29:58]
 Generated data for VEHICLE : 11,12,10,17,14,
SOURCE 1: creates INFO packet. [time--> H:M:S = 21:29:58]
SINK 49: creates REQUEST packet. [time--> H:M:S = 21:29:58]
SINK 49: creates REQUEST packet. [time--> H:M:S = 21:29:58]
SINK 49: creates REQUEST packet. [time--> H:M:S = 21:29:58]
PATH : 49  41  34  26  18  17   9   8   1
SOURCE 1: creates DATA packet. [time--> H:M:S = 21:29:58]
PATH : 49  41  34  26  18  17   9   8   1
SOURCE 1: creates DATA packet. [time--> H:M:S = 21:29:58]
PATH : 49  41  34  26  18  17   9   8   1
SINK 49: ALL DATA HAS BEEN RECEIVED FROM SOURCE: 1
SINK 49: ALL DATA HAS BEEN RECEIVED FROM SOURCE: 1
SINK 49: ALL DATA HAS BEEN RECEIVED FROM SOURCE: 1
*** CURRENT ENERGY of WHOLE NETWORK TOPOLOGY is = 3526 ***
---Total received packet count in the network: 1026
---Total sent packet count in the network: 348
~~~~~~~~~~~~~~~~END REEP~~~~~~~~~~~~~~~~~
```

Figure 5.15: REEP output showing Single Source detects Multiple Objects

Table 5.6: Performance varies with Multiple Objects (Single Source)

Number of detected objects	Average Dissipated Energy	Average Packet Transmission	Average Delay
1	13.449	6.7245	0.4
2	22.102	11.051	0.7
3	29.2245	14.6122	1

Multiple Sources detect Single/Multiple Object(s)

Figure 5.16 shows a snapshot of the output file, where it can be seen that each of the sources 1, 2, 8 and 9 have detected three objects HUMAN, ANIMAL and VEHICLE. Table 5.7 also shows the performance of REEP, where one can observe that the values for the mentioned three metrics are very high in case multiple sources detect multiple objects, compared to the single source performance case shown in Table 5.6.

Table 5.7: Performance varies with Multiple Objects (Multiple Sources)

Number of detected objects	Average Dissipated Energy	Average Packet Transmission	Average Delay
1	36.8571	18.4286	1.3
2	65.9184	32.9592	2.5
3	98.5306	49.2653	3.7

```
~~~~~~~~~~~~~~START REEP~~~~~~~~~~~~~~~~~~~

*** CURRENT ENERGY of WHOLE NETWORK TOPOLOGY is = 14700 ***

SINK 49: creates SENSE packet. [time--> H:M:S = 22:22:56]
SOURCE 9: generates data for object type--> 1. [time--> H:M:S = 22:22:56]
 Generated data for HUMAN : 6,7,11,18,9,
SOURCE 9: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 9: generates data for object type--> 2. [time--> H:M:S = 22:22:56]
 Generated data for ANIMAL : 16,18,5,18,17,
SOURCE 9: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 9: generates data for object type--> 3. [time--> H:M:S = 22:22:56]
 Generated data for VEHICLE : 16,3,10,5,14,
SOURCE 9: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 2: generates data for object type--> 1. [time--> H:M:S = 22:22:56]
 Generated data for HUMAN : 5,6,6,18,17,
SOURCE 2: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 2: generates data for object type--> 2. [time--> H:M:S = 22:22:56]
 Generated data for ANIMAL : 16,1,4,13,12,
SOURCE 2: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 2: generates data for object type--> 3. [time--> H:M:S = 22:22:56]
 Generated data for VEHICLE : 17,19,9,16,4,
SOURCE 2: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 8: generates data for object type--> 1. [time--> H:M:S = 22:22:56]
 Generated data for HUMAN : 16,4,1,6,19,
SOURCE 8: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 8: generates data for object type--> 2. [time--> H:M:S = 22:22:56]
 Generated data for ANIMAL : 3,9,18,16,2,
SOURCE 8: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 8: generates data for object type--> 3. [time--> H:M:S = 22:22:56]
 Generated data for VEHICLE : 5,4,16,10,1,
SOURCE 8: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 1: generates data for object type--> 1. [time--> H:M:S = 22:22:56]
 Generated data for HUMAN : 6,7,13,5,17,
SOURCE 1: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 1: generates data for object type--> 2. [time--> H:M:S = 22:22:56]
 Generated data for ANIMAL : 5,12,8,16,6,
SOURCE 1: creates INFO packet. [time--> H:M:S = 22:22:56]
SOURCE 1: generates data for object type--> 3. [time--> H:M:S = 22:22:56]
 Generated data for VEHICLE : 10,7,4,4,1,
SOURCE 1: creates INFO packet. [time--> H:M:S = 22:22:56]
---Total received packet count in the network: 3681
---Total sent packet count in the network: 1195
~~~~~~~~~~~~~~~~END REEP~~~~~~~~~~~~~~~~~
```

Figure 5.16: REEP output showing Multiple Sources detect Multiple Objects

Multiple Sources and Multiple Sinks in REEP

Figure 5.17 shows one instance of REEP, in which paths are created between multiple sources and sinks. Nodes 96 and 80 are the sink nodes. Nodes 11 and 2 are the source nodes, which have generated data for the detected objects. However, this figure does not indicate how many objects have been detected by each source. Such information can be extracted from its output log-file. According to the REEP scheme (Chapter 4), the sink nodes initiates the path setup mechanism with the source nodes based on the received information about the source nodes and their detected object types, then they start receiving the data along the corresponding paths.

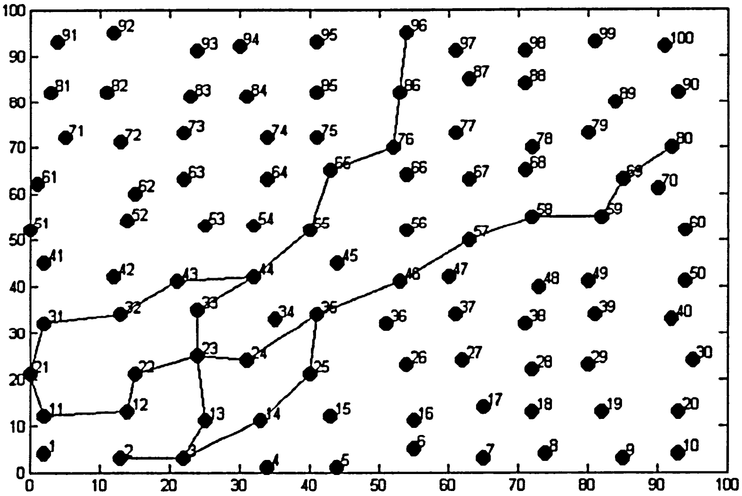


Figure 5.17: REEP with Multiple Sinks and Multiple Sources

5.5 Summary

An analysis of our experimental results shows that REEP performs better than DD both with respect to energy efficiency and fault tolerance, for the specified application types and scenarios. In order to maximize the lifetime of a wireless sensor network, energy resources of each individual sensor node must be spent in an effective way. Unnecessary flooding of control information is avoided in REEP to maximize the node's lifetime as well as the entire network lifetime.

The network density is another important parameter that can significantly alter the performance of a protocol. For example, a dense network introduces more connectivity among nodes with a large number of neighbors. This mechanism simply trades off some energy efficiency for increased robustness. In most of our experiments (except those reported in Sections 5.6.3 & 5.6.4), we have maintained a constant network density.

Results of our study have shown a clear view of the performance of both protocols in the studied usability scenarios. We believe that these performance criteria should be reflected in a similar manner in most of the usability scenarios.

Most of the sensor network protocols are application specific, so are data-centric type protocols. Since we have followed the design structure of DD, it is obvious that both DD and REEP are suitable for the same type of applications. But depending on the performance analysis, REEP will work better for large network sizes compared to DD.

Chapter 6

CONCLUSION

This chapter concludes the thesis work by including our main contributions and some discussions arising from our work. We also discuss possible future works in this field of study.

6.1 Thesis Contributions

Although we have mentioned several times in our work that our proposed protocol has been designed following the DD approach, there are some design techniques that are our own contribution in this thesis work. They are highlighted as follows:

- The design idea of the first phase (i.e. *sense event propagation*) of REEP, in which all the sensor nodes sense for a specified time to scan the whole network for available objects.

- The design idea of the second phase (*i.e. info event propagation*) of REEP, where *info events* are forwarded after saving all the necessary information from that event (see details in Section 5.1.3).

- Checking of the energy threshold value during only *request event* and *data* transmission.

- The maintenance of the FIFO queue for each task in every node. Although the use of this queue introduces some looping problem, we overcome this problem by manipulating the queue in such a way that it removes specific neighbor information from the queue (see Section 4.7.2 for details).

- A technique for reducing the data loss (illustrated in Figure 4.9). In this technique, the data packet is sent to the next neighbor by changing the status of the packet as negative. This technique has been validated through our experimental results as shown in Figure 5.8-(d) and Figure 5.9-(d).

6.2 Discussions

Wireless sensor networks are not used for general-purpose communication. Based on the knowledge of survey results [AS02][AK04][AY05] on wireless sensor network protocols, many of them are likely to be application-specific. We have designed and described in detail our new proposed REEP protocol. Communication paths established in REEP are inspired by the observation of strictly local communication in physical system [GS02]. In such systems, path setup functions cannot use global topology metrics.

From the detailed design of REEP, we can summarize several key features that indicate how it differs from other data-centric routing protocols in the sensor networking paradigm. Firstly, REEP is an interactive on-demand protocol in which path establishment can be done based on the choice of any user or application. Secondly, each node maintains an energy threshold value and participates in the path setup with adequate energy for data transmission. Finally, the FIFO queue is used for solving some looping problems and for alternate path setup, in case of path failure, without invoking periodic low rate flooding.

Furthermore, REEP is best suited for security maintenance by location tracking applications, where periodical observation of environmental phenomena is not required. REEP can also be modified for event-triggered applications, in which sensor nodes sense all the time and notify the sink whenever any event has been detected.

6.3 Future Works

REEP has been designed for sensor networks where sensor nodes are stationary. One possible future work is the redesign of REEP with mobility features included. In addition, REEP can be adopted for event-triggered applications, which requires all time sensing of changing physical phenomena.

To help a user selecting a specific task in the network, one can introduce the location information of the detected object as an additional parameter within the *info event* of the REEP protocol. Adding this feature requires some work at the REEP design level.

As mentioned earlier, sensor nodes are energy constrained and capable of less complex computations, it may be possible in future to use some special kind of power generator nodes in the network to supply extra energy to the low powered sensor devices. This technology will give flexibility to designers to design more complex algorithm for sensor networks. Finally, signal amplifiers can be used within the sensor field to transmit poor signals properly from one end to the other end of the sensor network.

BIBLIOGRAPHY

[MA01] A. Manjeshwar and D. P. Agrawal; "*TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks*", In the Proceedings of the 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing, San Francisco, CA, April 2001

[MP02] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, John Anderson; "*Wireless Sensor Networks for Habitat Monitoring*", Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, Atlanta, Georgia, USA, 2002, pp.88 - 97

[PC06] C Park, PH Chou; "*eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera*", Proceedings of the 4th international conference on Embedded networked sensor systems, 2006, pp.359 - 360

[PCB06] C Park, PH Chou, Y Bai, R Matthews, A Hibbs; "*An Ultra-Wearable, Wireless, Low Power ECG Monitoring System*", Proceedings of IEEE BioCAS, Nov, 2006

[SS01] C. Schurgers and M.B. Srivastava; "*Energy efficient routing in wireless sensor networks*", In the MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force, McLean, VA, 2001

[SSJ01] C. Shen, C. Srisathapornphat, C. Jaikaeo; "*Sensor information networking architecture and applications*", IEEE Personal Communications, August 2001, pp.52–59

[IG03] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva; "*Directed Diffusion for Wireless Sensor Networking*", IEEE/ACM Transactions on Networking (TON), February 2003, pp.2-16, 11(1)

[LK03] Changling Liu, Jörg Kaiser; "*A Survey of Mobile Ad Hoc network Routing Protocols*", University of Ulm Tech.Report Series, Nr. 2003-08, Oct. 2005

[CK03] Chee-Yee Chong, Srikanta P. Kumar; "*Sensor networks: evolution, opportunities, and challenges*", Proceedings of the IEEE, Aug. 2003, pp.1247-

1256, 91(8)

[BE02] D. Braginsky and D. Estrin; "*Rumor Routing Algorithm for Sensor Networks*", In the Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA, October 2002

[GG01] D. Ganesan, R. Govindan, S. Shenker, D. Estrin; "*Highly Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks*", ACM SIGMOBILE Mobile Computing and Communications Review, 2001, pp.11–25, 5(4)

[CE04] David Culler, Deborah Estrin, Mani Srivastava; "*Overview of sensor networks*", IEEE Computer Society, New York, NY, Aug. 2004, pp.41-49, 37(8)

[BC97] F. Bennett, D. Clarke, J. Evans, A. Hopper, A. Jones, and D. Leask; "*Piconet: Embedded Mobile Networking*", IEEE Personal Communications, Oct. 1997, pp., 4(5)

[HS00] G. Hoblos, M. Staroswiecki, A. Aitouche; "*Optimal design of fault tolerant sensor networks*", IEEE International Conference on Control Applications, Anchorage, AK, Sep. 2000, pp.467–472

[PK00] G.J. Pottie, W.J. Kaiser; "*Wireless integrated network sensors*", Communications of the ACM, 2000, pp.551– 558, 43(5)

[PK0] Gregory J. Pottie and William J. Kaiser; "*Embedding the internet: wireless integrated network sensors*", Communications of the ACM, May 2000, pp.51–58, 43(5)

[AS02] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci; "*Wireless sensor networks: a survey*", Computer Networks, Elsevier, 15 March, 2002, pp.393-422, 38(4)

[KK99] J.M. Kahn, R.H. Katz, K.S.J. Pister; "*Next century challenges: mobile networking for smart dust*", Proceedings of the ACM MobiCom'99, Washington, USA, 1999, pp.271–278

[AK04] Jamal N. Al-Karaki, Ahmed E.Kamal; "*Routing techniques in wireless sensor networks: a survey*", Wireless Communications, IEEE, Dec. 2004, pp.6- 28, 11(6)

[KR99] Joanna Kulik, Wendi Rabiner, and Hari Balakrishnan; "*Adaptive Protocols for Information Dissemination in Wireless Sensor Networks*", In Proceedings of the

Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, WA, 1999

[HS01] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan; "*Building efficient wireless sensor networks with low-level naming*", In Proceedings of the ACM Symposium on Operating Systems Principles, Banff, Canada, Oct. 2001

[AY05] Kemal Akkaya, Mohamed Younis; "*A Survey on Routing Protocols for Wireless Sensor Networks*", Ad Hoc Networks, Elsevier, May 2005, pp.325-349, 3(3)

[NX06] KS No, Q Xie, R Kwong, A Cerussi, BJ Tromberg, PH Chou; "*HBS: a Handheld Breast Cancer Detector Based on Frequency Domain Photon Migration with Full Heterodyne*", Proc. IEEE Biomedical Circuits and Systems, 2006

[GS02] M Gunes, U Sorges, I Bouazizi; "*The Ant-Colony based routing algorithm*", International workshop on Ad-hoc Networking (IWAHN 2002), 2002

[CH02] M. Chu, H. Haussecker, and F. Zhao; "*Scalable Information-Driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Networks*", The International Journal of High Performance Computing Applications, August 2002, pp., 16(3)

[SK03] N Sadagopan, B Krishnamachari, A Helmy; "*The ACQUIRE mechanism for efficient querying in sensor networks*", First International Workshop on Sensor Network Protocol and Applications, Anchorage, Alaska, May 2003

[SR02] R. Shah and J. Rabaey; "*Energy Aware Routing for Low Energy Ad Hoc Sensor Networks*", In the Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Orlando, FL, March 2002

[SB02] S. Servetto and G. Barrenechea; "*Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks*", Proc. 1st ACM Int'l. Wksp. Wireless Sensor Networks and Apps., Atlanta, GA, 2002

[WS99] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley; "*The Design and Implementation of an Intentional Naming System*", In Proceedings of the ACM Symposium on Operating Systems Principles, Charleston, SC, 1999, pp.186–201

[XH01] Y. Xu, J. Heidemann, and D. Estrin; "*Geography informed Energy Conservation for Ad-hoc Routing*", Proc. 7th Annual ACM/IEEE Int'l. Conf. Mobile Comp.and Net., 2001, pp.70–84

[YG02] Y. Yao and J. Gehrke; "*The cougar approach to in-network query processing in sensor networks*", In SIGMOD Record, September 2002

[WIKI] http://en.wikipedia.org/wiki/Ad_hoc_routing_protocol_list

[PSTR] http://robotics.eecs.berkeley.edu/~pister/SmartDust/

[BRK1] "http://www-bsac.eecs.berkeley.edu/archive/users/warneke-brett/SmartDust/index.html"

[CHOU] http://www.ece.uci.edu/~chou/research/

[MOTE] http://www.ecomote.net/

[BRK2] http://webs.cs.berkeley.edu/800demo/

[DSON] http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/2006/02&file=x6glo.xml&xsl=article.xsl

GLOSSARY OF TERMS

WSN: Wireless Sensor Networks

DD: Directed Diffusion

REEP: Reliable and Energy Efficient Protocol

ADC: Analog to Digital Converter

MANET: Mobile Ad-hoc NETworks

GPS: Global Positioning System

FIFO: First-In-First-Out