Multipath Route Construction Methods for Wireless Sensor Networks

by

Saad Rizvi

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba

Abstract

Routing plays an important role in energy constrained Wireless Sensor Networks (WSNs). To conserve energy in WSN, energy-efficiency of the routing protocol is an important design consideration. These protocols should maximize network lifetime and minimize energy consumption.

In this thesis, a novel multipath routing protocol is proposed for WSNs, which constructs multiple paths based on residual energy of the nodes. The protocol allows the source node to select a path for data transmission from the set of discovered multiple paths based on cumulative residual energy or variance. Choosing a next-hop node based on energy, and using an alternative path for routing achieves load balancing.

The results show that the proposed algorithm M-VAR has lower residual energy variance (96%, 90%, 72%, 12% less) and longer network lifetime (404%, 205%, 115%, 10%) than basic Directed Diffusion, load-balanced Directed Diffusion (LBDD-ED-RD), multipath Directed Diffusion (MDD-CRE), and the proposed algorithm M-CRE, respectively.

Acknowledgements

First of all, I would like to thank my advisor Dr. K. Ferens for his guidance and support in the development of this thesis and throughout my graduate studies. Thank you for being a great advisor and an inspiring person to work with.

I would also like to thank the University of Manitoba and the Government of Manitoba for providing me the research scholarship and other grants to pursue my higher education in Canada.

I sincerely acknowledge and appreciate the friendship of my colleagues and friends.

Finally, I would like to thank my parents for their unwavering love and support throughout my life, and their encouragement to pursue my higher studies.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A Wireless Sensor Network (WSN) contains spatially dispersed sensor nodes that are normally deployed in an ad hoc manner (randomly scattered). A WSN is formed by a large number (hundreds, thousands) of autonomous nodes that are lightweight, inexpensive, and small in size. The main task for such a network (Fig. 1.1) is to collect data from the environment and disseminate it to a central location. The node that generates events by sensing physical data is called the "Source" node, and the node where data is collected is called the "Sink" node. The "Sink" node is different from other nodes and is used to collect data from the WSN and send it to a remote base station. A sink node or a base station is a special node which is rich in resources, such as computational capability, storage capacity, energy, and communications resources (wired or wireless). All other sensor nodes have limited computation, energy, and communications resources.

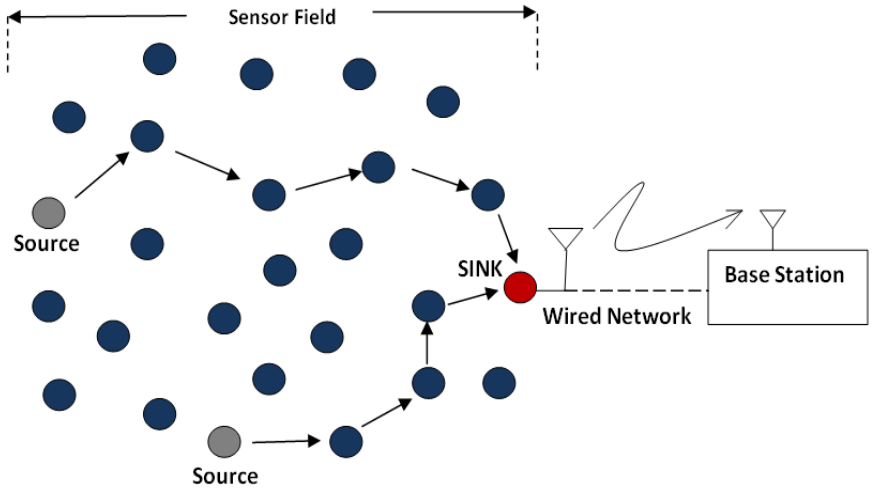

Fig. 1.1 A description of a WSN.

Wireless sensor networks are being used for many different applications. These applications can be broadly classified as habitat monitoring, healthcare, environmental monitoring, and surveillance applications. In a habitat monitoring application, sensor networks can be used to study the behavior of plants and animals [1]. An environmental monitoring application [2][3] involves modeling and forecasting of data for different physical phenomena such as temperature, humidity, flooding, and pollution. In a surveillance application, sensor nodes can be sprinkled over a large field for monitoring purposes. In these applications, sensor nodes establish a self-organizing network by registering nodes within their radio range (one-hop neighbors). The common objective in all of these applications is to gather sensor data and reliably transfer it to a resource rich (in terms of computation, energy supply, storage capacity) destination node (sink node) or to a remote base station, which can process reported events, trigger necessary actions, and maintain a database for logged events. This makes data dissemination an important issue for wireless sensor networks.

A WSN node (Fig. 1.2) consists of three main components: sensor, microcontroller, and radio electronics with antenna. A WSN node has an integrated sensor on board to sense different physical phenomena such as temperature, humidity, and turbidity. Each node is equipped with a microcontroller to handle sensing tasks, process data, and control wireless communications. One of the most important modules in a sensor node is the wireless communications module (transceiver). It is used for communicating with other nodes and transferring data from the source node to the sink node (or the base station) through multiple hops (or by covering large distances). With the help of Integrated Circuit (IC) technology and current developments in

Microelectromechanical Systems (MEMS), the integration of these components have become possible, fabricating compact nodes with reduced power consumption and low cost [4].

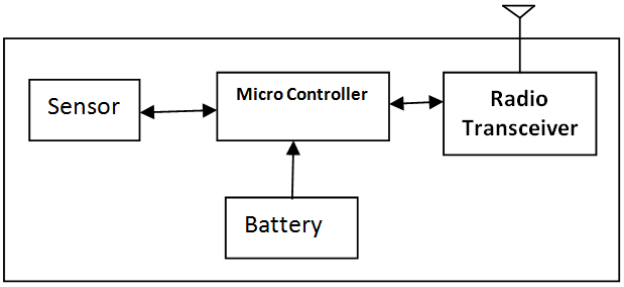

Fig. 1.2 Basic block diagram for a WSN node.

A WSN node is powered by a battery, which provides the primary source of energy. Although power harvesting is considered an alternative, it comes with hardware complexity, and ordinary nodes are equipped with small batteries. These sensor nodes are deployed in different remote locations where they operate in an unattended environment, and may not have the opportunity to replenish their batteries due to unavailability of grid power (or other sources). As a result, these nodes tend to operate for months without replacing their batteries. This makes energy consumption and its management a key issue for WSNs.

The limited energy supply of the sensor node also impacts data communications. Since wireless transmission is the main energy consuming process for the node, it is very costly to send data over long distances. Multi-hop data communications forwards data to the destination through multiple sensor nodes, using short distance communications, and this makes it more feasible and energy-conserving [5][8]. A simple and robust way to disseminate data is by flooding, in which each node that receives an incoming packet sends it out to every other node. However, flooding generates duplicate messages and a

lot of redundant traffic, which makes it energy-inefficient. For conserving energy, more sophisticated techniques are required. By carefully selecting paths to route data towards destination node, energy can be conserved. This requires the design of efficient routing schemes for WSNs.

For a wireless sensor network a lot of energy-efficient routing schemes have been proposed. These schemes can be classified based on the network structure into flat, hierarchical, and location-based routing schemes. For a simple flat-network topology, two commonly used protocols are Sensor Protocols for Information via Negotiation (SPIN) [6] and Directed Diffusion [7]. Both of these data-centric protocols can be applied to a flat network with homogeneous nodes. Hierarchical routing was proposed to achieve energy efficiency by defining a hierarchy in the network. One of the most popular routing protocols for hierarchical routing is the Low Energy Adaptive Cluster Hierarchy (LEACH) [8]. By making cluster heads in charge of routing for the network, this algorithm tries to minimize energy consumption, and effectively off-loads tasks from other normal nodes. For location based routing, a prime example of an energy-efficient protocol is Geographic and Energy Aware Routing (GEAR) [9]. GEAR uses the method of interest propagation like the Directed Diffusion, but reduces energy consumption by limiting interest flooding to a certain area in the network.

However, there are some problems associated with the routing algorithms mentioned above. In all of these protocols, communications may take place more frequently in certain parts of the network, while some parts are not used as often during the network lifetime. Furthermore, static communications paths are formed for routing data for a longer period of time. This causes the energy consumption of the nodes in the

network to be non-uniform. As a result, nodes consuming energy at a faster rate would die (fail) sooner, creating partitions in the network and making it dysfunctional. In a WSN nodes are supposed to consume low energy to extend the lifetime of the network to the maximum. While various definitions of network lifetime have been used in the literature [10], we choose to define network lifetime as the time at which any one node in the network fails (dies).

In order to extend the lifetime of the network, considering limited energy resources associated with sensor nodes, we need to design an energy-efficient solution for routing. One approach to meet this requirement of extending network lifetime is to introduce load balancing in routing algorithms. A load balancing algorithm balances the work load among each node in the network in an equitable manner. In an ideal load balanced network, the energy of each node is consumed at the same rate and time. The data needs to be forwarded such that all nodes in the network are utilized and energy consumption in the network is balanced. The network will gracefully lower energy for each node in the network by judiciously choosing nodes for routing data based on some parameters (e.g., delay, energy) during network operation. Thus, the network will cease to operate when all nodes are depleted of energy (zero residual energy) at the same time. In short, load balancing would allow maximizing the network's lifetime by uniformly distributing the energy consumption among nodes in the network.

Load balancing can be easily applied on a flat network, where nodes are homogeneous (i.e., having same capabilities) and ideally should die at the same time. In such a densely populated flat WSN, it is not a good approach to assign network identity (e.g., IP address) to each and every node in the network. For sensor networks, it is more

convenient to establish communication based on data (called data-centric routing) instead of address (address-centric routing). For a WSN, a user is interested in data generated by the sensors. If the sensor node is identified based on the data it samples, it would be easy to find the node with the required data. This type of communication is achieved using a data-centric routing protocol. Using data-centric protocol, the user can use attribute-value pairs to query certain nodes in a desired region for the data it is interested in. Data-centric protocols eliminate redundancy of data transmissions by aggregating data coming from multiple sources going to a particular destination. This makes these protocols energy-efficient.

One of the most popular data-centric routing protocols is the Directed Diffusion [7] protocol. In Directed Diffusion a user (at the sink node) can inject a query about the data it is interested in by diffusing an "interest" message in the network. An interest message contains the description of the sensing task the user is interested in. The sensed data is the physical phenomenon which a user wants to capture. This data is sent towards the user in the form of short events. The interest message is flooded within the network creating gradients. Gradients are paths used to draw events back from the source node towards the sink through multiple paths. When the interest message finds a node which can serve the task specified by the interest, this node becomes the "source" node. The source node then sends exploratory data packet towards the sink through multiple paths. The sink then reinforces one of the best paths (e.g., a path with the least-delay), and data starts flowing towards the sink through this reinforced path. In its basic form, Directed Diffusion does not have the provision of load balancing. Nodes along the reinforced path

would deplete energy faster and would cause the network to die prematurely while most of the nodes have higher residual energies.

The idea of load balancing was applied to the basic Directed Diffusion in [11]. To route sensed data from the source node towards the sink, this algorithm takes a greedy approach to find the best path. The algorithm develops a system of network wide energy updates among nodes by piggybacking residual energy values onto reinforcement message packets (RMPs) and reinforced data packets (RDPs). During the path reinforcement phase, each node reinforces the neighbor which has the highest residual energy. The algorithm, therefore, attempts to find a routing path, which has all nodes high in energy, and has the least probability of failure while data is routed through it. By routing data through multiple paths between source and sink nodes, load balancing was applied forcing each and every node to participate in routing. However, by locally choosing the next node that has the highest residual energy, we may end up at a node along the reinforced path which is surrounded by nodes low in energy. This node then has no option but to reinforce a weak node (very low in energy). So by the time communication takes place on the established path, the weak node fails, disrupting the data flow. Clearly, this greedy method of locally choosing the best neighbor may not find the best path. Although, this method provides sufficient load balancing compared to basic Directed Diffusion, it fails to find the best path every time.

To increase the probability of finding the best path, a better idea is to discover multiple paths and give the decision making to the source node to select the optimal path for routing. Multiple paths can be found by using multipath routing technique. Multipath routing has been used for distributing traffic along multiple paths between source and

sink nodes to incorporate load balancing [12]. In some multipath routing methods, an optimum path is chosen for data routing while additional paths are kept for fault tolerance purpose [13]. The proposed multipath protocol takes the approach of alternative path routing. Using this approach the protocol discovers multiple paths periodically and chooses an optimal path from the set of discovered paths for routing source data. By doing this, the protocol tries to select a path which would least likely result in nodes along the path depleting their energy during data routing.

The proposed multipath routing protocol has two variants. The first variant constructs multiple paths based on Cumulative Residual Energy (CRE) of the nodes along the path. The second variant constructs paths based on energy variance between nodes along the path. The method of multipath route construction based on CRE tries to construct multiple paths based on local decisions. Therefore, each node would reinforce the next-hop node which has the highest residual energy. Also, the method allows the source node to have information about the constructed paths. This information is the total residual energy (CRE) of the nodes along the path, and source selects the path which has the highest CRE value. The source node also updates the CRE value of the path used for data transmission, and dynamically switches traffic using alternative paths based on high CRE. By doing this, the source data is routed using a healthy path which has the least probability of failure during data transmission. However, path constructed based on CRE using local decisions may have a weak node (low in residual energy) on it. This node may fail during the data transmission phase, resulting in a partitioned and dysfunctional network.

To filter out such weak nodes from the constructed path, another method of multipath route construction was developed which constructs paths based on energy variance between nodes along the path. The method constructs multiple paths based on energy variance between nodes along these paths, and the source node is given this information. The source node selects the path which has the lowest residual energy variance between its nodes. The performance of the two proposed multipath routing protocols is compared in Chapter 5.

1.1 Thesis Statement

In this thesis, the focus is on the challenges involved with routing of data in Wireless Sensor Networks, and the development of an energy-efficient routing scheme. The thesis proposes the design of a multipath routing protocol which aims at maximizing network lifetime and minimizes energy consumption in the network. These goals are achieved by incorporating load balancing, energy based path construction, source based path selection, and a repair algorithm in the protocol.

1.2 Contributions of the Thesis

The main contributions of the thesis are as follows:

1. Multipath Routing Protocol: The method of multipath route construction is used to discover multiple (braided) routing paths. Paths are constructed either based on cumulative residual energy of the nodes along the paths, or based on the energy variance between nodes along the paths. The source node is provided the control to select the optimum path for routing data from the source to the sink. Source selects path based on the knowledge about the paths in hand. The probability of finding the best path increases

with the source being able to select a path from a set of good paths. The multipath protocol integrates route construction with route discovery, which helps in conserving energy. Paths constructed by the protocol are loop-free and have minimum latency (compared to looped paths). The constructed paths are also good quality paths (in terms of energy).

2. Cost Analysis of Multiple Paths: The thesis presents an analysis of energy cost associated with multipath route construction phase. Although the method of multipath discovery comes with a minimum overhead by forming braided paths (as opposed to disjoint paths), it is necessary to find the energy cost associated with constructing multiple paths and to choose an optimum number of paths to be used.

3. Load Balancing Algorithm: To incorporate load balancing a method of network wide energy updates is introduced. Each node receives an update about residual energy of its neighboring nodes and also sends its own value to them. This is done by "piggybacking" residual energy values onto the reinforcement packets and data packets. This method of energy updates can help in reinforcing nodes based on their residual energy. By selecting the next-hop neighbors which have the highest residual energy, load balancing can be achieved in the network. In addition to this network wide load balancing, a path level load balancing can be applied when the source dynamically switches traffic between multiple paths.

4. Reactive Repair Algorithm: During network operation some nodes may go down temporarily, e.g., to replenish their batteries. To recover from this situation, a reactive repair algorithm is proposed, in which the sink can detect missing events and initiate recovery by finding alternate paths. There is also a mechanism of removing nodes from

the routing path when they reach a certain threshold energy value (e.g., 2% of initial residual energy).

5. Visualization Tool: For our Java based simulator, a visualization tool is developed to evaluate the performance of the protocol. The software is a general purpose tool which can be integrated with any network simulator. This tool is used to analyze energy data from sensor networks and display it on an intensity map. This type of visualization tool can be used for (not limited to) evaluating the performance of energy-efficient routing protocols, debugging problems with such algorithms, visually tracking energy consumption in the network, visualizing path formation in the network, identifying failures in the network, and locating path merging (in case of multipaths). The software is developed in LabVIEW and was integrated with the Java-based network simulator to generate intensity maps and analyze network data (energy distributions).

6. Performance Comparison: The thesis compares the performance of the proposed multipath routing protocol with other protocols such as basic Directed Diffusion, load balanced directed diffusion, and multipath directed diffusion. These protocols are evaluated in terms of network lifetime, degree of load balancing, and residual energy variance.

7. A load balanced directed diffusion protocol was developed as a part of this thesis, and was published in [11].

8. A part of this thesis involves the development of a multipath routing protocol based on directed diffusion, which was published in [47].

1.3 Outline of the Thesis

The organization of the thesis is given as: Chapter 2 presents the related work for the proposed multipath routing protocol. Chapter 3 describes the operation of the proposed multipath routing protocol in detail. It also provides details about the two variants of the proposed protocol. In chapter 4, design and development of the Java-based simulator is explained. This chapter provides technical details, flow charts, and UML class diagrams for the simulation design. The design of the visualization tool in LabVIEW is also covered in this chapter. Chapter 5 gives the results from the experimental work. It explains the experiments design, presents the results, and analyzes the results. Finally, Chapter 6 provides the conclusions of the thesis; it also discusses the shortcomings of the proposed work and outlines future directions.

Chapter 2

Related Work

A routing protocol for WSN is influenced by the inherent characteristics of the network such as limited energy supply, limited communication bandwidth, distributed nature and physical changes affecting connectivity. Therefore, a routing protocol must be designed keeping all factors in mind. One of the most important considerations in this regard is the energy-efficiency of the routing protocol. Although a lot of work has been done to make sensor nodes energy-efficient, still their operational energy cost is high. This requires routing protocols to consider energy-efficiency and deliver data through energy-efficient routing, prolonging the lifetime of the network.

In this chapter, we begin by classifying multipath routing protocols and establishing motivation behind their use. In the end, issues germane to the proposed multipath routing protocol are discussed.

2.1 Multipath routing

Multipath routing is used to overcome the problems associated with single path routing. Single path routing is easier to implement (compared to multipath) and is widely used in wireless sensor networks. In a dense network, it is easy to find a single route between source and sink. Many single-path routing protocols are discussed in [14] [15]. Single path routing is considered energy-efficient [6] [7]. In some cases [11] it also incorporates load balancing, but not to the maximum. However, single path routing comes with many drawbacks. Multipath routing has following benefits over using single path routing:

- Minimizing traffic congestion
- Fault tolerance
- Load balancing
- Improving quality of service
- Reducing end-to-end delay
- Secure communication
- Reliability
- Reducing frequent route discovery

Multipath routing involves the discovery of multiple paths which are used for routing purposes. Multipath routing allows the source node to route its traffic towards the sink using any of several efficient paths. By using these multiple paths, alternatively or simultaneously, multipath routing takes off the load from a single path and distribute it across the network. Traffic switching between multiple paths allows communication even in case of failures.

Multipath routing has its roots in communication networks from early years. One of the first works in multipath routing was implemented by Maxemchuk in 1975 [16]. Multipath routing was used for store and forward data networks to disperse data over several paths. Multipath transmissions were found to achieve lesser average delay compared to single path transmissions. Multipath routing also reduced retransmission due to errors. It worked efficiently despite link failures, providing fault tolerance. These advantages of multipath routing encouraged the use of same idea in other types of data networks as well [17][18]. Multipath routing was also employed in wired networks to provide QoS routing, increase effective bandwidth between nodes, reduce congestion, and achieve reliable packet delivery as discussed in [19][20][21][22].

Multipath routing is also implemented taking inspiration from nature. Many bio-inspired algorithms are being designed analyzing the behavior of social insects (e.g., ants,

termites, wasps) to achieve intelligent routing. AntHocNet [23] and Termite [24] are two prime examples in this regard. These algorithms are used in wireless ad-hoc networks to gain benefits of multipath routing.

Multipath routing has been employed by WSNs to achieve reliable data transmission, resilience against path failures, load balancing, energy-efficiency, and quality of service for delay constrained applications. In the next sections, several multipath routing algorithms based on above mentioned features are discussed.

2.2 Types of Multipath routing in Wireless Sensor Networks

Multipath routing protocols for WSNs are broadly classified into two groups, i.e. alternative path routing and concurrent multipath routing as discussed in [25].

2.2.1 Alternative Path Routing

Alternative path routing uses discovered multiple paths one at a time for routing data between source and sink. Thus, each node in the network maintains information about alternative next-hop neighbors or paths towards the sink. An important phase in alternative path protocols is the path discovery phase. In this phase, the protocol discovers multiple paths between source and sink, and stores this information for future use. After multiple paths are discovered, it is at the discretion of the protocol to either use multiple paths (concurrent multipath routing) or a single path (alternative path routing). After path discovery, the source node chooses a single optimal path to route data towards the sink. However, other discovered multiple paths are not discarded and are kept as back-up paths. In case, when the primary path fails, these back-up paths are used to switch traffic between them and avoid service interruptions.

The primary objective of using multipath routing was to provide fault tolerance in WSNs. This is necessary because of the unreliability of a wireless medium. This objective is mostly achieved by alternative path protocols which incorporate fault tolerance from path (or node) failures [13]. Alternative path routing reduces the frequency with which paths discovery takes place. This is possible because of the availability of back-up paths. Alternative path routing, since it uses one optimal path at a time, does not have inter-path interference problems, as faced by concurrent multipath routing. Alternative path routing also considers energy efficiency [42][43]. One drawback with alternative path routing is the capacity limitation of the single path, which limits the throughput.

2.2.2 Concurrent Multipath Routing

After the path discovery phase, as described above, multiple paths can also be used simultaneously to route data towards the sink node. Concurrent multipath routing uses several multiple paths simultaneously to route data from source towards the sink. It was used initially to distribute the load across the network and reduce transmission delays [16].

Concurrent multipath routing is used to achieve reliable data transmission over unreliable wireless media [37][38][39]. Reliable transmission in a WSN is challenging [26] considering its characteristics such as wireless interference, low-power operation, and device failures. Protocols under this category use concurrent multiple paths to send either duplicate packets or sub-packets with error correction (erasure coding) to achieve reliable data transmission in a WSN. These protocols also address QoS and reliability issues associated with time-critical applications e.g., multimedia streaming, and disaster

warning. They perform QoS routing to ensure reliable data delivery with no loss and delay [40][41]. Thus, throughput can be increased using concurrent multipath routing.

Considering limited resources of a sensor node, it is necessary to balance the use of resources across the network [45]. Concurrent multipath routing tries to balance resources among the nodes. It implements load balancing in the network, making it energy-efficient. By splitting data between multiple paths, this technique also minimizes congestion which occurs in high rate applications.

A drawback associated with concurrent multipath routing is the interference between adjacent paths. For example, in a network using a single wireless channel, by concurrently transmitting data along multiple paths, the interference between paths (which are in the coverage area of each other) results in packet collisions and decreasing the desired throughput. This effect is called the route coupling effect, as pointed out in [27]. This problem is handled by using multi channel sensor nodes or by using zone-disjoint routing [46]. Another drawback with concurrent multipath routing is that it does not focus purely on fault tolerant routing. Also, distribution of traffic along multiple paths is implemented with some overhead, which is not present in alternative path routing.

2.3 Objectives for Multipath Routing

As discussed previously, multipath routing is used to provide features which are not accompanied by single path routing. This section presents multipath routing protocols which are specifically designed with an objective to incorporate load balancing, reliability, quality of service, energy awareness, secure communication, and interference minimization in WSNs.

2.3.1 Fault Tolerance

Fault tolerance in a WSN is a mechanism to recover from node or path failures. It adds resilience to the network against failures, such that when the primary path fails, the network could recover from this failure by providing alternate paths for service continuation. The main source of failure could be sensor node's energy depletion or a temporary outage. Unlike single path routing algorithm, where this failure would trigger costly discovery mechanism (e.g., flooding), multipath routing has a sophisticated recovery mechanism. Multipath routing avoids discovery process by maintaining back-up paths at a low energy cost. This would allow the network to conserve energy used for recovery. Multipath routing was primarily designed to achieve fault tolerant routing.

An important protocol with multipath routing capability is the Directed Diffusion [7]. This protocol was developed initially to provide single path energy-efficient routing. However, it can be used for multipath routing, as it can discover multiple paths. It has four phases, namely interest propagation, exploratory data propagation, reinforcement phase and reinforced data propagation phase. During the "*interest propagation"* phase the user can request for a named data by diffusing an interest within the network. An interest message is a query for the sensing task the user is interested in. An interest is usually a list of attribute-value pairs with different attributes e.g., interest type, location, and duration for an interest. An Interest message is flooded within the network and would eventually find a node that serves the task specified by the interest. This node becomes the source node. During interest propagation, whenever a node receives an interest message, it first checks to see if a matching entry exists in memory or not. If a node receives an interest it has never received before, it sets up a 'gradient' towards the

sending node. Gradients are used to draw events back towards the sink node. Interest message is sent out periodically by the sink in order to refresh the interest memory at each node. The source node starts the second phase (Exploratory Data phase) of the algorithm by sending exploratory data packet to all of its gradients (neighbors) for the received interest. Each node has an interest cache and also stores all the gradients associated with a particular interest. Exploratory data packets arrive at the sink node through multiple paths, and at this point the sink needs to reinforce (*Reinforcement* phase) only one path. The sink node does that by reinforcing one of its neighboring node that sends the exploratory message packet first, i.e., it establishes a least-delay path. Similarly the reinforced node reinforces one of its neighbors and eventually a path is reinforced. The source node then starts sending data at a higher rate through the reinforced path. Fig. 2.1 shows the Directed Diffusion operation. Whenever a node detects any degradation in the reinforced path, it negatively reinforces the established path and tries to establish a new path for routing. Thus, using low-rate flooding, the algorithm allows local repair around failed nodes. Directed Diffusion falls under the category of alternative path routing protocols since it can recover from path failures by discovering new paths. However, the original work in [7] does not implement multipath routing.

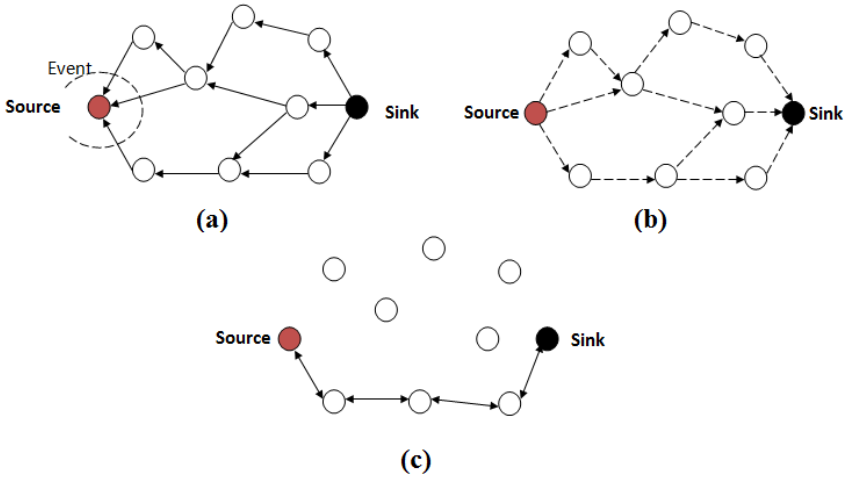

Fig. 2.1 (a) Interest Propagation, (b) Gradient setup, (c) Path reinforcement and reinforced data propagation.

Another data-centric protocol, which uses multipath routing to achieve fault tolerance, is described in [13]. This work implements multipath routing using directed diffusion. The protocol falls under the category of alternative path routing. This protocol uses multipath routing to discover alternative paths between sink and source nodes. The basic operation is similar to directed diffusion, where the sink node reinforces a primary path during the reinforcement phase. Unlike directed diffusion, this algorithm also constructs alternative paths around the primary path during the reinforcement phase. The sink node passes "primary-path reinforcement" message to its best neighbor (based on least-delay) and eventually the message is received by the source. Alongside this primary path establishment, the protocol also constructs multiple paths using "alternative path reinforcement" messages. The sink node passes the "alternate path reinforcement" message packet to its second best neighbor. This second best neighboring node passes this alternate reinforcement to its best neighbor and so on. Similarly, intermediate nodes on the primary path also generate this "alternate path reinforcements" towards their second best neighbor. By doing this alternate paths around primary paths can be

constructed. Whenever, a node on the primary path receives alternate path reinforcement, it does not propagate this further. The protocol manages to construct multiple braided paths. In case, when the primary path fails, these alternate paths are used to provide back-up for transmitting data from source towards the sink. Since the path discovery process can be expensive considering the requirement of network wide flooding as in directed diffusion, the protocol provides fault tolerant routing with minimum cost. The protocol avoids expensive flooding operation for path discovery as done by the original directed diffusion. The protocol maintains the discovered multiple paths by sending low-rate keep alive data through them. This work also compares braided paths with disjoint paths. It proposes the use of braided multi-paths since they are constructed with minimum overhead and provides a short length path between sink and source nodes. The algorithm is tested with isolated and patterned failures and proves to be robust and highly resilient against failures. A drawback of such a braided multi-path approach is that a failure of one node may result in a failure of multiple paths using that node. Also, the capacity limit of a single path is a problem.

2.3.2 Secure Communication

WSNs due to node's resource limitations (limited computation and energy) can be attacked from different sources. Thus, the security of a WSN is a key concern and must be addressed properly. There are different sources of attacks in a WSN. An attack can be from a passive attacker which tries to access unauthorized data. Such an attacker would only monitor the data but would not modify it. A major attack can be from an attacker who could emulate the role of a sensor node in the network. Such an attacking source could launch a serious attack by modifying the data, drawing event data, and dropping it,

generating excessive flooding in the network to compromise energy resources, or stealing identities of multiple nodes. A detailed description is provided by [28][29]. These attacks are countered by using cryptographic encryption algorithms. There are two types of cryptographic encryption schemes i.e., symmetric and asymmetric encryption. Symmetric key cryptography is found to apply on WSNs [30], whereas asymmetric key cryptography is not efficient for resource constrained WSNs [31].

Multipath routing is used to incorporate security in routing. An efficient concurrent multipath routing algorithm is discussed in [32]. The algorithm is based on curve based greedy routing algorithm [33]. The algorithm adds an encryption algorithm on top of multipath curve based routing to make it secure. By using curve based greedy routing, nodes can select their next-hop neighbors greedily. The algorithm works by having location information about the sink and a source. The source node constructs curved trajectories (B-spline curve) and incorporates information about these trajectories into the forwarded packets. Once these packets are received by intermediate nodes, they extract the information inside the packets and create a dynamic forwarding table (DFT). Intermediate nodes then forward the packets by greedily selecting a neighbor along the curve based on DFT information. Also, a node along the curve can ask a nearby node to take up the task of relaying packets, implementing path switching. The routing algorithm encrypts the forwarded packets with different encryption keys (sharing keys). Each node in the network is assigned a master key. Packets are forwarded along multiple paths to incorporate reliability and load distribution features. When these packets are received from multiple paths, the sink node compares these packets to check whether any packet was modified or not. This helps in tracing out an attacker. The algorithm achieves

security, reliable transmission, and load balancing using concurrent multipath routing. Results show that the algorithm can be implemented with low complexity. One drawback of this protocol is that it does not aim for achieving energy-efficiency.

Another secure multipath multi-base station routing is discussed in [34]. The algorithm addresses the security challenges faced by the base station, where all the data from the sensor network is collected. A base station can be attacked by a denial of service (DOS) attack or a rushing attack that tries to isolate the base station from the sensor network. The algorithm solves this issue by providing multipath redundancy in the network. It uses the concept of having multiple base stations, such that traffic is distributed to these base stations through multiple paths. These redundant paths are setup to block the attacks which could disrupt traffic routed through one path towards one base station. The algorithm constructs a tree-like network structure. Each base station floods the network with a REQ message. The REQ message is broadcasted only once by each node. Each node maintains one path towards each base station. To counter a forged REQ message which may be sent by an attacker, one-way hash chain is used to authenticate message integrity. The algorithm also provides a solution to rushing attack in a multipath network using an echo-back algorithm. A rushing attack is when an adversary node transmits with high transmission power to compromise the hash chain method. To counter this problem, nodes verify their authentic neighbors using echo-backs. The idea behind using echo is that an attacking node is far away from authentic node's transmission range and may not hear it. So once a node receives a REQ message it sends an echo message back to the sending node and waits for a feedback. If a feedback is provided, the node enters this sender node in its 'verified neighbor list'. Thus, nodes in

the network can identify a rushing attacker and isolate it. The paper also presents an analysis to disguise the location of the base station, avoiding attacks. The algorithm is a good example of implementing concurrent multipath intrusion tolerant and fault tolerant routing. A drawback of this algorithm is that it survives only a few types of attacks, and for other types the security may work to some extent. Another problem with this scheme is that its operation is costly because of redundancy.

2.3.3 Reliable Transmission

Reliable data transmission requires efficient end-to-end data delivery. It is achieved by multipath routing, employing data distribution techniques with erasure coding. Most of the protocols use concurrent multipath technique to achieve reliable communication. Protocols under this category acquire reliability information within the network and then select optimal paths for routing.

A simple approach to enhance the reliability of a WSN is to send duplicate copies of a message over multiple paths. Although this is easy to implement, this technique generates a lot of traffic. In [36], a trade-off between reliability and traffic has been discussed. The work proposes a method of splitting the data packet into sub-packets and sending them along multiple paths. The number of sub-packets is same as the number of disjoint multiple paths. The total number of sub-packets is a function of the multipath degree and failing probabilities of these paths. It is shown that even if some of the sub-packets were lost, the original data packet could still be reconstructed.

Another protocol which applies the concept of packet duplication is discussed in [37]. This protocol tries to achieve reliability in a multi-hop sensor network with high channel errors. In ReInForm (Reliable Information Forwarding using Multiple Paths), the

source node before sending its data, measures for the data transmission reliability using the collected data. The source generates a packet and defines its criticality i.e., how critical the sensed information is for the sink. Based on this criticality the source computes its reliability and puts this information in the packet header. The source or the forwarding node computes and fills the data packet with information e.g., error rate of channel, number of paths to create, and hop count in its Dynamic Packet State (DPS) fields and sends multiple copies of this data over multiple paths. The number of multiple paths is decided by the source based on the degree of reliability measured from the collected information. When this data packet with DPS fields is received by an intermediate node, it looks at the information in the DPS fields and decides the number of duplicate packets it should forward to its neighbors. Eventually, the packets reach the sink and reliable communication is established. The paper also discusses the use of acknowledgements from sink to the source for reliable delivery. It shows that by using acknowledgement in a network with high errors and hop count, acknowledgements from the sink towards the source for data reception are lost. When the hop count and errors are low, overhead in using acknowledgement is high. The paper concludes that the scheme which does not use acknowledgements is better. Although, ReInForm promises reliability of data transmission, it is not energy-efficient because of duplicate message forwarding. It requires high energy resources for its operation.

N-to-1 Multipath routing protocol is discussed in [38]. This protocol is a hybrid multipath algorithm which combines both concurrent and alternate path routing. The protocol constructs multiple node-disjoint paths from every node towards the sink. It uses concurrent multipath routing to distribute data. It also uses alternate multipath routing at

each node. Since each node has multiple disjoint paths towards the sink, these multiple paths can be used alternately, to add reliability to the data delivery process. The protocol uses one discovery phase to discover multiple node-disjoint paths from all the sensors towards the sink. The discovery process is divided into two stages. During the first stage the protocol uses simple flooding triggered by the sink node to construct a spanning tree. This stage is called the "branch aware flooding". The sink node broadcasts a "route update" message first. If an intermediate node receives this update, it sets the sender as its parent node and rebroadcasts the route update to its neighbors. The branch aware flooding constructs a basic spanning tree with each node having a path towards the sink. The second discovery stage is called the "multipath extension flooding". This stage is used to find multiple paths towards the sink from each node. It constructs links between two nodes from different branches of previously constructed spanning tree. In this way, these two nodes can have additional paths towards the sink. Nodes during multipath extension flooding exchange information among each other about node-disjoint paths discovered previously. After the construction of full routing tree (Fig. 2.2(b)), source nodes split data along discovered multiple paths.

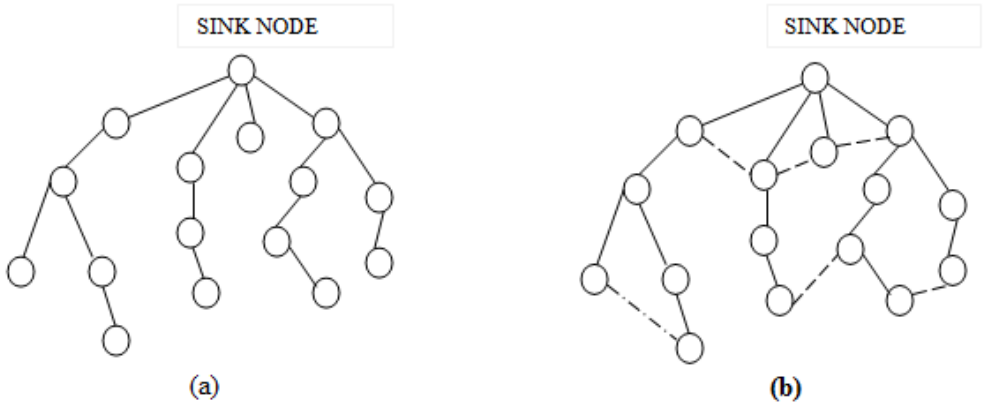

Fig. 2.2 (a) Basic spanning tree, (b) Fully constructed tree by extension flooding.

The protocol provides secure and reliable data transmission. However, it does not provide security to each node. It also benefits from per-hop packet salvaging strategy, in which for an unsuccessful transmission to the next hop node the packet is not dropped but is re-routed through another randomly selected node. The protocol uses simple flooding mechanism for path discovery which does not require special rules and control packets. However, such a flooding scheme does not guarantee good quality paths. The protocol also does not consider interference minimization, which is required for concurrent multipath routing, degrading network performance. An extension of this work is presented in [39]. The new protocol uses N-to-1 multipath route construction with secret threshold sharing scheme to provide resilience against node or path faults.

2.3.4 QoS Routing

QoS routing is important for applications which require high throughput and minimum latency. The "QoS based protocols" uses a quality of service metric (e.g., latency) to ensure energy-efficiency and reliable data delivery. For time critical applications, QoS routing may use multiple paths to achieve desired capacity. It also distributes traffic based on the QoS requirements. For example, multipath routing is used to send delay-sensitive packets on a high capacity path, and delay-insensitive packets using paths with lesser capacity.

Delay-Constrained High-Throughput Protocol for Multipath Transmission (DCHT) proposed in [40], is a concurrent multipath protocol. It uses Directed Diffusion [7], for high-quality multimedia streaming applications. The protocol uses erasure coding to add reliability to data transmission. The idea is to estimate Expected Transmission Count (ETX) which is a link layer metric to find high-throughput paths. The protocol

constructs paths which can maximize throughput and minimize latency. It begins with network wide interest flooding. The sink node tags the interest message with a time stamp, which is used to measure latency in the network. Once the source node with specified data is found, exploratory data is sent from the source towards the sink. Each intermediate node receiving exploratory data calculates the cost of transmission over the path from which the exploratory data is received. This cost is calculated using cost functions given by:

$$Path\_Cost = Path\_ETX^{\propto} \, x \, Path\_Delay^{\beta} \tag{2.3.1}$$

$$Path\_ETX = Max_{i=0}^{N-3} \left(\sum_{j=i}^{i+2} ETX_j \right) \tag{2.3.2}$$

Where N is the number of hops in the path, and ETX_j is the ETX of the jth hop. After cost calculation, the node which received the exploratory data broadcasts the lowest cost to its neighboring nodes. During the exploratory data propagation phase, each node estimates the ETX value of its neighbor by reading SNR from the received packet. Eventually, exploratory data packets reach the sink node through multiple paths. The ETX value of the discovered paths is used by the sink node to choose optimal paths with minimum interference. The sink node reinforces multiple node-disjoint paths with minimum end-to-end delay. After source node is informed of the reinforced paths, the source node encodes the traffic into multiple sub-streams using Multiple Description Coding (MDC) algorithm. These streams are sent independently from the source towards the sink using multiple reinforced paths. A problem with this protocol is the exact estimation of interference, which is not accurate. Also, multiple streams require sufficient number of node-disjoint paths to avoid interference, which is difficult to discover considering random deployment of nodes.

Another QoS based concurrent multipath protocol is presented in [41]. Energy-Efficient and QoS-based Multipath Routing Protocol (EQSR) is designed to provide reliable data transmission for real-time applications. The protocol starts operation by the broadcast of a HELLO message by each node. This broadcast help nodes to obtain transmission cost information from their neighbors. Next, the sink node passes the Route-request message to its best neighbor based on equation:

$$\text{Next hop} = Max_{y \in N_x}\{\alpha E_{resd,y} + \beta B_{buffer,y} + \gamma\, I_{interference,xy}\} \qquad (2.3.3)$$

Where N_x is the set containing node x's neighbors, $E_{resd,\ y}$ is the residual energy of neighbor node y, $B_{buffer,\ y}$ is the buffer size of neighbor y, and $I_{interference,\ xy}$ is the SNR for the link between x and y. These values are calculated by each node during the HELLO broadcast stage. The sink node constructs multiple paths by passing successive Route-request messages to its next best neighbors. Based on the propagation delay of Route-request message, the protocol groups L best paths for delay-sensitive data transmission, and keeps other discovered paths for delay-insensitive applications. EQSR adds reliability to data delivery by calculating error correction codes for the data using a light weight XOR-based Forward Error Correction (FEC) algorithm. The source node then sends the data through selected multiple least-delay paths. EQSR is a reliable protocol which aims at minimizing transmission delays. A drawback of this protocol is the high control packet overhead involved with the FEC algorithm which is used to ensure reliability. Also, interference estimate over a link using flooding may not be accurate. This results in construction of paths which do not have minimum interference.

2.3.5 Energy Awareness

Multipath routing protocols must be energy-efficient to prolong the lifetime of the network. Normally, energy-efficiency is achieved by using alternative path routing. A few protocols in this regard are presented in this section.

A protocol that achieves energy-efficiency using a single service path while keeping a backup node-disjoint path is discussed in [42]. The protocol floods the network with a "service-path request" message to discover a service-path. The source node acknowledges the discovery of service-path by sending a "service-path reservation" message towards the sink. Each intermediate node receiving this message reserves some portion of its residual energy for transmission over this path. This helps in preventing failures during transmission. After path reservation, data starts flowing from the source towards the sink. The protocol after constructing a single service path enters another discovery phase to construct a backup path. The main drawback associated with such an alternative path routing scheme is the capacity limitation of a single path. The protocol constructs energy efficient multiple paths but does not implement load balancing, which may lower its energy-efficiency.

Energy-aware routing [43] is a reactive routing protocol similar to Directed Diffusion [7]. It uses alternative path routing to achieve energy-efficiency. The protocol consists of three phases, namely setup or interest propagation phase, data propagation phase, and route maintenance phase. During the setup phase, a request is flooded in the network by the sink node. This request message is used to discover possible paths between source and sink nodes, and to find energy cost along each path. Each node would forward the request message to its neighboring node which is closest to the source

node and farther away from the sink. The request message carries the total path cost. Each node receiving the request message computes the cost between itself and the sending node and appends this value to the total cost. This cost is then forwarded through the request packet to the next-hop node. When a node (node j) receives a request packet from node i, it computes the energy metric for the local link (between node *i* and *j*) as:

$$C_{ij} = e_{ij}^{\alpha} R_i^{\beta} \quad (2.3.4)$$

Where R_i is the residual energy level of node *i*, e_{ij} is the transmission energy required between link i and j, and α and β are the weighting factors. If node N_j receives a request from node N_i, it calculates the cost of the path as:

$$C_{N_j, N_i} = Cost(N_j) + C_{ij} \quad (2.3.5)$$

Node N_j only stores neighboring paths with low costs in its forwarding table and high-cost paths are discarded. Only the neighbors N_i are added to the forwarding table FT_j of N_j.

$$FT_j = \left\{ i | C_{N_j, N_i} \leq \propto \left(\min_k C_{N_j, N_k} \right) \right\} \quad (2.3.6)$$

Node N_j also assigns a probability to each neighbor in the forwarding table.

$$P_{N_j,N_i} = {1/C_{N_j,N_i}} \Big/ {\sum_{k \in FT_j} 1/C_{N_j,N_k}} \quad (2.3.7)$$

Thus, each node (say N_j) finds a number of neighbors through which data can be routed towards the destination. The Source node then sends the data packet to any one of the nodes in its forwarding table by randomly selecting it with a probability. Similarly, each intermediate node would select a neighbor at random and forward the data packet which eventually reaches the sink node. Energy aware routing maintains multiple paths for routing and uses an optimal path each time a data is to be routed from the source towards

the sink. The protocol also incorporates load balancing by allowing each node to select its next-hop neighbor randomly from a set of good neighbors. This balancing of load across the network prevents burdening of a single path. A drawback for this scheme is the capacity limitation of a single path. An extension of this work that adds energy-awareness in directed diffusion is discussed in [44].

2.3.6 Load Balancing

Most routing protocols designed for WSNs earlier, uses single path for routing data between sink and source nodes. Single path routing makes the nodes along the used path to consume more energy and deplete energy at an early stage. This makes network to die prematurely, limiting its lifetime. To solve this problem, load balancing needs be incorporated so that the data is transmitted using multiple paths. The distribution of data through multiple paths helps in sharing the work load in the network. This further allows nodes to consume energy at the same rate, and increase network lifetime.

A multipath routing protocol which incorporates load balancing is discussed in [45]. The protocol aims for maximizing network lifetime by implementing a load balancing algorithm for data distribution. At the start of the protocol, each node would broadcast HELLO and CONNECTIVITY messages, to update its neighbor table and identify sink nodes. Each node also has information about the energy levels of its neighbors, exchanged via HELLO packets. The source node after detecting an event starts the route discovery process by sending multiple REQUEST messages. These messages construct multiple node-disjoint paths towards the sink. Intermediate nodes select their best next-hop neighbor, neighbors not used for any other path, using expression:

$$\arg min_{b \in N_a} \{ (1 - \frac{e_{b,residual}}{e_{b,init}})^{[\frac{\beta(1-(\Delta d+1))}{d_{ay}}]} \} \quad (2.3.8)$$

N_a is the set of a's neighbors, d_{by} is the distance between node b and sink, d_{ay} is the distance between node 'a' and sink, Δd is the difference between d_{by} and d_{ay}, $e_{b,init}$ is the initial energy level of node 'b' and $e_{b,residual}$ is the residual energy level of node 'b'. The sink node after receiving the first REQUEST packet sets a timer and accepts only high-quality paths which are discovered before the timer expires. Other discovered paths are discarded by the sink. The load balancing algorithm is used to allocate traffic rates along multiple paths. Sink node allot data rates to the discovered path using expression:

$$r_j = \frac{R}{P_j} \sum_{i=1}^{N} P_i \, , j = 1,2, \ldots . , N \quad (2.3.9)$$

This expression is computed from load balanced ratio for multiple paths, where R is the requested rate, 'r_j' is the allotted rate. The protocol constructs multiple paths and uses them concurrently to distribute data between source and sink. By considering the residual energy levels of nodes and hop count towards the sink, the protocol incorporates load balancing along with energy-efficiency. The protocol does not consider interference affects which may lead to degradation of network performance.

2.3.7 Interference Minimization

When multiple paths are used concurrently, then transmission of nodes on one path may interfere with the transmission of nodes on the adjacent paths. This results in packet collision at the nodes and network performance is degraded. To counter this problem, multipath protocols are designed specifically for interference minimization.

Energy-Efficient and Collision-Aware Multipath Routing Protocol (EECA) is presented by [46]. The protocol constructs two collision free paths between sink and

source nodes. These two paths are constructed carefully by the sink, and are at the extreme ends of the line joining source and sink. The paths are constructed such that the distance between them must be greater than the interference range of the nodes. The process is started by the source node which finds two neighbor sets on both sides of line between source and sink. The source node, then, broadcasts a route request packet towards these nodes to construct two zone-disjoint paths. Intermediate nodes also use the same procedure to select the best neighbor and broadcast the route request packet towards the sink. Request packet is broadcasted by the selected node only to its neighbors. If any node overhears a transmission from a node on the other route, this overhearing node should not be a part of any route. The protocol also controls the transmit power of each node on the route to avoid collisions. Each node which receives the route request message starts a back-off timer based on the residual energy and distance towards the sink. This timer is used to avoid collision and overhead involved. The sink node when receives the route request packet sends a "route reply" message towards the source node, which then establishes a path. The route reply message contains the total route energy, and the identity and residual energy of a node with least residual energy on the route. Once the "route reply" reaches the source, it stores the energy updates, and select all available paths for data transmission. Source node tries to balance the traffic load by sending data along each constructed path according to path's energy information. The protocol minimizes collisions by constructing two paths physically apart and controlling transmission power. Power control also helps in energy savings. However, the capacity of traffic is only limited to two collision free routes. While constructing collision free

routes, the distance the two routes should not be large, which would then cause longer routes to be formed and more energy consumption.

2.4 Design Considerations for Multipath Routing

The method of multipath route construction proposed in this thesis uses alternative path routing to incorporate load balancing, energy-efficiency, and resilience against path failures. The proposed algorithm uses alternative path routing to avoid interference between adjacent paths which leads to packet collisions at the node. The method constructs multiple routing paths and use only one for routing data from the source towards the sink. These multiple paths are constructed by taking local decisions at each node, reinforcing next neighbor which has the highest residual energy. As discussed in [11], such a local decision based on residual energy proves to be greedy. Since a node does not have global view of the network, it cannot make an optimal decision. The work in [11] uses single path directed diffusion to incorporate load balancing, but does not guarantee the optimality of the reinforced path. The proposed algorithm takes the approach of constructing multiple paths by locally reinforcing next-hop neighbors having the highest residual energy. To solve the problem with such a greedy approach, source node is allowed to investigate the optimality of these paths and deliver data using an optimized path. The path selected for routing should have the least probability of failure while data is being routed between source and sink. Another problem with local decisions to reinforce a path is that the constructed path only consider energy-balancing and would not try to make the path length short. So the constructed path may not be the shortest path and would not consider delay constraints. Thus, there is a trade-off between energy based longer paths and delay.

One important design consideration linked with multipath routing is whether to construct braided multipath or node-disjoint multipath (Fig. 2.3). A braided path is a path which can share nodes with other paths, whereas a node-disjoint path does not. The proposed protocol constructs paths by selecting next-hop neighbors based on residual energy. This may possibly construct braided paths. Braided paths are energy-efficient as they do not require extra handshake messages (e.g., negative reinforcements) as required by node-disjoint paths [13]. Therefore, the cost of constructing braided paths is low. Considering the random deployment of the nodes, the connectivity of some nodes may not be high. This makes braided paths more suitable, as node-disjoint paths require high connectivity in the network. A drawback with node-disjoint paths is the delay required to construct them [13]. They are constructed with extra energy cost. Secondary node-disjoint paths are also longer in lengths compared to the primary path. For braided paths, multiple paths braid around the primary path having almost similar lengths, and the energy cost of discovering them is almost the same as the primary path. An advantage with node-disjoint multipath is that even in case of primary path node failures, it is ensured that alternate paths are not affected. In case of braided paths, any failure on the primary path could affect other alternate paths braided through it.

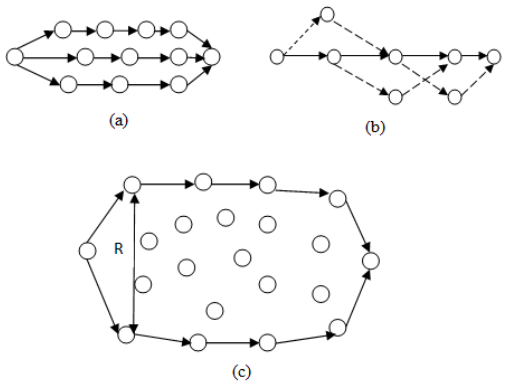

Fig. 2.3 (a) node-disjoint paths, (b) braided paths, (c) zone-disjoint paths.

To provide resilience against node failures (path failures), a reactive repair mechanism can be incorporated in the proposed protocol. To keep things simple, the sink node is allowed to detect a missing event and then initiate the repair mechanism. This is to avoid extra cost associated with negative reinforcements used to truncate a corrupt path and establish a new one, as implemented by the original Directed Diffusion [7]. Also, nodes are set with energy thresholds, such that a node which is near its death (e.g., 2% of total energy remaining) would not be allowed to participate during the reinforcement phase. The rationale behind using this approach is to avoid the use of a weak node for path establishment and to prevent situations where a node having some data to send fails. Also, the method of multipath routing provides resilience against path failures by not burdening a single path.

To make better routing decisions, one consideration is to hand over routing path selection to the source node. The source node needs information about the conditions of the constructed paths to make a better decision. One parameter which could provide information about the path condition is the total residual energy of the constructed path. This parameter would tell the source node about the health of the constructed path. By selecting a healthy path each time for data delivery, source ensures reliable communication.

To achieve maximum network lifetime and minimum energy consumption at each node, load balancing could be applied by establishing a system of network wide energy updates [11]. By reinforcing the next neighbor which has the highest residual energy, the proposed method constructs different paths each reinforcement phase. This helps in

balancing the load in the network. Also, the source decision in selecting the optimal path among multiple paths would incorporate a path-level load balancing.

2.5 Summary

In this chapter, classification of multipath routing protocols into two main categories is presented. Protocols in the first category uses alternative path routing to achieve fault tolerance, load balancing and energy aware routing. The second category uses concurrent multiple paths to distribute traffic for load balancing, provide reliable data transmission considering QoS parameters, and apply interference minimization techniques. This chapter also categorizes these protocols according to their objectives. These routing protocols are explained briefly highlighting their advantages and disadvantages. The proposed method of multipath route construction is based on alternative multipath routing. Design considerations for the proposed multipath routing scheme are also discussed in this chapter.

Chapter 3

Multipath Routing Protocol

This chapter describes the proposed multipath routing algorithm in detail. The first section describes the network architecture, different definitions and assumptions used for it. In the next section, the proposed multipath routing algorithm with its different phases is explained in detail.

3.1 Network Design

A dense wireless sensor network is considered with N nodes having same capabilities. Sensor nodes are deployed in a random fashion over a square area (A). The network connectivity is set to be high, allowing no partition to exist in the network. This is done by increasing the radio range of each node such that it can cover a large number of nodes. Thus, each node would have a large number of neighbors. The sink and the source node can be placed randomly anywhere in the network. However, for the intensity maps (discussed in the next chapter) sink and source nodes are placed on the extreme ends of the square grid. This is done to have traceability of the sink and the source node on the map to track paths in between them. The network is considered flat with homogeneous sensor nodes. Based on application requirements, nodes can have different types of sensor (e.g., temperature, humidity, and strain) on board to detect an event. Each node has a configurable radio range of "r" meters. A path constructed between source and sink nodes can have n nodes on it, where $n < N$. The radio range r of a node is assumed to be short and the network would employ a multi-hop communication for data dissemination. A battery is used by the sensor node as an energy source. It is assumed

that nodes cannot replenish their batteries due to unavailability of a power source. Also, considering hazardous environments a WSN is deployed in, it is assumed that the battery of a node cannot be replaced. The network can support multiple sources and sinks.

The energy model for the proposed protocol is explained in chapter 4. The energy cost of transmission or reception electronics is assumed to be 50 nJ/bit [8]. The residual energy of a node is the remaining energy for its operation defined as $energy_{resd}$. This residual energy value is used by a node to choose its next best neighbor having the highest residual energy. There exists multiple paths between sink and source nodes which are discovered by the proposed algorithm. By locally constructing paths, the protocol could possibly construct braided paths.

3.2 Multipath Routing Algorithm

In this section the proposed multipath routing protocol is discussed. The protocol has two variants which are also discussed in detail. The proposed protocol has three main phases:

1. Query Propagation
2. Multipath Route Construction
3. Data Transmission and Path Selection

The protocol allows the Source node to select a path for data transmission.

3.2.1 Setup Phase

During the network start-up phase, the network enters the neighbor discovery mode. During neighbor discovery, each node would determine the presence of its neighbors. Neighbor discovery is an important process where each node despite its low-power operation registers all nodes in its range '*r*'. It is assumed that a node has

bidirectional links, i.e. it can transmit and listen to its neighbors. Each node during this phase broadcasts a discovery message to get registered with its neighbors and vice versa. Each node would discover its one-hop neighbors, and maintains a *Neighbor_List* which has identities of its neighbors. Radio range (r) for each node is set such that it allows the node to have a high-connectivity and connect with maximum number of neighbors. This is to discourage partitions in the network.

3.2.2 Query Propagation

Once the nodes in the network have information about their one-hop neighbors, the network enters into the second phase called the "Query Propagation". During this phase, *each node in the network gets residual energy information about its neighbors, and about the query the user is interested in*. The sink node is the node from where the user can access the network. A user can connect to any node in the network. A user having a query about particular data, floods this query within the network. The query packet has fields shown in Fig. 3.1. The query packet can also be appended with some other fields based on application requirements.

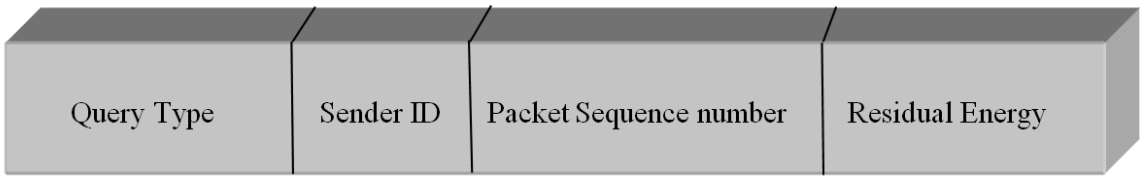

Fig. 3.1 Query Packet.

In the query packet, the "Query Type" defines the type of data the user is interested in. This can be a specific event e.g., sensing temperature, humidity, or detecting a disaster like fire, or an earth quake. The proposed protocol is a data-centric algorithm in which a query for a particular event is in the form of a named-data. Thus, a query for temperature

reading could be sent by putting "*Temperature"* in the "*Query Type"* field (Fig. 3.1). Finer details about the event can also be specified in the query such as:

Type= temperature // type of event requested
Sample rate = 0.5 s // sensor sampling rate
Coordinates = (22, 10) // node location
Expiry time = 00:02:00 // query expiry time

Each sensor node supports a particular type of named-data and would respond to the query packet if the packet type matches node's type. Each node has a *Query_Table* which stores information (query type, sequence number, and expiry time of the query) about the queries received by a node from its neighbors. The query in the *Query_Table* needs to be updated over time; otherwise it expires after some fixed time and is purged out of the *Query_Table*. The sink node broadcasts the query packet by putting its node identity in the *"Sender ID"* field of the query packet. It also generates a fresh *"Packet Sequence number"* for the query packet, every time a query packet is being sent. This sequence number is used to refresh the query in the *Query_Table* of a node. A node receiving a query packet with new sequence number updates the expiry time of the query in its *Query_Table*. An important field in the query packet is the "Residual Energy" field. This field is used by the node to convey its residual energy value to its neighboring nodes. Considering the broadcast nature of radio transmission, the idea is to allow neighboring nodes to snoop the query packet from their neighbors and extract this residual energy update. The sink node sends this query packet to all neighboring nodes on its *Neighbor_List.*

When a node receives the query packet, which it has not received previously, would make a new entry in its *Query_Table* (query type, sequence number, and expiry

time of the query) along with a gradient for the query. For each query, the corresponding *Gradient_List* stores the identity of the sender node which sent the query, and its residual energy value, which are extracted from the query packet. Thus, *every query has an associated Gradient_List*. The *Gradient_List* is used by a node to select a neighbor based on residual energy. The receiving node further broadcasts this query towards its neighbors on its *Neighbor_List*, putting its own identity and residual energy values into the query packet. An important point to mention here is that the proposed protocol unlike other directed diffusion based protocols [11] uses the Query Propagation phase of the protocol to piggyback residual energy updates. This would allow removing a separate phase for the energy updates, conserving considerable energy. This energy saving by removing a protocol phase would also balance the energy cost associated with flooding of queries in the network. This extra phase used by directed diffusion based protocols is the Exploratory Data phase, which is helpful in finding the least-delay path and updating energy of nodes in the network. Since the proposed protocol construct paths based on residual energy, this type of phase is not required. A representation of a query in the *Query_Table* with associated *Gradient_List*, and *RCP_List* is shown in Fig. 3.2. The *RCP_List* is used in the Route Construction Phase.

When a node receives a query packet for a query which already exists in the *Query_Table*, the node would check for the "Sender ID" field in the query packet. If the sender is new, the "Sender ID" and "Residual Energy" are stored in as a gradient for that specific query. If the sender already exists in the *Gradient_List*, the node updates the residual energy value for the sender in its *Gradient_List*. The node then checks for the sequence number of the query packet. If the sequence number is new, the node updates

sequence number in its *Query_Table* and broadcasts the query packet to its neighbors on its *Neighbor_List* by piggybacking its residual energy and node identity on it. Otherwise, the packet is dropped by the node to suppress excessive flooding.

The sink sends out query packets periodically (with a new sequence number each time) to maintain the freshness of the query, if the data requested is still required by the sink node user. The sink node also starts a timer (5s) to arrange for the periodic query transmissions. Eventually, all nodes in the network will have residual energy updates about their neighbors. Also, each node in the network will now have information about the query the user is interested in, and supports routing data for this query.

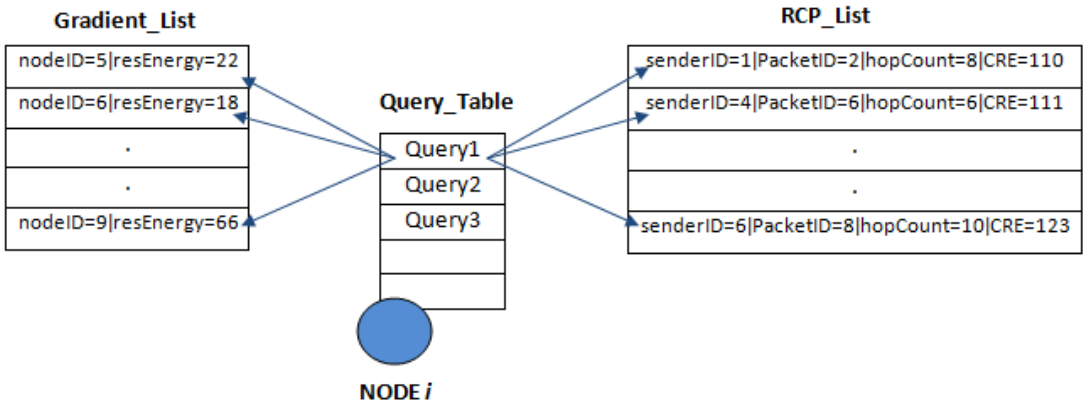

Fig. 3.2 Node i's tables for sorting queries.

3.2.3 Multipath Route Construction

After the query has been propagated, the sink node waits for a predefined time (5s) and then checks its *Gradient_List* for updates. Once the sink has energy information about all of its neighbors in the *Gradient_List*, it begins the *Multipath Route Construction* phase. During this phase, the sink node constructs loop-free high-quality multiple paths between source and sink nodes. *To construct multiple paths, the sink node sends Route Construction Packets (RCPs) to all of its neighbors, while intermediate nodes pass the RCP to one neighbor having the highest residual energy*. After constructing multiple

paths between sink and source, the protocol allows the source node to select an optimal path (having least probability of failure during data transmission) from the set of constructed multiple paths. However, unlike other protocols [13] which use already constructed paths, the proposed algorithm constructs multiple paths each *Multipath Route Construction* phase. In this way, the algorithm keeps searching for different paths during its operation.

The proposed method has two variants. One method constructs paths based on Cumulative Residual Energy (CRE), and the other one constructs them based on energy variance of the nodes along the path. Details of these methods, for the multipath route construction phase are described in the following subsections.

3.2.3.1 CRE Based Route Construction

For the CRE based method, the sink creates multiple Route Construction Packets (RCPs) per query, each with the fields shown in Fig. 3.3.

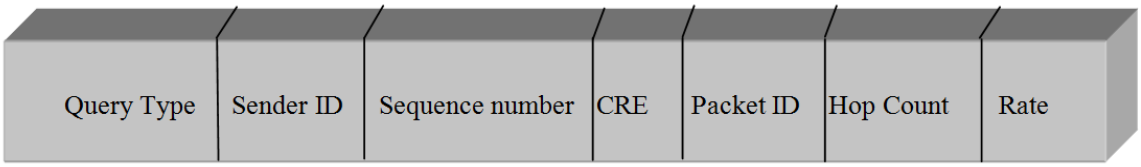

Fig. 3.3 Route Construction Packet for the CRE based method.

These RCPs are sent by the sink to its immediate neighbors on its *Neighbor_List*. To each neighbor, the sink would send a separate RCP with a unique packet ID. Some important fields in the RCP are Query Type, SenderID, CRE, Packet ID, and Hop Count. The "Query Type" field is used by each node to see if the node has received the query packet previously, and to access its *Gradient_List and RCP_List* for the particular query. *Each node has an RCP_List per query* (Fig. 3.2), which is used to store route construction

packet information (CRE, SenderID, PacketID, HopCount) at each node. The "Sender ID" field contains the ID of the node which sends the RCP. This "SenderID" is used in the next phase of data transmission to track back the constructed route. Each node receiving the RCP would store this sender identity into its *RCP_List*, so that it can send back the Data Packet towards the node with this identity, and towards the sink (details in next section). The "Packet ID" field has the RCP ID. To construct multiple paths towards the source, the sink node creates a separate RCP for each of its neighbor and gives it a unique "Packet ID". The "CRE (Cumulative Residual Energy)" field is used to carry the total route residual energy value towards the source node. The "CRE" value is used by the source to select an optimal path for sending its data towards the sink. The selection process is explained in the next section. "Hop Count" field is used to carry the total number of hops taken from the sink towards the source.

The sink generates multiple RCPs with a new "Sequence number" (each route construction round), assigning each RCP a unique "Packet ID". It puts its residual energy value into the "CRE" field and sets the "Hop Count" field as 'one'. The sink adds its identity in the "Sender ID" of each packet, and specifies the "Rate" it wants the data to be received at. The sink then passes these RCPs to its neighbors on its *Neighbor_List*.

When the RCP is received by an intermediate *node i*, the node checks the "Query Type" in the packet and finds a match in its *Query_table*. If the query exists, node *i* makes an entry in the associated *RCP_List* using the information in the RCP. Thus, *node i* stores the Packet ID (copied directly from the received packet), CRE value (after adding receiving node's residual energy value with packet value, equation (3.2.1)), Hop Count value (copied directly from the packet), and the corresponding Sender ID (copied from

the packet) (Fig. 3.2). This information is used in the last phase to track the constructed path and route data towards the sink. The "Rate" field is used by the source node to send the data at the specified rate towards the sink. *Node i* modifies the received Route Construction Packet by adding the "CRE" field with its own residual energy value. For an intermediate *node i*, this value is computed as:

$$CRE = CRE + energy_{resd,i} \tag{3.2.1}$$

Node *i* increments the hop count value in the "Hop Count" field. This hop count value is used by the source node to estimate cost of transmission during the data transmission phase. Node *i* also puts its own identity as "Sender ID" in the RCP. After modifying the RCP, the intermediate node would select one of its best neighbors (from the *Gradient_List*) and passes the RCP to it. If NS_i is the gradient neighbor set for node *i*, the next hop is chosen using expression:

$$Next\ hop = arg \max_{a \in NS_i} \{energy_{resd,a}\} \tag{3.2.2}$$

The route construction packets are characterized by the "PacketID". If a node receives a RCP with a "PacketID" which already exists in its RCP_List, the node simply updates the entries in the list for CRE, SenderID, and Hop Count values against the PacketID received.

Since the sink node generates multiple RCPs, these packets would construct multiple paths by locally reinforcing nodes. This route construction using local reinforcement rules may possibly result in the formation of braided paths. For example, a braided path is constructed when two RCPs coming from different paths reaches a neighborhood where both the nodes having these packets to send finds a common node to be high in residual energy compared to other neighbors (Fig. 3.4). Since next hop nodes

are selected based on local decisions, the formation of braided paths is possible and the protocol does not try to maintain disjoint paths which requires extra energy cost. Eventually, these RCPs would reach the source node through multiple paths. Like other intermediate nodes, the source node updates its *RCP_List* by extracting values from the RCPs and initializes data transmission process. The sink node sets up a timer and periodically sends RCPs when the timer expires (15s). An illustration of multipath route construction based on node's residual energy using two RCPs (an example case) is shown in Fig. 3.5. In this figure, the sink node chooses two neighbors, and intermediate node chooses one neighbor having the highest residual energy.

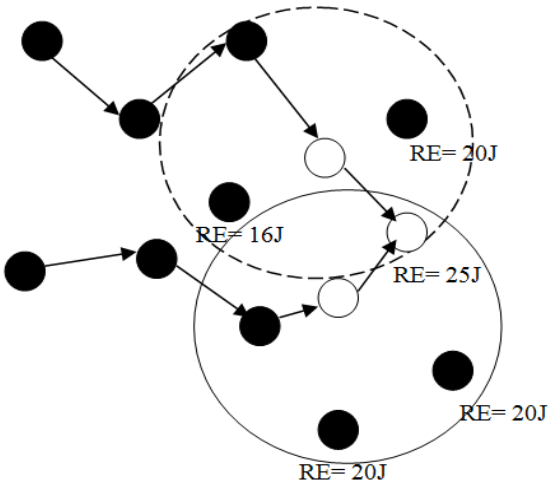

Fig. 3.4 Formation of braid in multipath.

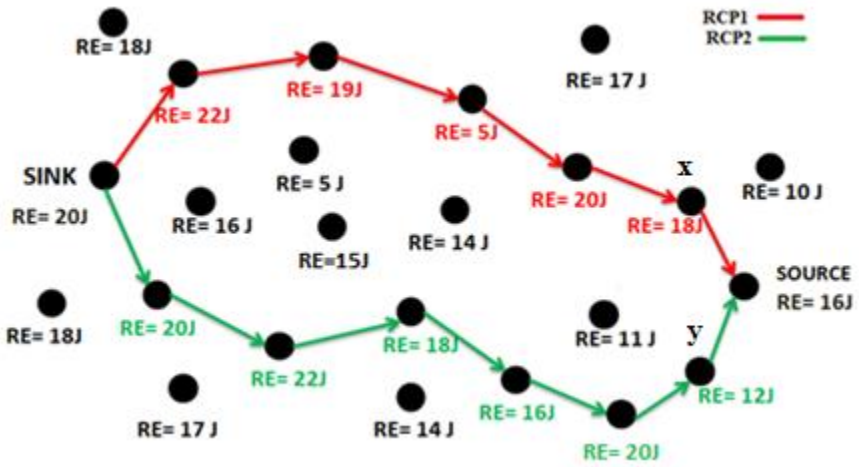

Fig. 3.5 Multipath Route Construction phase with two RCPs propagating in the network.

A representation of source node's *RCP_List* for the network shown above is shown in Fig. 3.6.

RCP_List

|Sender ID= x| PacketID = 1| HopCount =6 | CRE = 120|
|Sender ID= y| PacketID = 2| HopCount =7 | CRE = 139|
.
.
.

Fig. 3.6 Source node's RCP_List.

3.2.3.2 Variance Based Route Construction

The proposed protocol based on CRE comes with a drawback. By constructing paths based on the total residual energy of the nodes along the path, there may be a case when a path which has a weak node (node which is near failure) be chosen over a path which may not have a weak node. This may be because the path with a weak node has a

large number of nodes on it, making its CRE value higher compared to the other path not selected. This scenario is shown in Fig. 3.7. As a result, when data is transmitted over this path, the weak node may fail, making network dysfunctional. The network may have lived longer if this path was avoided, choosing the other path which had no weak nodes. To counter this problem, the proposed method was slightly altered to construct paths based on energy variance between nodes along the path. For this method, the Route Construction Packet is shown in Fig. 3.8. For the variance based method, the standard formula for variance is not applied. To find variation of energy along the constructed path, a different method is used based on residual energy difference between successive nodes on the path.

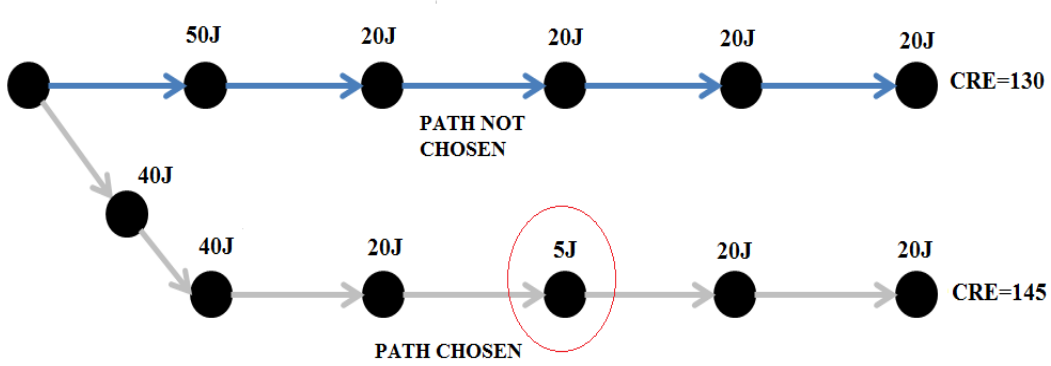

Fig. 3.7 Selection of bad path based on CRE (weak node is circled).

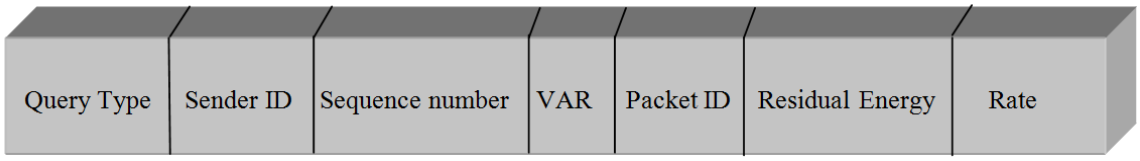

Fig. 3.8 Route construction packet (RCP) for the Variance based method.

For the variance based method, the sink would send multiple RCPs to its neighbors on its *Neighbor_List*. Each RCP has a unique "Packet ID". The sink generates

RCPs with a new "Sequence number" each route construction round. The sink specifies the "Query Type" in the RCPs. This type is checked at each node to see if the node has received the query packet previously, and to access *RCP_List* and *Gradient_List* for the particular query. The sink also piggybacks its residual energy value using the RCPs. The sink puts its node ID in the "Sender ID" field and its residual energy value in the "Residual Energy" field of the outgoing RCPs. The VAR field is set as 'zero' by the sink at the start. The sink also specifies the rate of data transmission to the source using the "Rate" field.

When an intermediate node receives this RCP, it uses the energy update by the sender to calculate relative residual energy difference. The receiving node would extract the residual energy of the sender from the RCP ("Residual Energy" field) and subtracts this value from its residual energy. The result is scaled down, and this result is averaged with the value extracted from the "VAR" field of the received RCP. The result of this operation is then stored in the "VAR" field of the outgoing packet (and in node's *RCP_List*). For example when *node i* sends an RCP to *node j*, then calculation of the VAR field by *node j* is given as:

$$x = \frac{\left| energy_{resd,j} - packet.Residual\ Energy \right|}{2} \tag{3.2.3}$$

$$VAR = \frac{x + packet.VAR}{2} \tag{3.2.4}$$

The intermediate node (*node j)* puts this newly computed "VAR" value, its node ID, and its residual energy into the RCP and forwards this new RCP to the neighbor

which has the highest residual energy. Node j with gradient neighbor set NS_j, selects the next hop node using expression:

$$Next\ hop = arg \max_{a \in NS_j} \{energy_{resd,a}\} \quad (3.2.5)$$

This next-hop neighbor is selected using node's *Gradient_List* for the particular query in hand. The *RCP_List* of the receiving node (*node j*) is also updated for the received values of Packet ID (copied from the received packet), VAR value (received value modified, equation (3.2.4)), and the Sender ID (copied directly from the received packet). The Packet ID and SenderID are used for tracing back the constructed path. This process of tracking back the reinforced path is explained in the data transmission section. The *Rate* field is used to specify the data transmission rate to the source node. The "Packet ID" distinguishes RCPs in the *RCP_List*. If a RCP is received with an existing PacketID, the entries (*VAR, SenderID*) against this Packet ID are updated in the *RCP_List* and the packet is passed on to next hop node using equation (3.2.5). The RCPs reach the source node through multiple paths. The source node receiving an RCP would calculate "VAR" value using equation (3.2.4), and updates it *RCP_List* (with *Packet ID*, *VAR*, and *Sender ID*) for the particular query.

3.2.3.3 Loop Prevention

Each node also has a *Tabu_List*, which is used for loop prevention. The *Tabu_List* prevents a locally reinforced node, during Multipath Route Construction, to reinforce the node which reinforced it. While the RCP is passed between sink and source, each node would copy the contents of the received *Tabu_List* in its own, and passes its own modified list forward. The sink node enters its own identity, and the identity of the node it is going to pass the Route Construction Packet to, into its *Tabu_List*. This *Tabu_List* is

passed on with the RCP to the next hop neighbor. The receiving node appends its existing *Tabu_List* with the received *Tabu_List* identities. Next it puts the identity of its chosen next hop neighbor in its *Tabu_List* and passes it forward. This list is used by a node to check if the node it chooses as the next hop node is either on the *Tabu_List* or not. If the chosen node is found in the *Tabu_List*, this node is not selected. In this way the algorithm avoids the formation of loops. The *Tabu_List* at each node is cleared at the start of new route construction round.

3.2.4 Data Transmission and Dynamic Path Selection

The source node after receiving the RCPs stores the values contained by these packets into its *RCP_List* memory (associated with the query). It also strips off the rate at which data is to be sent towards the sink, specified by the "Rate" field. The source would then have information about the constructed routes towards the sink. At this stage, the source node has to select the best path which is reliable for routing requested source data towards the sink. The proposed algorithm gives the path selection control to the source node by providing it with the network information. Thus, the source node has a global view of the constructed paths.

3.2.4.1 CRE Based Method

For this method, the source node makes the decision of selecting an optimal path based on the *Cumulative Residual Energy (CRE)* value of the path. The CRE value for a path is the total residual energy of the nodes along that path. This value for *n* nodes on a constructed path is given by:

$$CRE = \sum_{i=1}^{n} energy_{resd,i} \tag{3.2.6}$$

This CRE value specifies the health of the constructed path. Therefore, the source node should select the healthiest path from the set of constructed paths. This is done by selecting the path (P) which has the highest CRE value, stored in source's *RCP_List*.

$$Path\ for\ Transmission = max\{P_{CRE1}, P_{CRE2}, \dots\dots\dots, P_{CREn}\} \quad (3.2.7)$$

The source node then creates a Data Packet with the fields shown in Fig. 3.9. The process of energy update is also carried out during the data transmission phase by the protocol. Each node sending the data packet would piggyback its residual energy value in the data packet to update its energy at the receiving node. The source node puts its residual energy value into the Data Packet to update its residual energy value at a higher rate. The source increments the "Sequence number" value of the Data packet each time it is sent out.

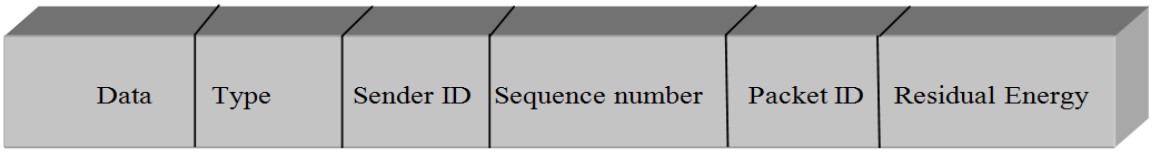

Fig. 3.9 The format of a Data Packet.

This sequence number is used to detect a missing event and initiate a recovery process by the sink. The "Type" field in the packet is used to access the *RCP_List* for the requested type at each node. The "Packet ID" field is used to track back the constructed route. Each node in its *RCP_List* has information about the RCP sender (Sender ID) and the packet number (Packet ID). An intermediate node receiving the Data Packet, checks the "Packet ID" field, and finds the next hop neighbor corresponding to this Packet ID from its *RCP_List.* This operation is explained by Fig. 3.10, which shows the operation of *RCP_List* and its parameters. If node 1 sends the RCP (RCP X) to node 2, then node 2 would store the information contained in the RCP X packet into its *RCP_List*. Node 2

then passes this packet by incorporating its details towards the source node (S) as RCP Y packet.

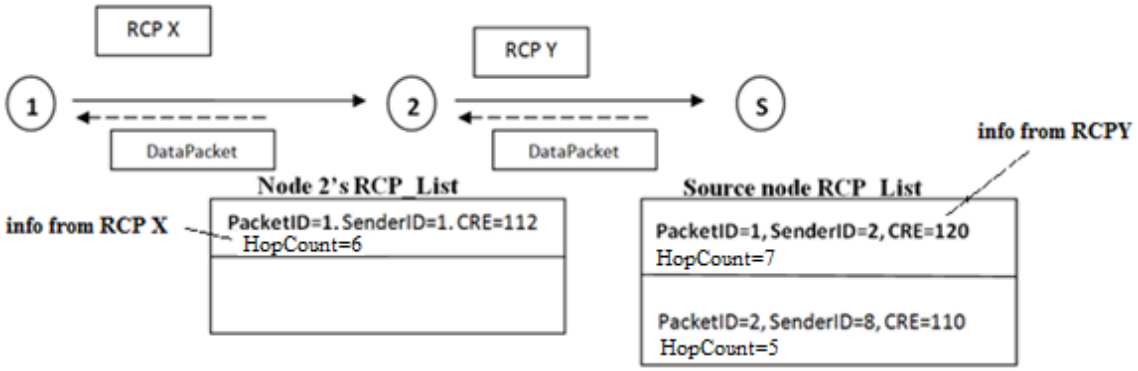

Fig. 3.10 Example of "RCP_List" operation.

Source node stores the information in this received packet by making entry into its *RCP_List*. When the source node has to send data back towards the sink using the constructed route, it does not have to know about all nodes on this route. After selecting the route with the highest CRE (CRE = 120), it would fill the Data Packet with the PacketID (PacketID=1) corresponding to this CRE value stored in its *RCP_List*. This "Packet ID" is useful for intermediate nodes. The Data Packet is sent by the source (S) towards node 2 (since its identity is present in *RCP_List* against the chosen CRE). When node 2 receives the Data Packet, it extracts the PacketID (the value is 1) from it and searches in its RCP_List for the SenderID corresponding to this PacketID. This SenderID = 1 (for this PacketID = 1) is the node to which the Data Packet is sent by node 2. In this way using the "PacketID" field, a constructed path could be traced. In the Data Packet, the node would incorporate its node identity in the "Sender ID" field, and residual energy value in the "Residual Energy" field to update its energy value in the receiving node's *Gradient_List* (for the query "Type" specified in the packet). Eventually this Data Packet

containing source Data reaches the sink. The source node sends the event data at the requested rate (highest rate compared to other phases) towards the sink.

Since the next *Route Construction* takes place after a certain amount of time (15s), the source node uses the constructed paths alternatively. To use these paths alternatively, the source node should have a mechanism to estimate the energy dissipation across the used path, so that other paths can also be used (based on the highest CRE). The source node updates the CRE value for the used path in its *RCP_List* dynamically using an expression:

$$CRE = CRE - (HopCount * energy_{TX} + HopCount * energy_{Rx}) \quad (3.2.8)$$

The "Hop Count" value is specified in source's *RCP_List* for the used path and the "energy$_{TX}$ *(energy$_{Rx}$)"* is the transmission (reception) energy cost. By estimating the CRE value for the used path, the source node can better compare the CRE values of the multiple constructed paths. In case if a used path degrades, the traffic is switched to an alternative path dynamically. For this case, if source node finds any unused path higher in CRE compared to the used path, it will route data using this alternate unused path. By doing this, the source node can ensure that it would send the data using the healthiest path which has the least probability of failure while data is routed through it. The proposed protocol discovers multiple paths every *Route Construction* phase. Using local decisions to construct multiple paths and allowing the source node to select the optimal path incorporates efficient load balancing.

3.2.4.2 Variance Based Method

For the variance based method, the source node selects the path (P) which has the lowest "VAR" value in its *RCP_List*.

$$Path\ for\ Data\ transmission = \min\{P_{Var1}, P_{Var2}, \ldots\ldots, P_{Varn}\} \quad (3.2.9)$$

After selecting the path, the source generates a data packet which is shown in Fig. 3.9. The Data packet is sent by the source using the selected path towards the sink. Each data packet is assigned a "Sequence number" by the source node. This sequence number is carried by the data packet, and is used by the sink to detect a missing event. For example, in Fig. 3.11, the source node (S) selects the path constructed with "VAR=1.66", and then selects node 2 (SenderID=2) corresponding to the VAR value and sends the data packet to it. In the data packet the source node puts its identity, residual energy, and Packet ID corresponding to selected "VAR" value. The receiving node (2) updates the energy value of the data packet sender in its *Gradient_List* for the specified "Type" in the data packet. The receiving node extracts the Packet ID (Packet ID=1) from the data packet and finds a corresponding node (Sender ID =1) for this Packet ID. This receiving node would then generate a Data Packet putting its own energy update and identity and passes it to next hop node (node 1). In this way the data packet reaches the sink, updating residual energy at intermediate nodes at a higher rate.

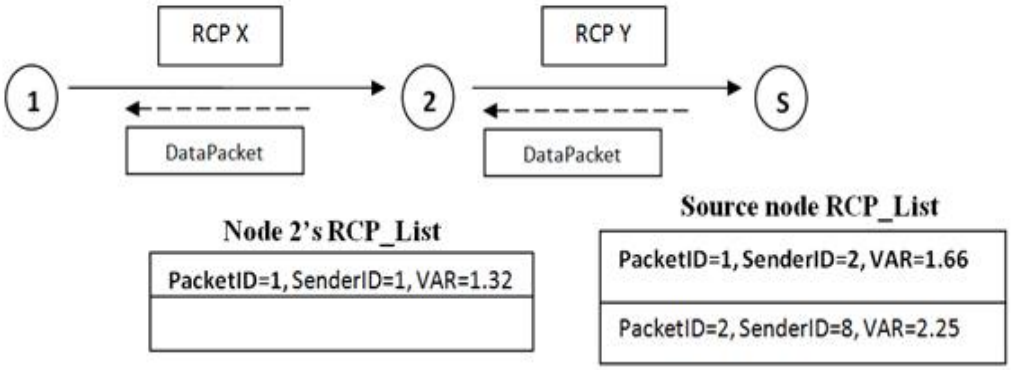

Fig. 3.11 Data transmission operation for the variance based method.

3.2.5 Protocol Operation

Fig. 3.12 illustrates the operation of the two main stages (*Route Construction* and *Data Transmission*) of the CRE method of constructing multiple paths (e.g., two paths). For example, the Sink node reinforces two nodes (node with Residual Energy = 20 Joules and Residual Energy = 22 Joules), which, in turn, will reinforce those upstream nodes having the highest residual energy. Each node along the constructed paths would add its residual energy when reinforcing its upstream neighbor, and, thus, the residual energy readings accumulate along the paths. Eventually, the RCPs propagate through two paths and find the Source node. The source node will determine the path with the highest cumulative residual energy (CRE2 = 139 J), and will select that path for transmitting its data towards the sink. Data is sent along the selected path until its value becomes lesser than the CRE of the alternative path. The CRE value of the used path is dynamically updated using equation (3.2.8).

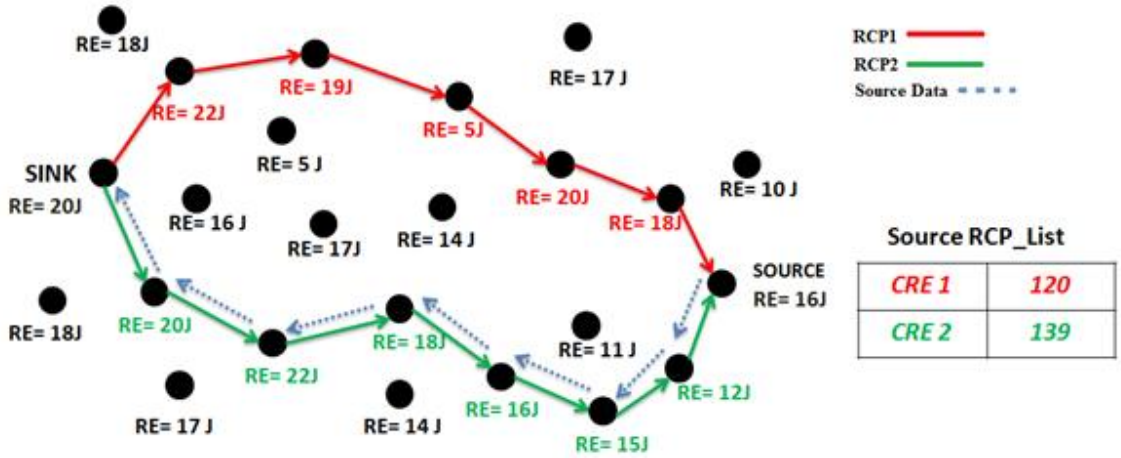

Fig. 3.12 Route construction and data transmission operation (CRE method).

The operation of variance based method is shown in Fig. 3.13. The route construction phase is similar to the CRE based method. The sink reinforces e.g., two of its neighbors (node with Residual Energy = 20 Joules and Residual Energy = 22 Joules).

These reinforced nodes would then select one neighbor with high residual energy value and the path is constructed till the source node is reached. At the source node, the information about the constructed paths is stored, and it would then select a path which has the lowest variance value (VAR2= 1.66). At each node, when an RCP is received, the node would compute a new VAR value using equations (3.2.3) and (3.2.4), which is passed on with the outgoing RCP. The general operation of the protocol is shown in Fig. 3.14.

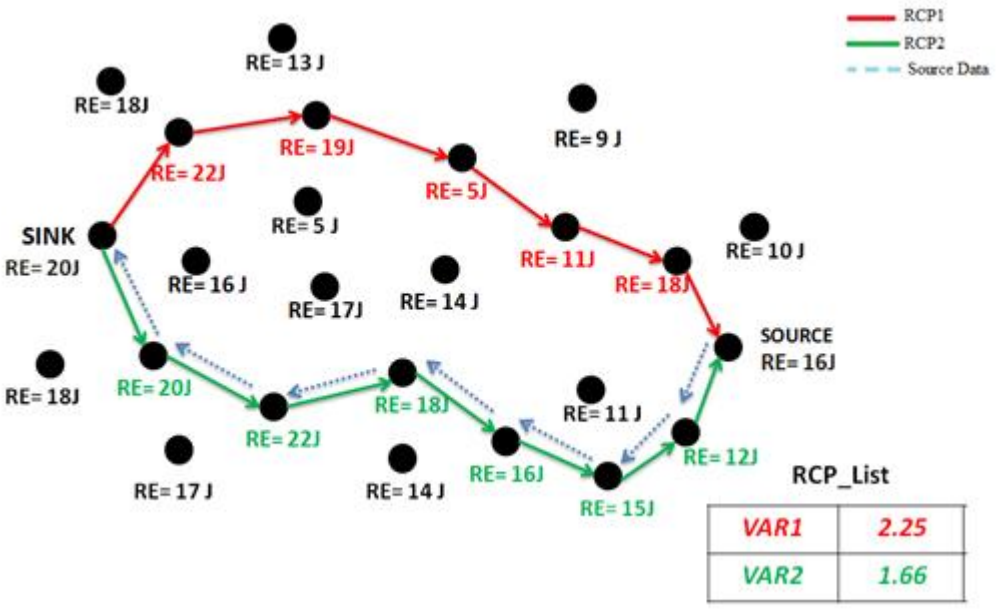

Fig. 3.13 Route construction and data transmission Operation (Variance method).

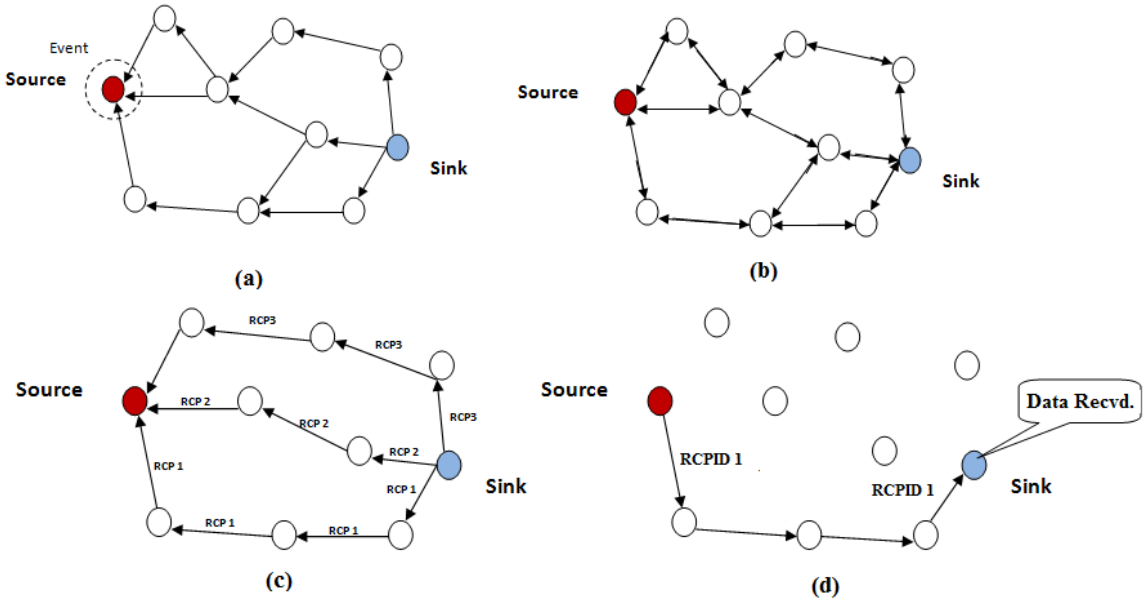

Fig. 3.14 Protocol operation (a) query propagation, (b) gradient setup, (c) route construction, (d) data propagation.

3.2.6 Path Maintenance and Repair

Although, the proposed multipath routing keeps searching for healthy paths each *Route Construction round*, a node failure is problematic in between these rounds. The temporary outage or a sudden energy depletion of a sensor node on the selected optimal path poses a serious problem to the proposed multipath routing method. Since the periodic *Route Construction* takes place after a certain amount of time (15s), the source node would have no information about a failed node. Therefore, the source node would keep on sending the data towards the sink on the selected path. To counter this problem, the proposed method also incorporates a reactive repair algorithm. Using this repair algorithm the protocol allows the sink node to detect a path failure and initiate route repair. The sink detects this failure by detecting missing event data which is not received by the sink. Sink detects a missing event by checking the *Sequence number* incorporated in the Data Packet by the source node, and comparing it from the previously received

Data sequence number. When a sink receives a Data packet, it captures its *Sequence Number* and waits for a time twice as the event reporting rate. After this, the sink again checks for the *Sequence Number* of the received data and compares it with the previously captured value. In case, if sink finds a missing event, it initiates path repair mechanism. During the repair process, the sink sends multiple RCPs towards the source. These RCPs propagate through intermediate nodes as explained in the previous sections. These RCPs construct multiple paths, and the information about these paths is handed over to the source node. The source node would then choose a new healthy path and continue its transmission.

Another point to consider here is the problem associated with a weak node (low in residual energy). By locally reinforcing neighbors to construct multiple paths, the protocol allows the use of these weak nodes on a constructed path. Although, the variance based scheme avoids choosing a path with weak nodes for data transmission, there still is a possibility of selecting such a path. The problem therefore is the data transmission reliability of the selected weak node. If such a node receives some data and fails, then the repair algorithm would be initiated more frequently. To avoid this situation a weak node should not be included in the multipath route construction process. To do this, each node keeps a check for its residual energy threshold value. If this threshold is below a certain predefined value, the node broadcasts a message to its neighbors informing them about its reluctance to participate in route construction process. The proposed algorithm defines a weak node as the node which has only two percent (2%) of the total energy left in it. This node would then send a *"Timeout"* message to all its neighbors. In this packet the node specifies its residual energy as zero. The receiving node would update their *Gradient_List*

for this weak node with the corresponding residual energy value as zero. By doing this, the weak node opts out from being part of the next multipath route construction process or the recovery reinforcements. By prematurely taking weak nodes out of data transmission process, the network can be made to live longer compare to not using this approach. This accounts for better fault tolerance on the part of the proposed algorithm.

3.3 Load Balancing

Over time, when sensor nodes do not have the opportunity to replenish their energy supplies, the network will cease to operate (die). Ideally, this should happen when all nodes (die) reach zero energy at the same time. This is possible if there is load balancing in the network. An ideal load balancing would cause all nodes to consume energy at the same rate and die at the same time. Load balancing done in an approximately uniform manner will extend the lifetime of the network to the fullest possible extent. Load balancing may not extend the life of a greedy node to the fullest, but would extend the lifetime of the network. A unique feature of the proposed protocol is the integration of path level load balancing with network wide load balancing. The network wide load balancing is incorporated during the route construction phase, whereas the path level load balancing comes into play during the dynamic path selection phase.

For network wide load balancing, the protocol allows each node in the network to select the next-hop neighbor which has the highest residual energy. This approach would allow nodes to construct a new path with different nodes on it each discovery phase. Adopting this approach would cause a lazy node in the network, which does nothing, to work. A problem with such load balancing approach is the use of greedy local decisions to construct optimum paths. In other words, without the global knowledge about network

conditions, a local decision may not guarantee an optimum path. It is shown through results [11] that such a load balancing approach could not achieve maximum load balancing. The proposed protocol makes such load balancing efficient by adding on a mechanism to select an optimum path and incorporate path level load balancing.

To achieve maximum load balancing, the protocol uses local decisions to construct multiple paths, and allow the source node to select an optimum path by providing it with information (CRE or VAR) about these paths. Therefore, the source node can make the best decision to select an optimum path. Thus, a decision made by the source could increase the probability of finding the best path each time, and incorporates path level load balancing in the network. This kind of path level load balancing would add on to increase the efficiency of load balancing algorithm, which in turns maximizes network lifetime. For the CRE based method, the source node could also use the discovered paths for alternative path routing, each time selecting a path having the highest Cumulative Residual Energy. Using equation (3.2.8) the source could update the CRE value for the used path and could switch to an alternate path if there is some degradation in the used one. Doing this the source node would use these paths carefully balancing between their healths.

3.4 Summary

The chapter explains the operation of the proposed multipath routing protocol. The operational details of the two variants are also explained. The protocols are simple in operation and energy-efficient. The incorporated load balancing maximizes network lifetime, lowering energy consumption. The protocol also has a reactive repair algorithm

which helps in recovering from node failures, allowing network to continue its operation. The implementation of the protocol on a simulator is explained in the next chapter.

Chapter 4

Simulator Design

A Java simulator was developed to simulate the performance of the proposed multipath route construction protocol. The network is modeled using object oriented programming. The Java based simulator can easily be extended for modeling other algorithms. The description of simulator development is presented in this chapter.

4.1 Protocol Phases

4.1.1 Neighbor Discovery Routine

After node objects are created and randomly placed on a square grid, neighbor discovery is initiated. The neighbor discovery process in the simulator uses the Postman Metric, which allows each node to determine its neighbors within its radio range (r). A node would compute its distance towards all nodes in the network, and stores only neighbors that are within its radio range, into its "*myNeighborList"* memory.

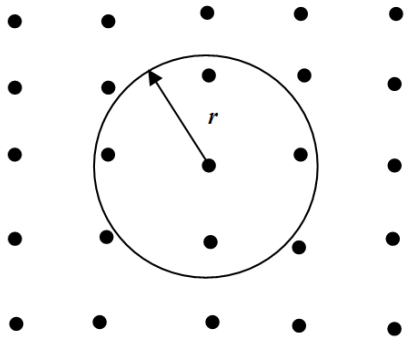

Fig. 4.1 Illustration of node's radio range (r) and nodes under it.

Each node has a "*myNeighborList",* which is used to store one-hop neighbors. This list is passed to the "Packet Transmitter" class each time a packet is to be sent to the

nodes on this list. The main routine for the simulator is represented by Fig. 4.2. A node class with its instance variables and methods is shown in Fig. 4.3. The simulation instantiate independent nodes using this class.

Main Algorithm

1: Initialize parameters
Dim: dimension of the square grid
N: number of nodes in the network
r: radio range
$energy_{init}$: Initial energy
$energy_{TX}$: Transmission energy cost
$energy_{RX}$: Reception energy cost

2: Generate a coordinate (x, y) list having random coordinates

3: Create N nodes with parameters defined in step 1, and assigning coordinates generated in step 2

4: for *i* = 1 to N
node(i).myNeighborList = one-hop neighbors (neighbor discovery routine)

5: start timers: querythread.start ()
energythread.start ()

6: IF (node ==SINK)

7: IF (queryFlag==True)

8: Create a query packet and pass it to all Sink neighbors;
sink.myNeighborList.myQueryPkt[]=queryPacket,
sink.myNeighborList.myQueryRxFlag=true

9: for *i* = 1 to N (Iterate over all the methods for each node)

node(i).checkQueryRxFlag ()
node(i).checkRcpSendFlag (), node(i).checkRcpRxFlag ()
node(i).checkDataSendFlag (),node(i).checkDataRxFlag ()

10: for *i* = 1 to N
IF ($energy_{resd,i}$ == 0)
Exit ()

11: RETURN step 6

Fig. 4.2 The main routine of the Java based simulator.

The main routine first instantiate node objects assigning them parameters such as coordinates, radio range, and energy values. Next, the routine computes distance based neighborhood for each node, assigning neighbor nodes to *myNeighborList* of each node. The routine then calls specific methods to process different phases of the protocol for each node. During operation, the routine checks if any one node dies, to stop the simulation, and exit the routine. The main routine is designed as a polling scheme to process each node's designated flags to handle data flow. This polling is shown by the flow chart in Fig. 4.4.

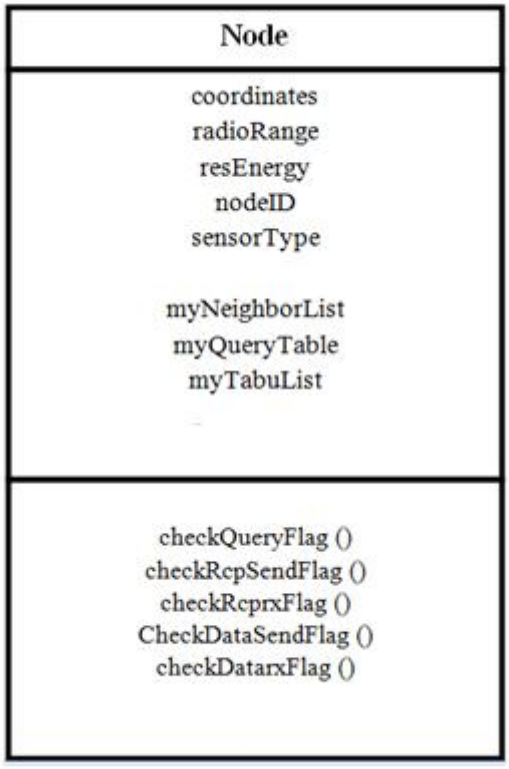

Fig. 4.3 The Node class.

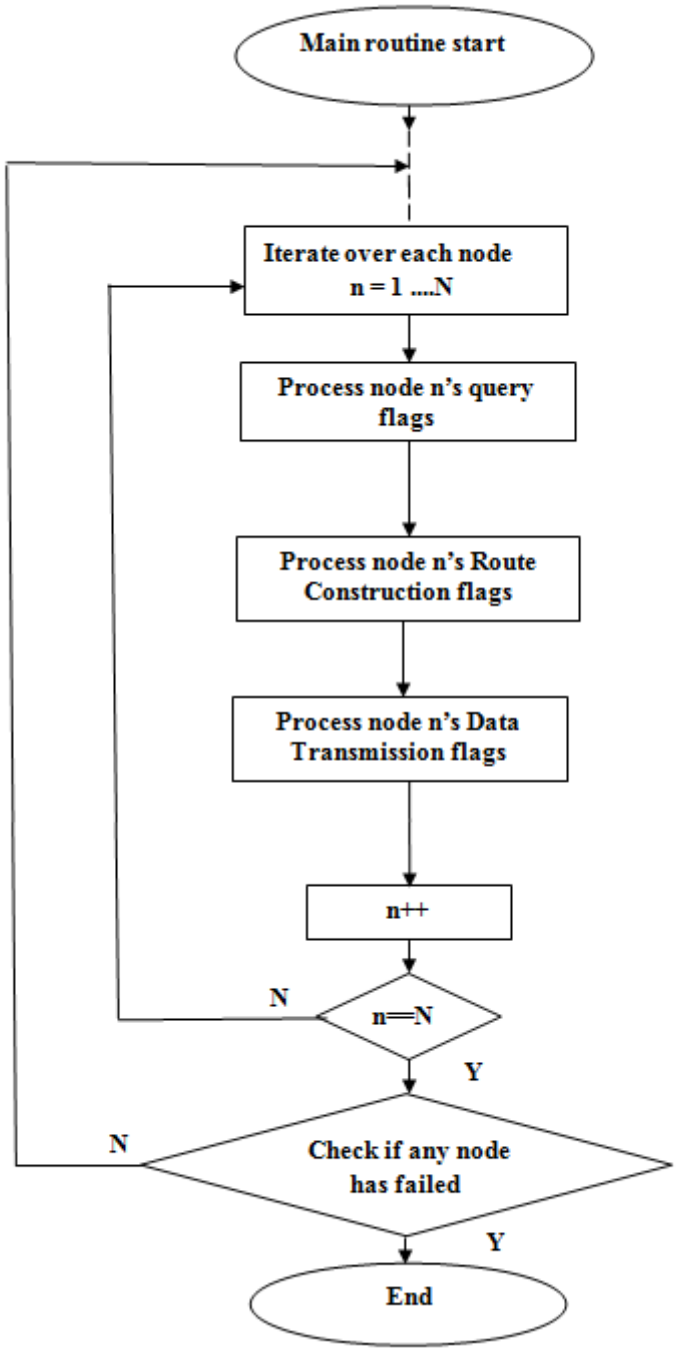

Fig. 4.4 Flow chart for general node processing.

4.1.2 Query Processing

After neighbor discovery, the sink would inject a query in the network for the sensing task it is interested in. Each node receiving this query would make an entry in its "*myQueryTable"*. For each query, a node has an associated gradient list (*myGradientList*). This gradient list would store the identity of the sender and its residual energy value. The class association for node, query table, and gradient list is shown in Fig. 4.5. Each node can support multiple queries. This is represented by the "*Node"* class having an association with the "*Query"* class. Each node has *myQueryTable* instance of "*Query"* class type. A query can also have multiple gradients. The "*Query"* class has *myGradientList* instance of "*Gradient"* class type. Each node can have multiple neighbors, as represented in Fig. 4.5. Each query also has an associated *myRcpList*. The sink node sends the query packet to its neighbors on its *myNeighborList*. An intermediate node *n* receiving the query packet would perform the processing shown by the flow chart in Fig. 4.6.

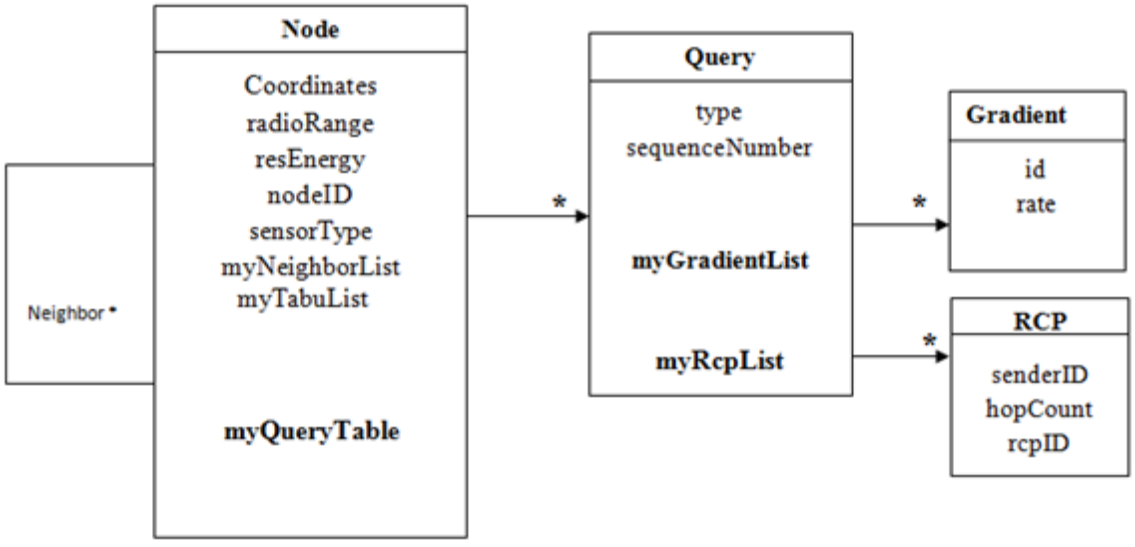

Fig. 4.5 Class association of node for Query processing.

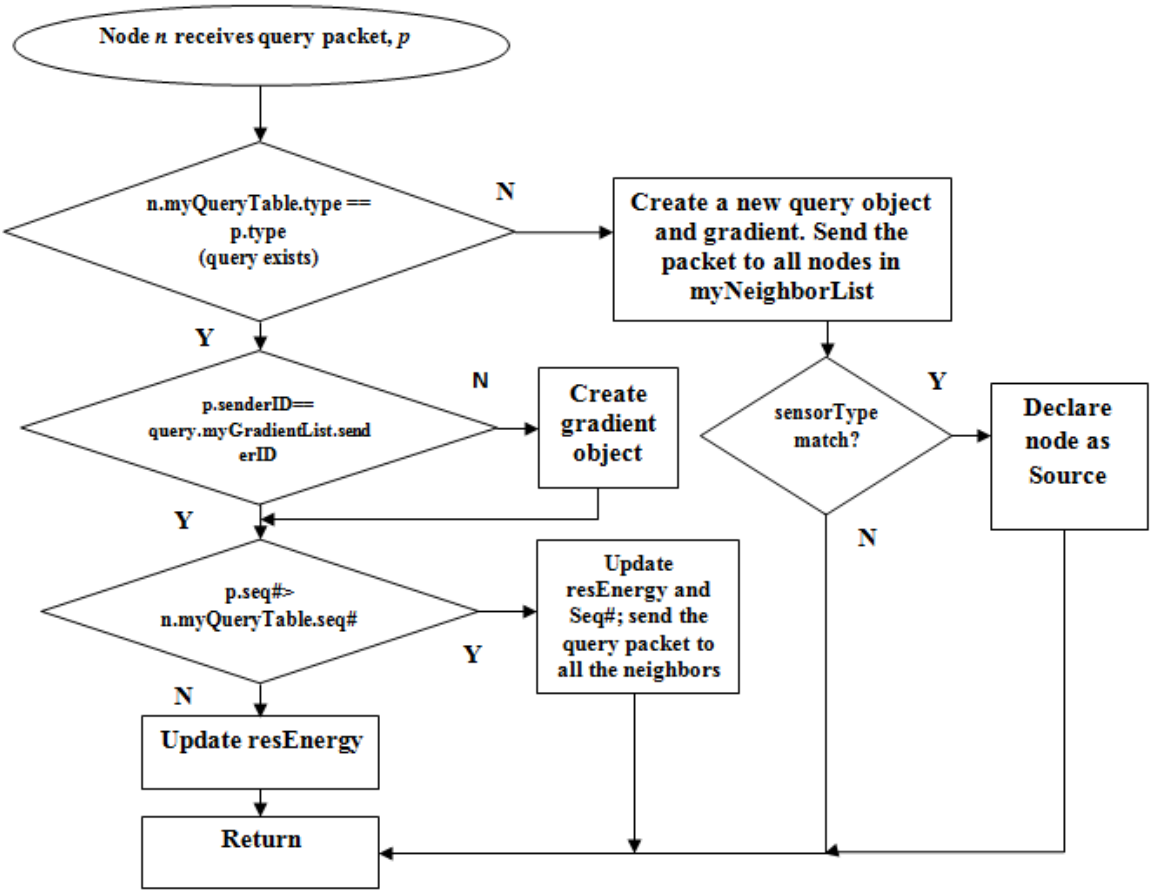

Fig. 4.6 Flow chart for processing the query packet received by a node.

4.1.3 Multipath Route Construction

After the sink has sent its query to all neighboring nodes, it starts a thread to wait for 5s before starting the next phase. After 5s, this timer thread would check if the sink has received updates from all of its neighbors. If the sink has received updates from all of its neighbors, it triggers the route construction phase. The flow chart for this initiation is given by Fig. 4.7. During the multipath route construction, the sink node would send multiple Route construction Packets (RCPs) towards the source node.

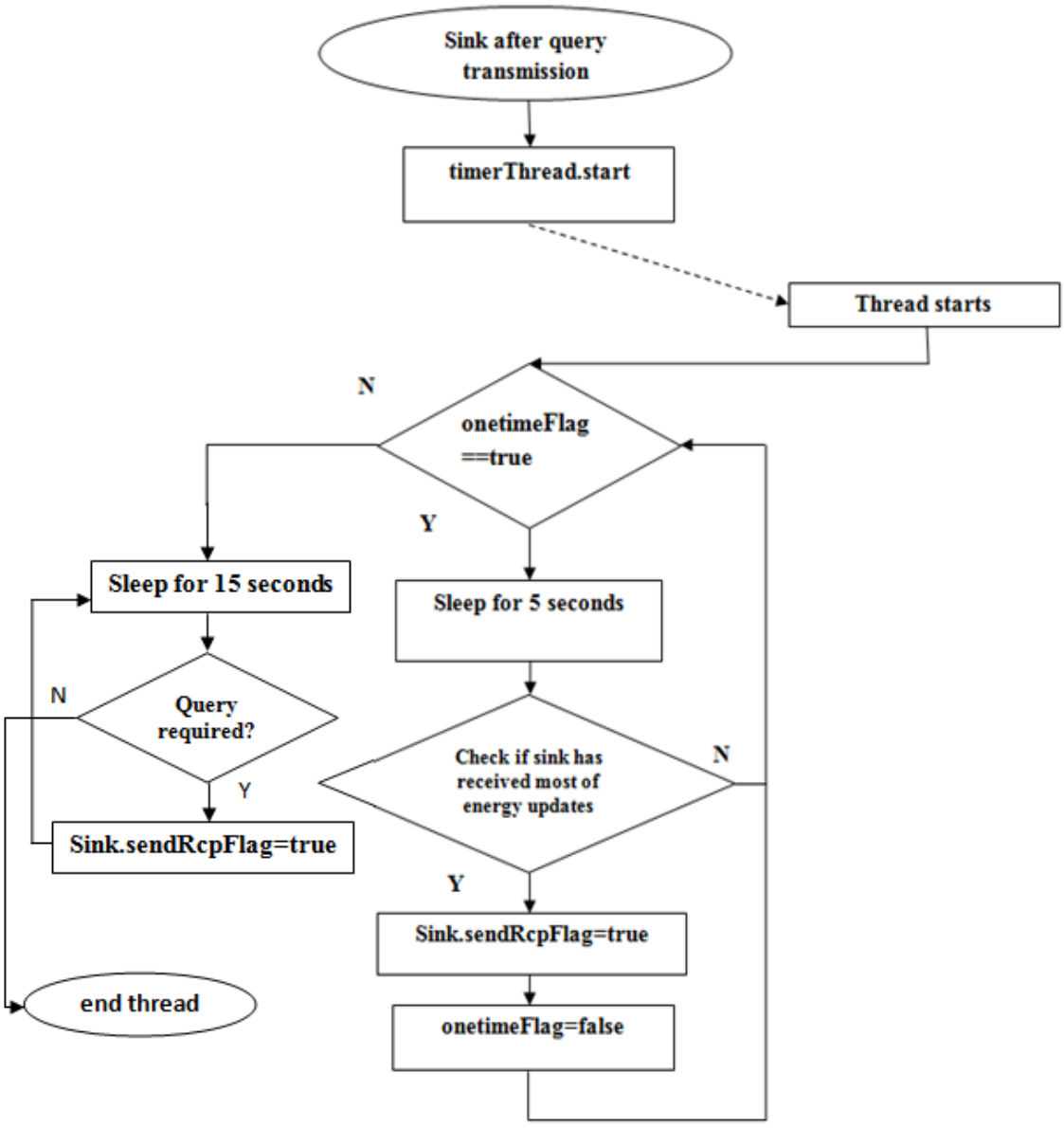

Fig. 4.7 Setting up route construction phase at the sink node.

The *sendRcpFlag* is used to indicate that the sink needs to arrange for transmission of Route Construction Packets. This flag is set every 15s to send Route Construction Packets towards the source. Each time this *sendRcpFlag* is set to true, the sink node would follow the routine indicated by flow chart in Fig. 4.8.

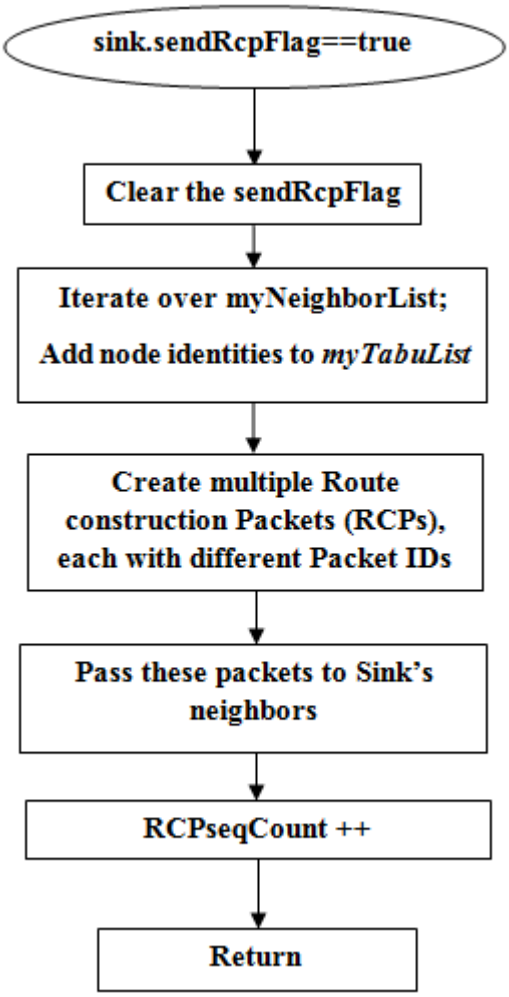

Fig. 4.8 RCP transmission by the sink.

When the Route construction Packet is received by a node, it would update its information (for the particular query) and pass this packet further towards the source choosing its best neighbor based on highest residual energy. This processing at an

intermediate node is explained by Fig. 4.9. Here, the *myRcpList* is the specific list for the query in hand (node.query.myRcpList).

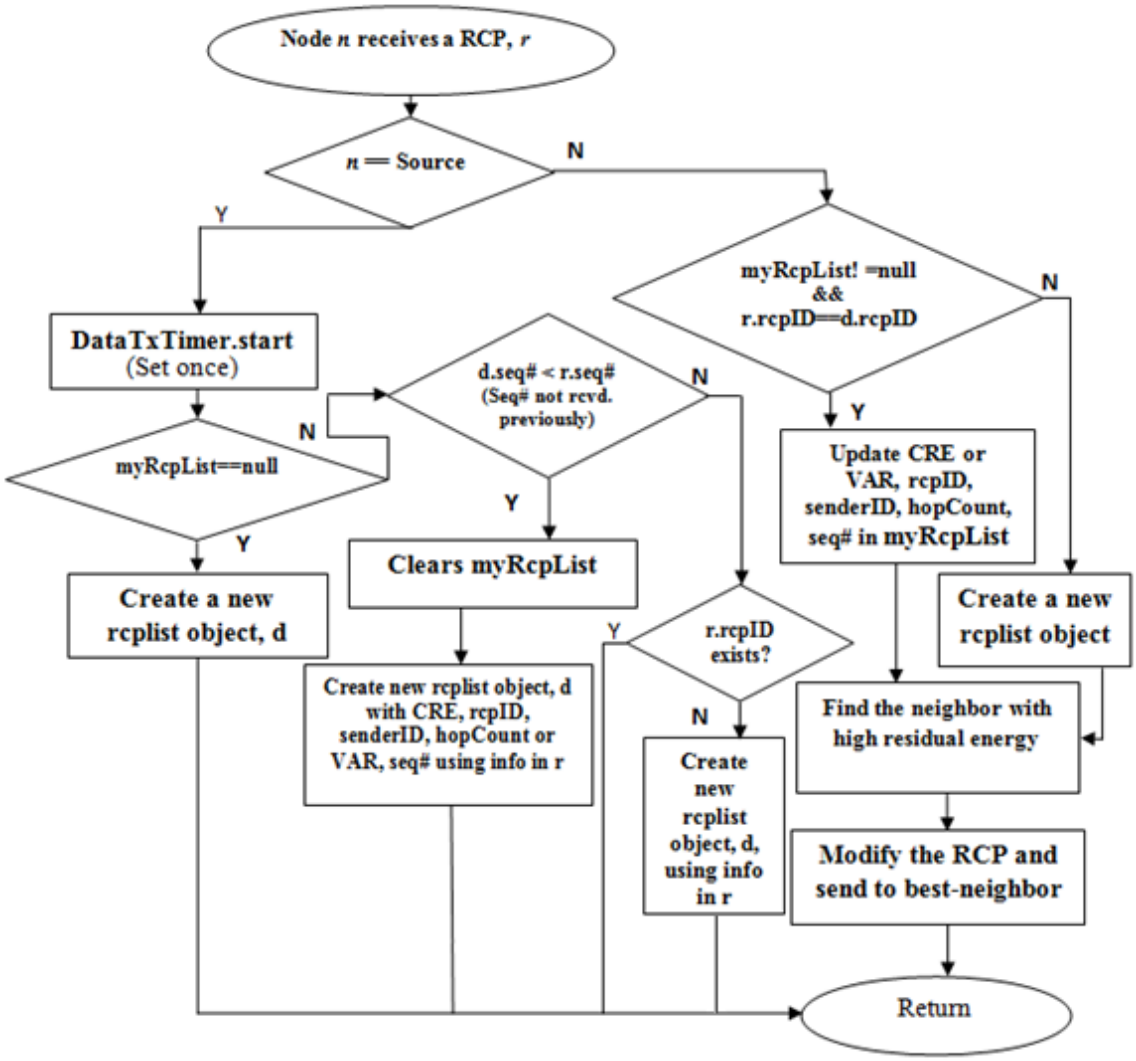

Fig. 4.9 Flow chart for processing the received RCP.

4.1.4 Data Transmission Process

The Source node triggers the DataTxTimer thread. The thread sleeps for a second (1s), and when it wakes up, it sets the sendDataFlag as high (true). This flag is read on the run from the main routine. When it is high, the source would arrange for the

transmission of data. The source node creates a "DataPacket" with its data, and sends it towards the sink using the best discovered route (Fig. 4.10).

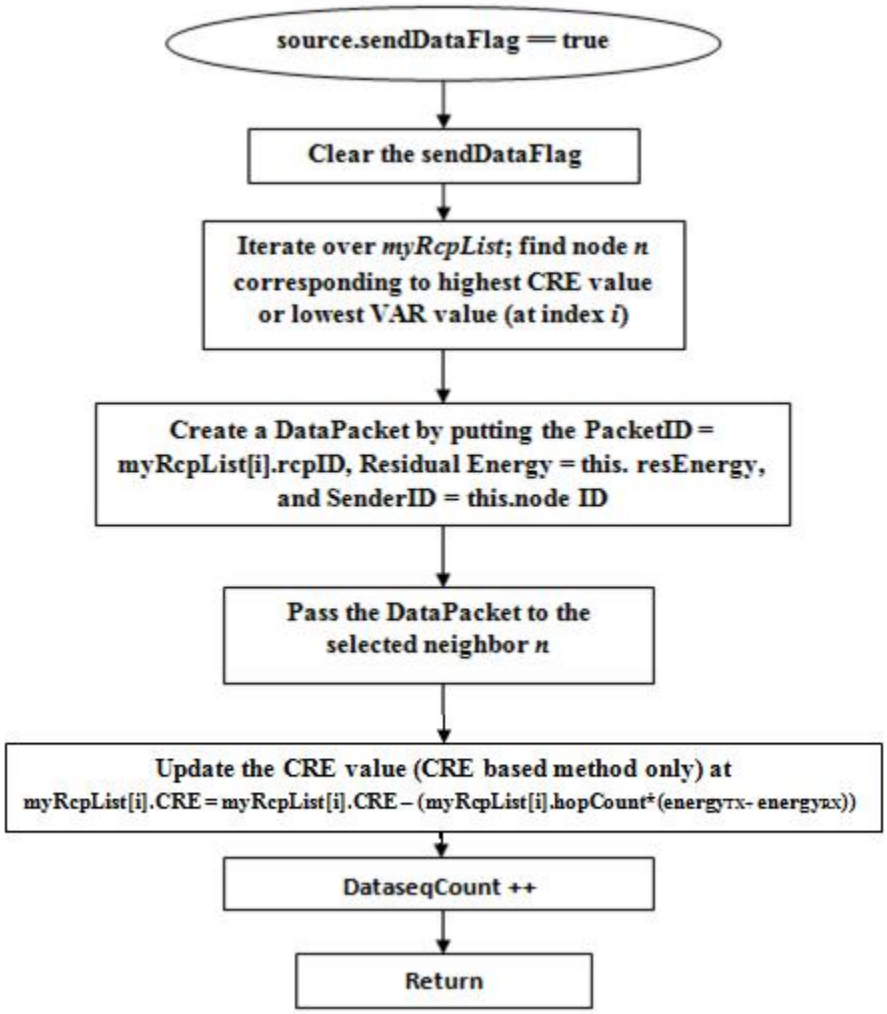

Fig. 4.10 DataPacket transmission at the Source node.

When this DataPacket is received at an intermediate node (Fig. 4.11), the node would update the sent residual energy value of the data packet sender in its *myGradientList* corresponding to the query type specified by the Data packet. To pass the DataPacket onwards, this intermediate node would then iterate over its *myRcpList* and finds a next hop node against the PacketID (*rcpID)* specified by the DataPacket received. The DataPacket is then sent towards this next hop node, after modifying the DataPacket,

incorporating sending node's residual energy and identity. If the receiving node is the sink, it would only update the energy value for the sender.

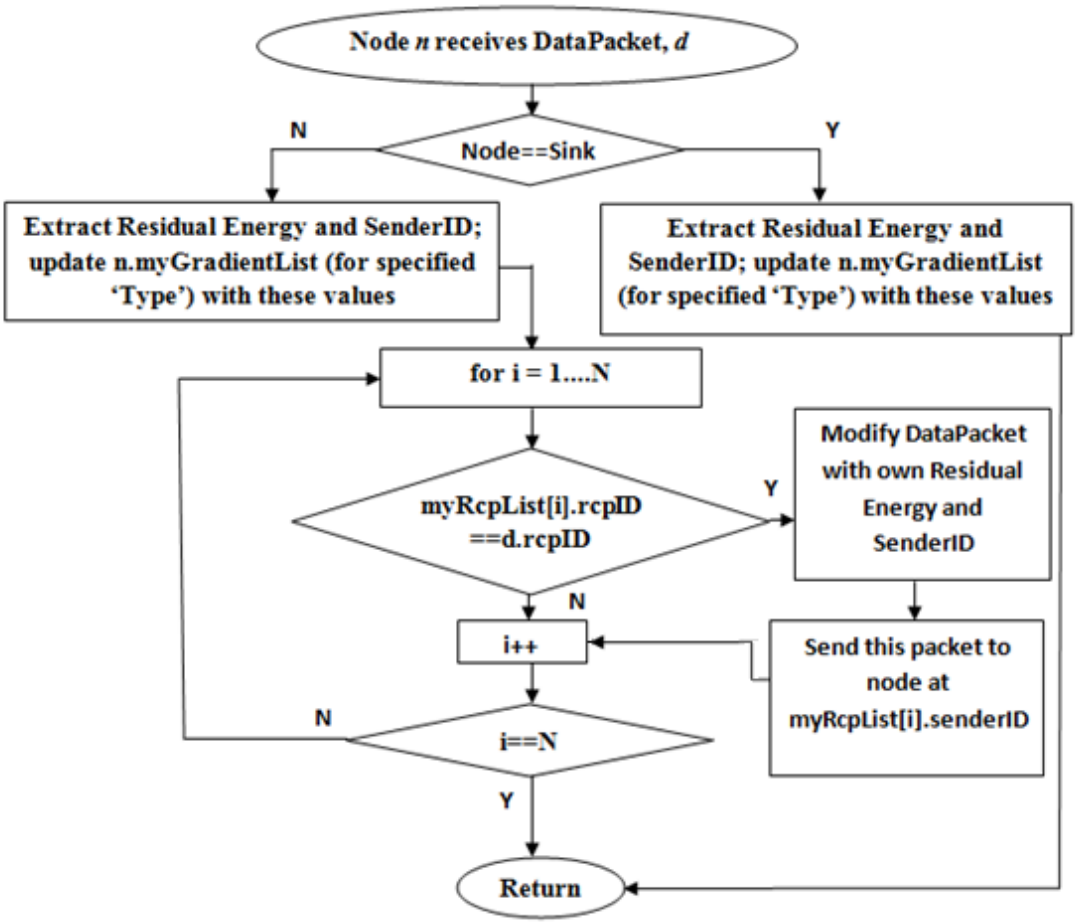

Fig. 4.11 Data Packet processing at an intermediate node.

4.2 Radio Model

In the simulation, the radio model is adopted from the work in [8]. The radio model assumes energy dissipation of 50 nJ/bit for operating the transmitter or the receiver. The transmit amplifier has an energy dissipation specification of 100 pJ/bit/m^2. In order to transmit a "k-bit message" at a distance d, the energy dissipation is given by the model as:

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

$$E_{Tx}(k,d) = E_{elec} * k + \epsilon_{amp} * k * d^2 \quad (4.2.1)$$

For the reception of the message the radio dissipates energy given by:

$$E_{Rx}(k) = E_{Rx-elec}(k)$$

$$E_{Rx}(k) = E_{elec} * k \quad (4.2.2)$$

Where $E_{Tx}(k,d)$ is the energy spent on transmitting "k-bits", and $E_{Rx}(k)$ is the energy spent on receiving "k-bits". The amplification factor for the transmit electronics is $\epsilon_{amp} = 100$ pJ/bit/m^2. The energy cost of operating transmitter/receiver electronics is $E_{elec} = 50$ nJ/bit. Using equation (4.2.1) and (4.2.2), the simulation for the proposed protocol can compute energy consumed by each node per bit sent or received. This radio energy model helps in testing the energy-efficiency of the protocol accurately. The block diagram of the model is shown in Fig. 4.12.

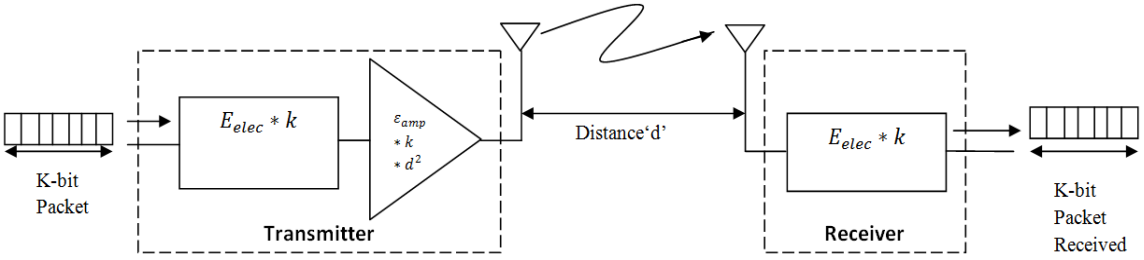

Fig. 4.12 First order radio model [8].

4.3 Packet Processing in Simulation

In the simulation, when a node wants to pass a packet to any of its neighbors, does that by putting the data packet into the memory of the receiving node and setting its flag high. Each node has three packet storing arrays namely, myQueryPkt[], myRcpPkt[], and myDataPkt[]. A node also has three flags of array type, namely myQueryRxFlag[], myRcpRxFlag[], and myDataRxFlag[]. If a node *j* has any type of packet to send, it will

store this packet in the memory of the receiving node *i* and set its receive flag as high. Node *i* is indicated by its receive flags if it has received something. If a node has received a packet, it will process the received packet, and store it in its appropriate list. For example to process a Route Construction Packet (RCP), node j would put the packet in node *i*'s "myRcpPkt[]" memory and set its "myRcpRxFlag[]" high. When node *i* is indicated by checking its "myRcpRxFlag[]", it will extract the packet from "myRcpPkt[]" and would store it in its "myRcpList" after processing it. This is shown in Fig. 4.13. The "PacketReceiver" class is used to process the packet received by extracting it from node's memory and storing values in designated lists. Similarly, the "PacketTransmitter" class is used to process the packet to be sent, by putting the packet into the memory of the nodes on *the list of nodes* provided to this class by the sender node.

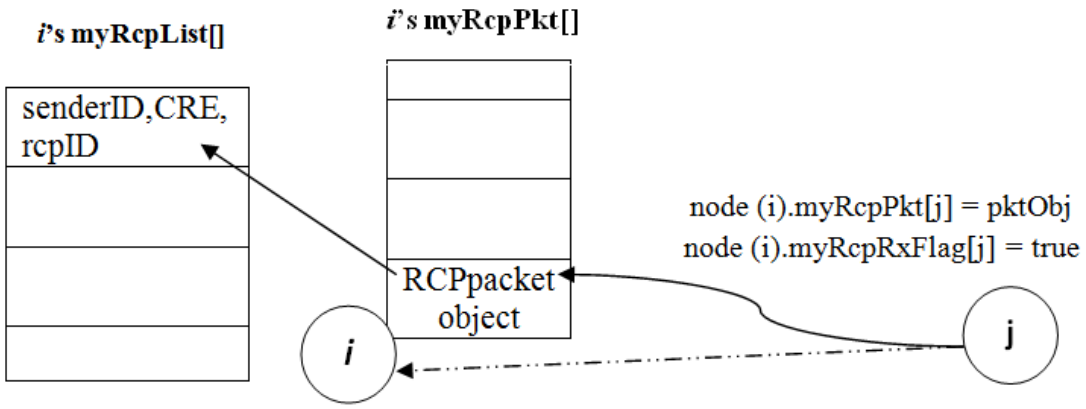

Fig. 4.13 Using flags and memory to pass the packet.

4.4 Simulator Model

The basic system model for the simulator is shown in Fig. 4.14. For other experiments (DD, load balanced DD) this model can easily be extended to the one shown in Fig. 4.15. Other class diagrams are shown in Fig. 4.16 and Fig. 4.17.

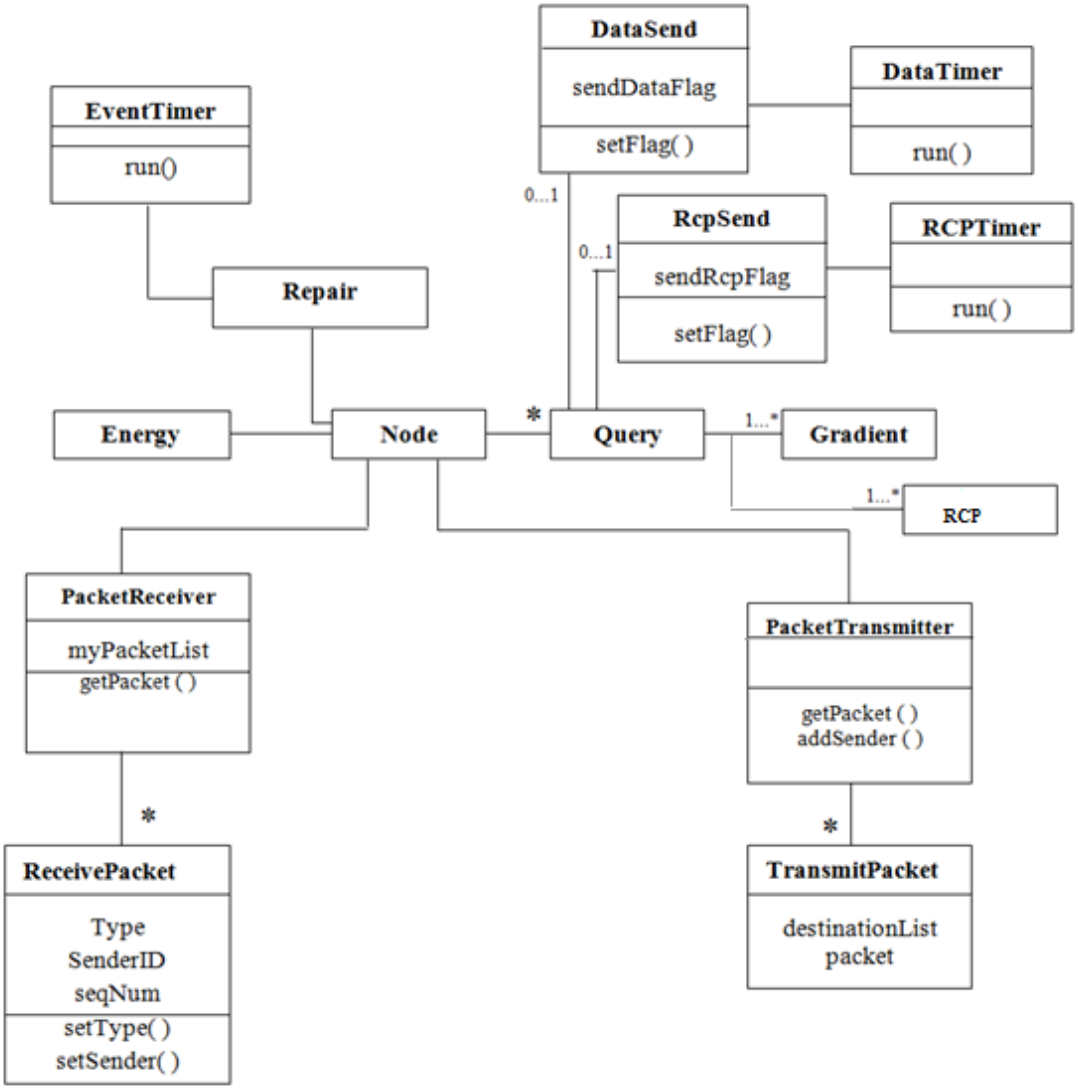

Fig. 4.14 Basic simulator model.

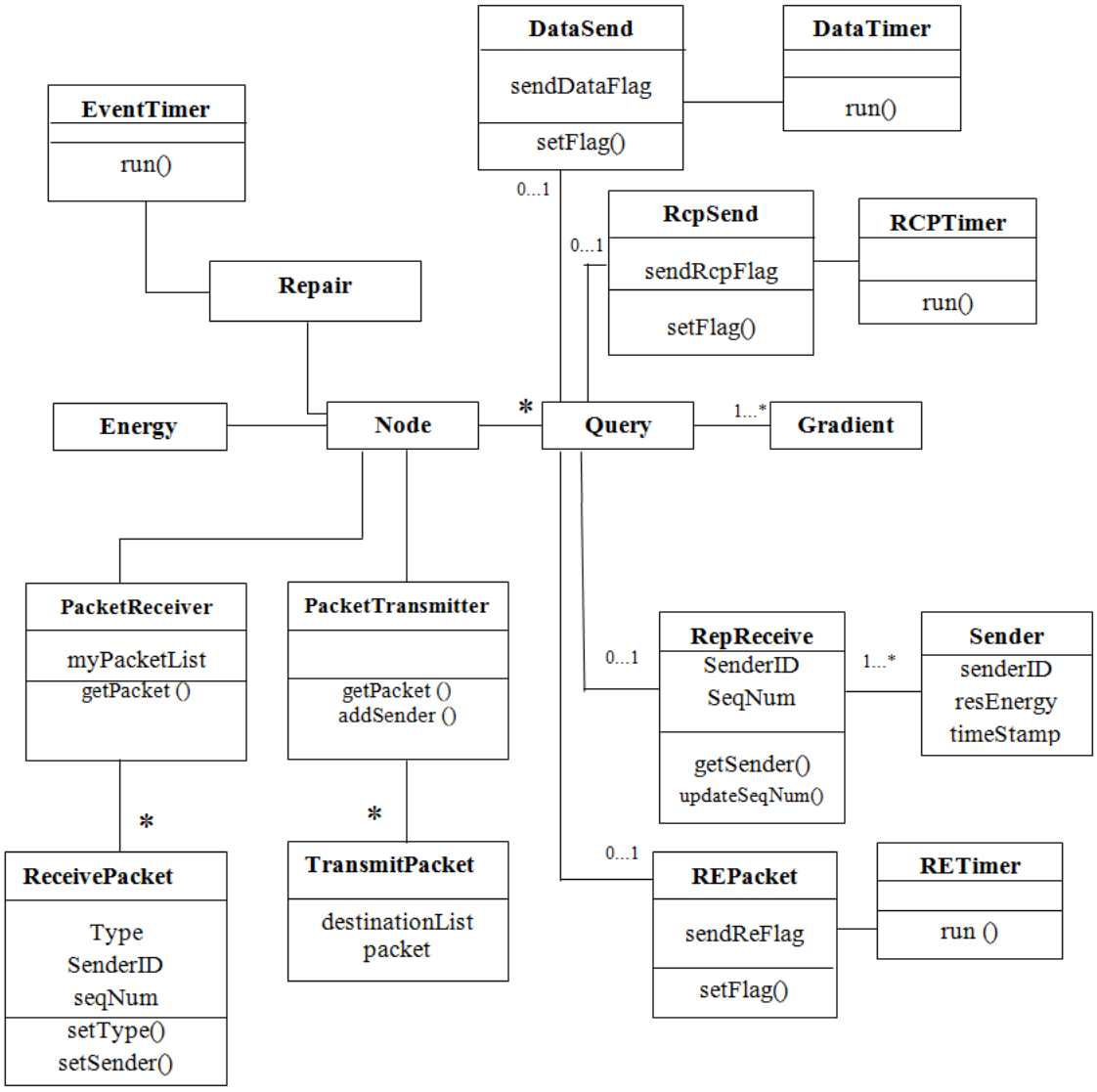

Fig. 4.15 Extended model for the simulator for directed diffusion based experiments.

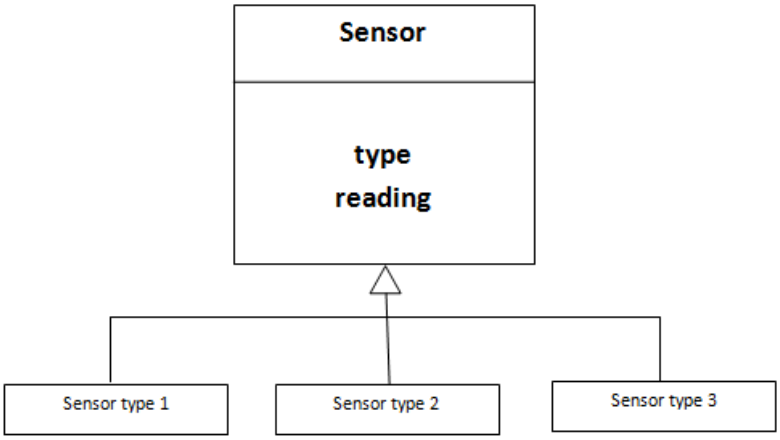

Fig. 4.16 Sensor node class diagram.

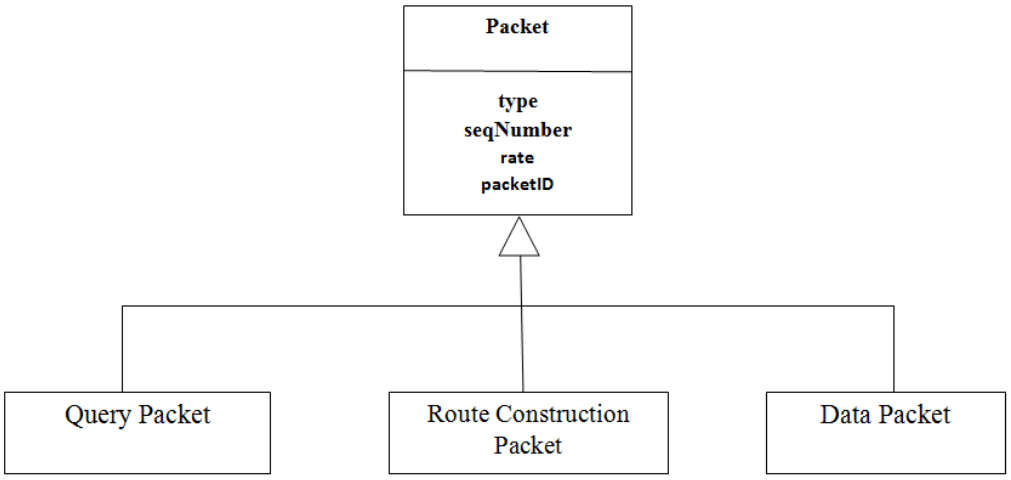

Fig. 4.17 Packet class diagram.

4.5 Energy Class

The "Energy" class is used to calculate energy variance of all nodes in the network. It iterates over all the nodes and captures the residual energy for each node. After this it performs the variance operation and stores all the data in a file. This class also stores the energy values for all nodes in a file, which is used to plot energy distribution on an intensity map. This class has an associated timer thread, which is

invoked every 20s to update logged files. The energy variance for a network with *N* nodes is calculated using simple standard expression:

$$Energy\ variance = \frac{\sum_{i=1}^{N}(energy_{resd,i} - \mu)^2}{N} \quad (4.5.1)$$

$$Mean\ (\mu) = \frac{\sum_{i=1}^{N} energy_{resd,i}}{N} \quad (4.5.2)$$

The energy class also computes node's average energy consumption using the metric from [7][45]:

$$Node\ energy\ consumption = \frac{\sum_{i=1}^{N}\left(energy_{init,i} - energy_{resd,i}\right)}{N\sum_{j=1}^{S} data_j} \quad (4.5.3)$$

Where *N* is the total number of nodes in the network, $energy_{init,i}$ is the initial energy of node *i*, *S* is the number of sink in the network, and $data_j$ is the number of data packets received by sink j.

4.6 Route Repair

The sink node when receives the first Data Packet, would initiate the route repair algorithm. The sink would trigger a timer thread which would allow the sink to check for a missing event. Whenever the timer wakes up, the sink would check the sequence number of the data received and compares it with the previously captured value to determine if route repair is required or not. The flow chart to do this is shown in Fig. 4.18.

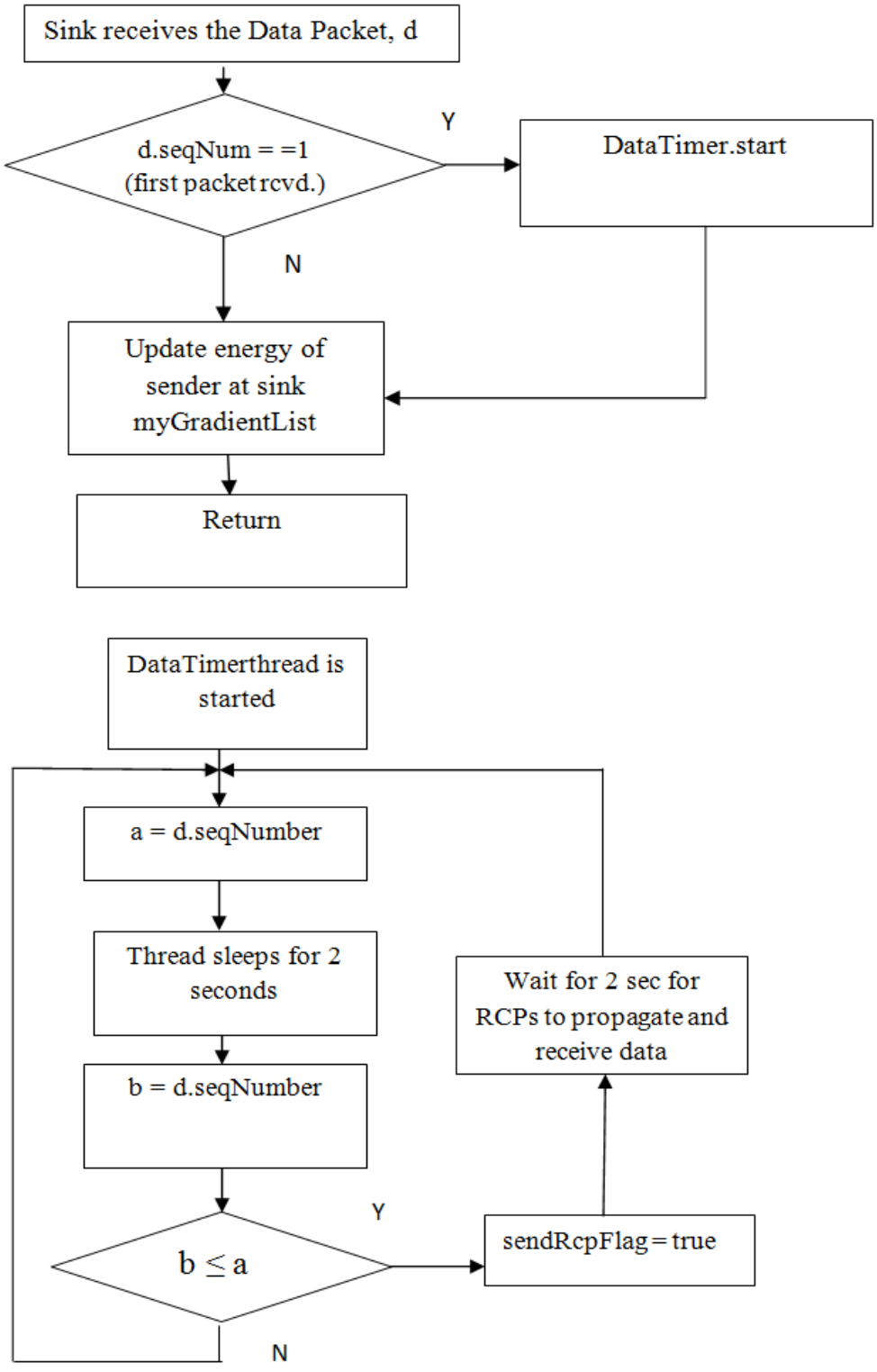

Fig. 4.18 Flow chart for repair algorithm.

4.7 Intensity Map Design

One of the major contributions by the thesis is the design of a visualization tool, which is used to visually analyze logged energy readings by the Java simulator. The visualization tool is used to visualize energy distribution across the network on an intensity map. The tool is general purpose, and can be used to plot energy readings logged by any simulator (or real network). The visualization tool software is developed in NI LabVIEW 2011 development suite. The tool works offline as well as online.

For online operation, the visualization software is integrated with the Java based simulator described in the previous section. The software would read the energy readings from the java simulator on the fly and would plot them over an intensity map. Therefore, energy consumption for each node can be monitored and the load balancing effect can be visualized. The software also captures intensity map images for various stages of network operation and stores them in the memory. This mode is useful for monitoring the network when the java simulation is running.

The software can also plot the energy readings in an offline mode. For the offline mode, it can read a file and plot energy values on an intensity map. The front panel of the software is shown in Fig. 4.19.

The axes on the intensity map correspond to the coordinates of a node (x, y). For the visualization tool, the java simulation can be run by placing the sink and source node at the extreme ends of the network. This is done to trace sink and source node, and to trace constructed paths in between them. The front panel of the software during operation looks like Fig. 4.20. The basic flow of the visualization software is illustrated by the block diagram shown in Fig. 4.21.

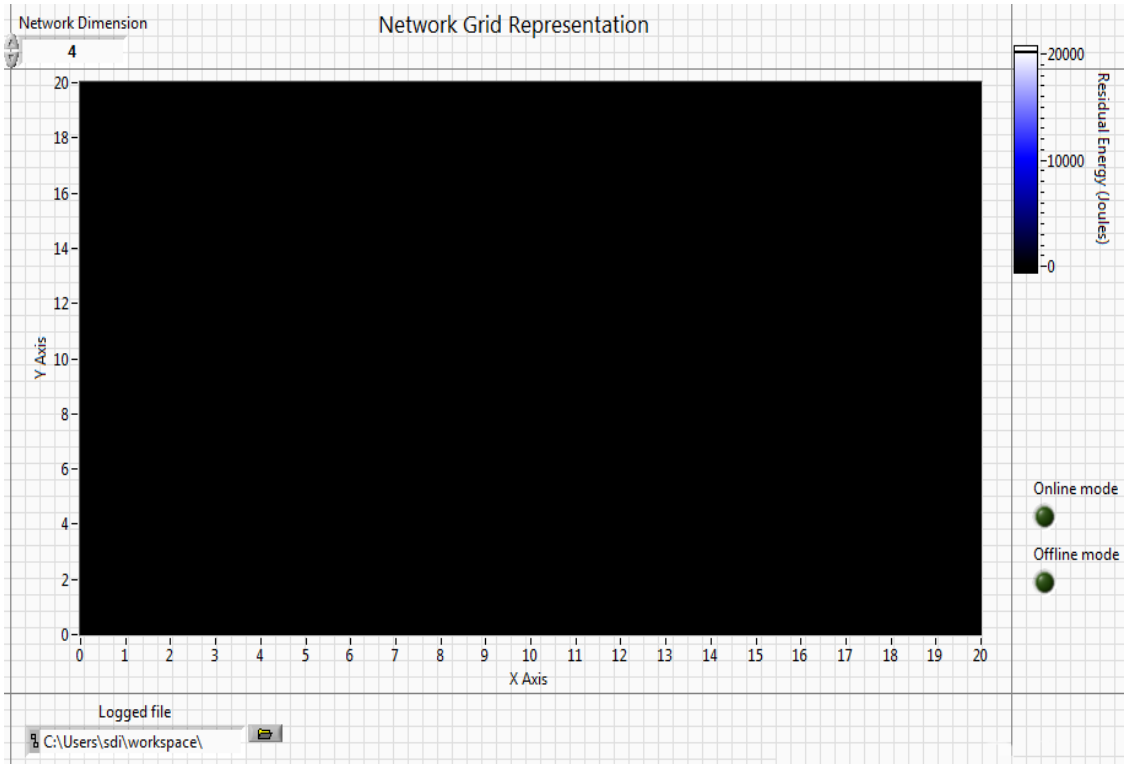

Fig. 4.19 Front panel diagram of the visualization tool developed in LabVIEW.

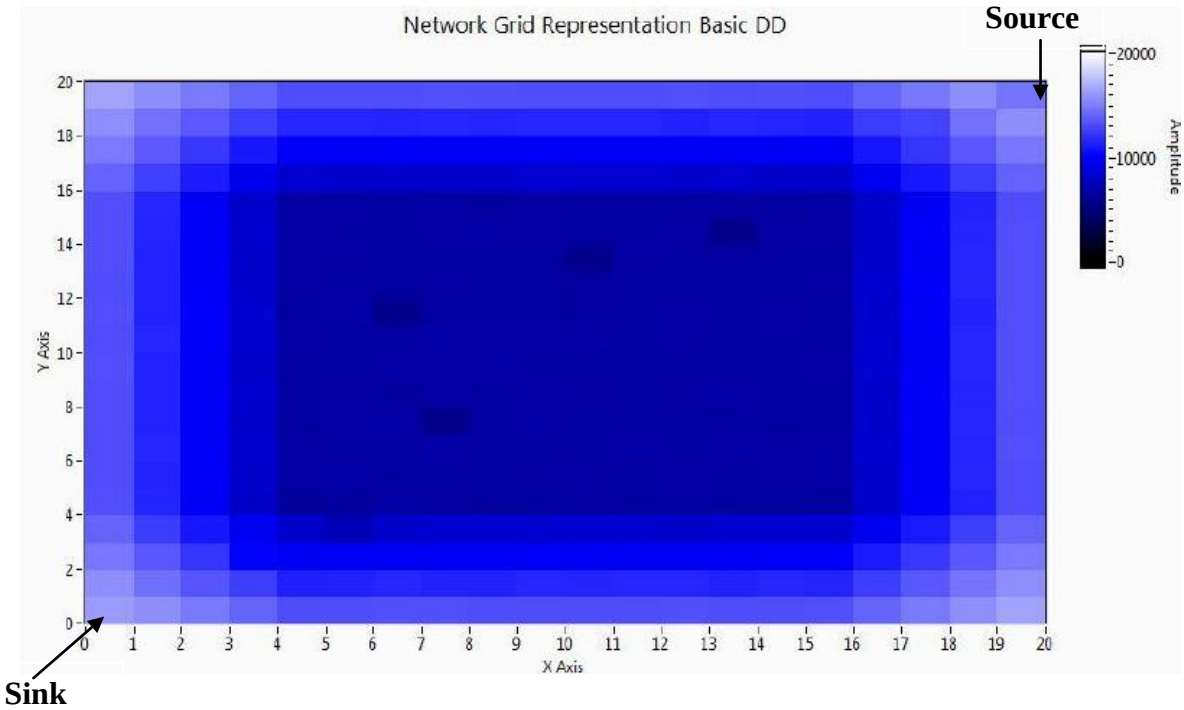

Fig. 4.20 Front panel of visualization tool during operation.

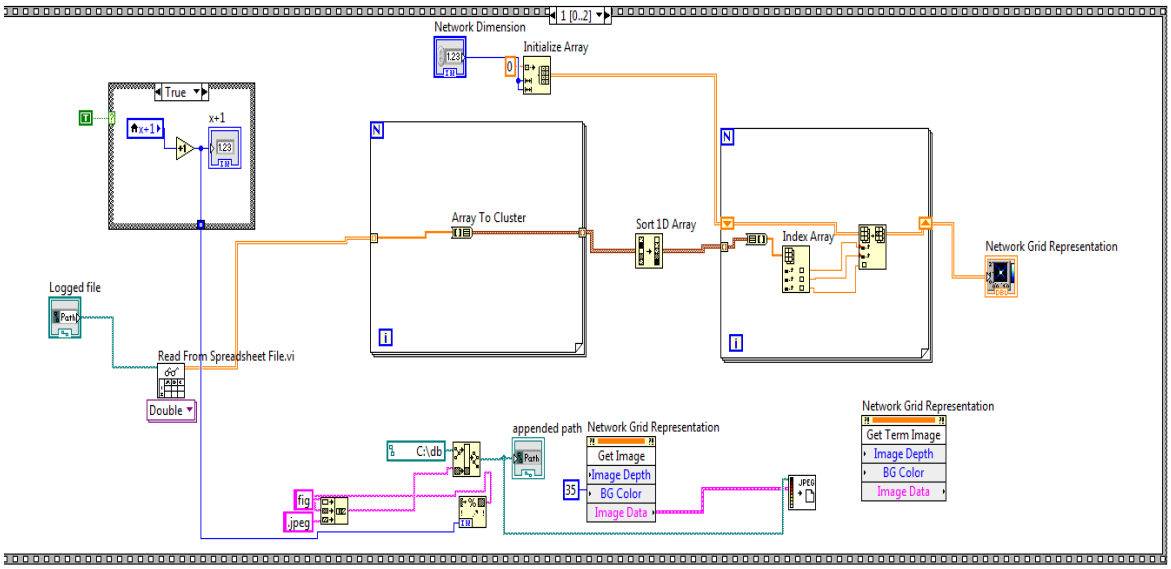

Fig. 4.21 Main block diagram for visualization tool in LabVIEW.

4.8 Summary

This chapter covers technical description of the simulator design. Class diagrams and flow charts are used to explain the design of the simulator. To visualize the effectiveness of load balancing, network can also be visualized on an intensity map. On this map each node is placed with respect to its coordinates in a square area and its residual energy value is plotted. Using this visualization tool, energy depletion over a constructed route, can be visualized. In the next chapter, experimental results are presented.

Chapter 5

Experimental Results

In this chapter, experimental details, results, and analysis for the proposed multipath protocol are presented. The proposed protocol is simulated on a Java based simulator and its performance for different number of nodes, random sink-source positions, and node failures is evaluated. The experiments are designed to evaluate energy variance of the network, network lifetime, load balancing, and node energy consumption. The proposed multipath routing protocol (both variants), in terms of its performance, is also compared with the following protocols:

- Basic Directed Diffusion [7]
- Load Balanced Directed Diffusion [11]
- Multipath method implemented on Directed Diffusion [47]

5.1 Experimental Setup

To evaluate the performance of the protocol, a Java based network simulator is developed, in which instantiated sensor nodes are deployed over a square field area (A). Deployed N nodes in the network are scattered randomly over the field and are stationary. The protocol is evaluated for different number of nodes varying N from 100 to 600. To ensure high network connectivity and avoid partitions in the network, the area of the network A is scaled to maintain a certain node density in the network. An example of a network with 100 nodes deployed randomly having high connectivity is shown in Fig. 5.1. The sink and source nodes are placed randomly in the network as well as at the extreme ends of the network to evaluate the protocol performance. The network supports

multiple source nodes as well as multiple sink nodes. However, to analyze the protocol for highly dense network, and verify proper implementation of the simulator, one sink-source pair is used in the network.

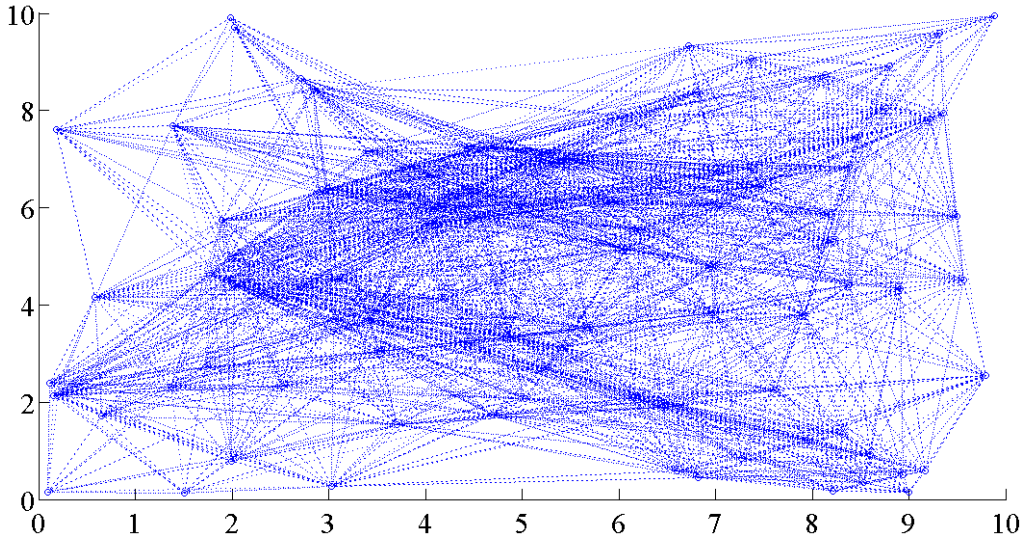

Fig. 5.1 Connectivity map of network for N = 100.

Each node in the network has a configurable radio range. The radio range in the simulations is fixed to r = 4m, which maintains high connectivity. Nodes in the network do not aggregate data. Each node can sense one particular stimulus. For the proposed protocol, the energy model chosen for the sensor node has an energy cost of 50 nJ/bit for both transmission and reception [8]. Each node is assigned an initial energy of 5 J. The energy cost of processing and idle operation for a node is negligible compared to the communication cost and is neglected in the simulation. Different phases of the protocol in the network have fixed periodic intervals. The output of the simulation is captured in a log file which is generated by the Eclipse IDE platform for Java. Eclipse prints all the communication details of the network as a text file, which is helpful for traceability and verification of the simulation. Also, for all the experiments, the simulation itself generates multiple log files. These log files are appended every 20 s with the values of

network such as energy variance, residual energy values of the nodes, average energy of the nodes, and energy samples used for intensity map plotting. These logged files can be used later for analyzing network lifetime and energy consumption. Simulation parameters for the proposed protocol and protocols used for comparison are kept the same. These parameters are given in Table. 5.1.

Table. 5.1 Simulation Parameters.

Parameter	**Value**
Grid Size	25 m x 25m
Number of Nodes (N)	100,200,300, ..., 600
Radio Range (r)	4 m
Initial Energy	5 J
Transmission Energy [8]	$E_{elec} = 50$ nJ/bit $E_{Tx}(k,d) = E_{elec} * k + \epsilon_{amp} * k * d^2$
Reception Energy [8]	$E_{elec} = 50$ nJ/bit $E_{Rx}(k) = E_{elec} * k$
Query Propagation Period	5s
Exploratory Data Propagation Period (only for Directed Diffusion based protocols)	10s
Route Construction Phase Period	15s
Data Transmission Phase Period	1s
Control packet sizes	6, 8, 46, and 50 bytes
Data packet sizes	43, 47, and 48 bytes

In the next section, different experiments conducted to evaluate the performance of the proposed routing protocol are presented. The motivation behind these experiments is explained with the results and analysis.

5.2 Experiment: Testing DD and Load Balanced DD

In this experiment, the performance of basic Directed Diffusion protocol (DD) [7] is compared with a load balanced directed diffusion protocol. A method of applying load balancing to the basic directed diffusion is developed [11], and is called the "Load Balanced Directed Diffusion with energy piggybacking (LBDD-ED-RD)". To incorporate load balancing in the network, the LBDD-ED-RD method has a mechanism of energy updates in the network. Thus, each node in the network gets energy update (residual energy values) about all of its neighboring nodes. This is done using two different phases of the Directed Diffusion. During the Exploratory Data (ED) phase of the DD, each node is allowed to piggyback its residual energy value onto the Exploratory Data packets (EDPs) and pass it to all of its gradients. In this way all the nodes in the network have residual energy updates about their neighbors. Also, this update is carried on during the Reinforced Data (RD) transmission phase of the DD, which allows nodes on the reinforced path to pass their energy updates to the next-hop node, updating energy at a faster rate. During the Reinforcement stage of the DD, the LBDD-ED-RD method reinforces a single path. The reinforcement however in this case is based on the residual energy of the nodes. Each node would reinforce a neighbor that has the highest residual energy, and eventually a path is constructed towards the source taking local decisions. The basic DD oppose to this method reinforces a path with least-delay and does not consider energy based path construction. The method of energy updates in LBDD-ED-RD is incorporated using basic DD's phases so that there is no extra expenditure of energy. One problem with this scheme is that by locally constructing one path towards the source, there are cases (28% of the trials) when it takes a lot of time (compared to

normal cases) to find the source node and get the data transmission started, degrading protocol performance. The results are presented below. The results of the bad trials are not taken into consideration.

5.2.1 Energy Variance

To account for network wide energy consumption, the residual energy variance of the nodes in the network can be compared for different protocols. This energy variance can be plotted as a function of number of data packets transmitted by the source. This would give an insight about the network lifetime and the degree of load balancing in the network. A protocol which aims to extend the lifetime of the network and minimize energy consumption by incorporating load balancing should have a lower energy variance compared to a protocol which is not efficient enough. The variance in the network is calculated as:

$$sum = \sum_{i=0}^{N} energy_{resd,i} \tag{5.2.1}$$

$$sumSq = \sum_{i=0}^{N} energy^2{}_{resd,i} \tag{5.2.2}$$

$$mean = \frac{\sum_{i=0}^{N} energy_{resd,i}}{N} \tag{5.2.3}$$

$$meanSq = \left(\frac{\sum_{i=0}^{N} energy_{resd,i}}{N}\right)^2 \tag{5.2.4}$$

$$sqMean = \frac{sumSq}{N} = \frac{\sum_{i=0}^{N} energy^2{}_{resd,i}}{N} \tag{5.2.5}$$

Using equation (5.2.4) and (5.2.5)

$$Network\ variance = sqMean - meanSq$$

$$Network\ variance = \frac{\sum_{i=0}^{N} energy^2{}_{resd,i}}{N} - \left(\frac{\sum_{i=0}^{N} energy_{resd,i}}{N}\right)^2 \quad (5.2.6)$$

Equation (5.2.6) gives the expression for computing variance. In this expression, N is the total number of nodes in the network and $energy_{resd,i}$ is the residual energy of node i. Variance is a measure of energy consumption in the network. Lower the variance, lower is the energy consumption. The experiments for energy variance are run with 600 nodes in the network and energy variance for different protocols is recorded for comparison. This variance is logged every 20 s during network operation. For each variance value, the corresponding number of packets transmitted by the source is also logged. These recorded values for 100 different trials are averaged to give statistically significant results.

The energy variance plots for the basic Directed Diffusion and load balanced directed diffusion protocols are compared, shown in Fig. 5.2. It can be seen that using a load balanced DD method, which constructs paths based on residual energy, an improvement is achieved compared to the basic DD. The load balanced DD protocol has a longer lifetime compared to the basic DD. The last sample on plot's x-axis corresponds to simulation stoppage because of any one node failing in the network, and represents the lifespan of a network. The LBDD-ED-RD method has 63% lower residual energy variance at the time of network death, compared to the basic DD. This lower energy variance achieved by LBDD-ED-RD corresponds to lower energy consumption in the network, which is because the protocol tries to distribute traffic load across the network and does not burden a single path (as in DD). By incorporating load balancing, the protocol routes data using different paths which are constructed based on highest residual

energy. Therefore, the protocol extends network lifetime and minimize energy consumption compared to basic DD.

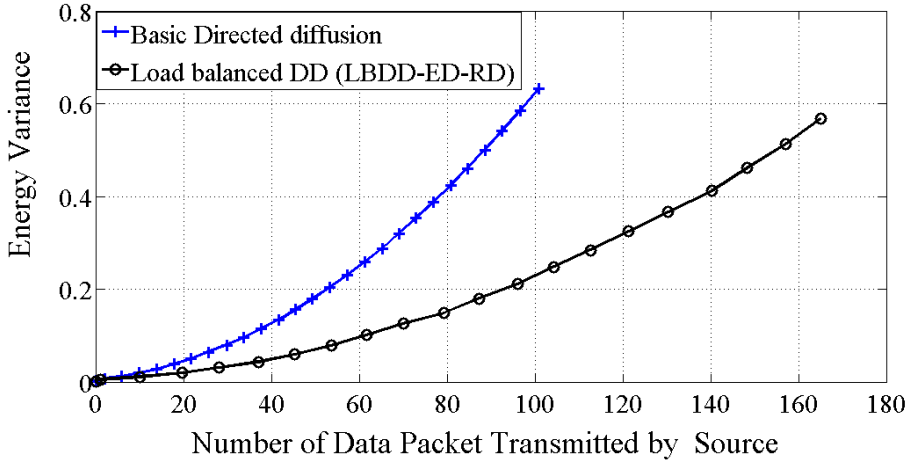

Fig. 5.2 Variance plots for basic DD and load balanced DD methods.

5.2.2 Network Lifetime

Network lifetime is defined as the time at which any one node in the network fails. This network lifetime can be represented by the "number of data packets transmitted by the source" before the simulation stops. These values are recorded every 20 s during network operation. The simulation stops when any one node in the network fails ($energy_{resd} = 0$). In this section and in next sections, network lifetime is tested with different number of nodes in the network from N = 300, 400, ..., 600. For each value of N, the simulation is run 100 times and the values are averaged for statistically significant results. Network lifetime is measured by plotting the total number of data packets transmitted (before a single node dies) as a function of node density (number of nodes).

For the basic DD and LBDD-ED-RD, network lifetime plot is shown in Fig. 5.3. It is clear from the graph that the LBDD-ED-RD method increases network lifetime significantly. This is because the basic DD establishes a single-path for data transmission,

causing nodes along this path to deplete energy sooner, creating partitions in the network, making it dysfunctional at an early stage. The load balanced scheme switches traffic through different paths, incorporating some load balancing in the network, allowing network to operate for relatively longer times. The LBDD-ED-RD method has 65% longer lifetime compared to the basic DD.

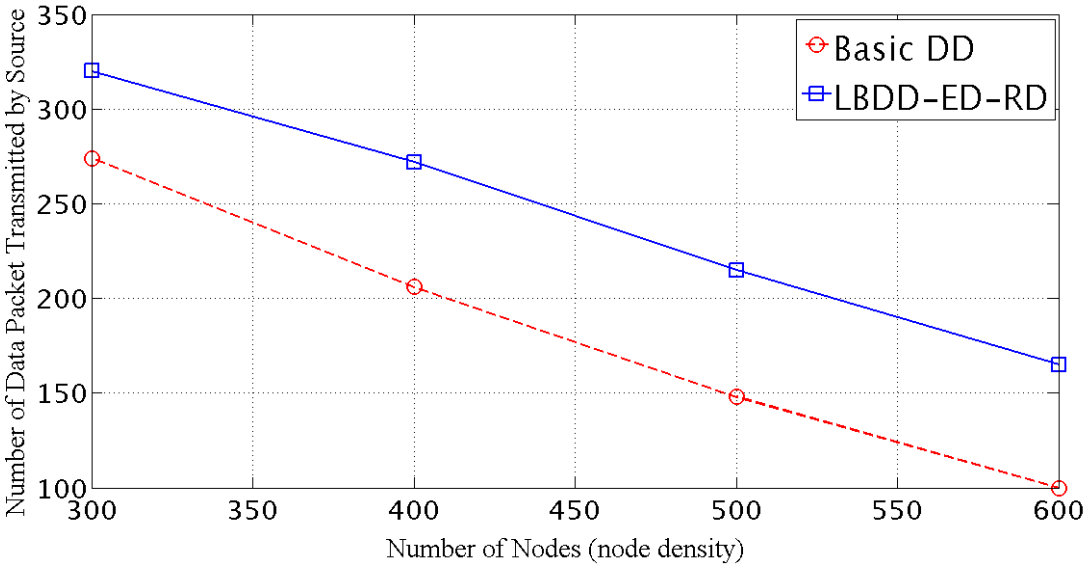

Fig. 5.3 Network lifetime for LBDD vs. DD.

5.2.3 Load Balancing

Load balancing in the network can be visualized by two methods. A simple method is to plot the residual energy of each node to check the overall distribution. Another approach is to use an intensity map for visualizing network on a grid with intensity based energy representation for each node. For the first method, residual energy values of the nodes are recorded every 20 s during network operation. Later on, these residual energy values for each node are plotted for four phases of network operation. For the intensity map, when a node in the network fails, a separate log file is saved for all nodes, with node's coordinates and energy values.

Energy distributions for the basic Directed Diffusion and LBDD-ED-RD are shown in Fig. 5.4 and Fig. 5.5.

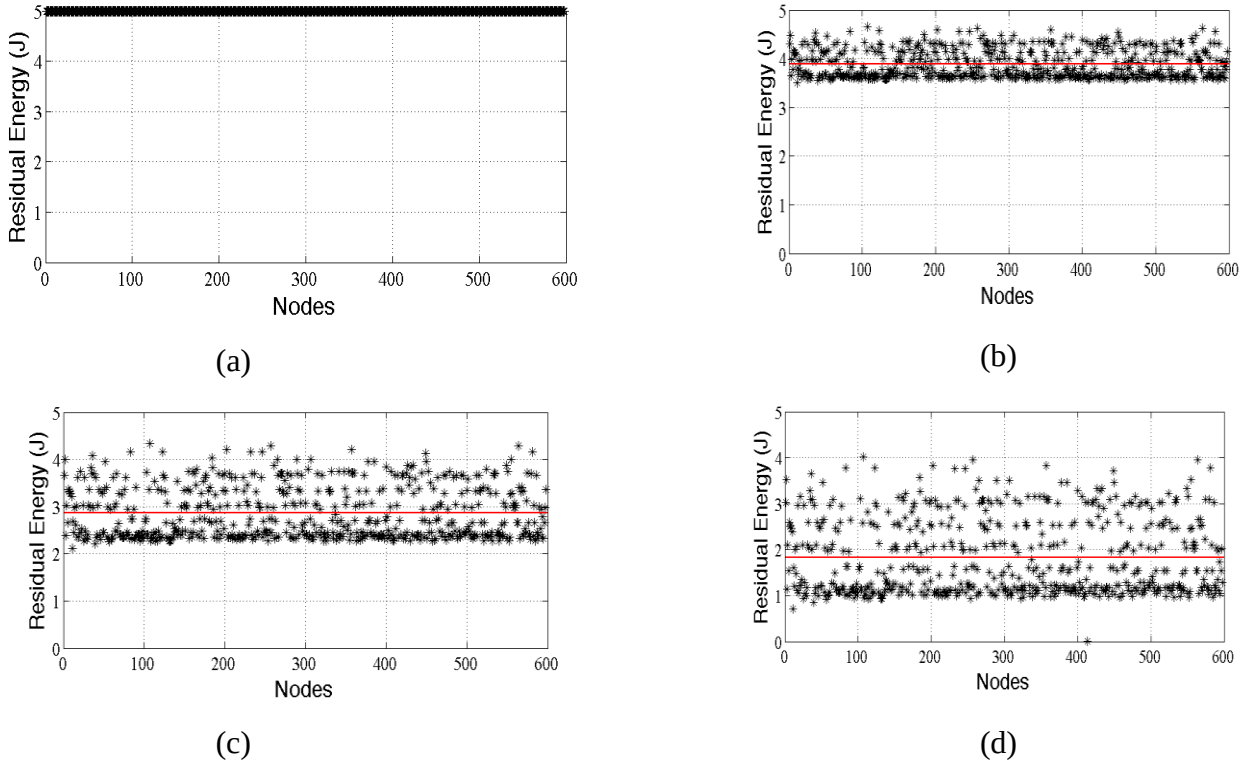

Fig. 5.4 Energy distribution for the basic directed diffusion ((a) start-up, (d) at network failure).

It can be inferred from these figures that the basic DD does not provide load balancing. The network dies (fails) at an early stage (Fig. 5.4(d)), when the nodes have an average residual energy (shown by straight bar in graph) of around half the initial value. It can be seen from Fig. 5.4(d) that node number 415 depleted its energy faster and reached the death floor ($energy_{resd}$ = 0) first, which leads to a partition in the network, making it dysfunctional. This response can be compared with the energy distributions for the LBDD-ED-RD method shown in Fig. 5.5. Although, the load balanced DD tries to incorporate load balancing, it is not efficient. By taking local decisions without having a

global knowledge about the network, this protocol cannot construct optimal paths. The LBDD method also uses a single path for data transmission and does not have a path level load balancing (as in multipath routing).

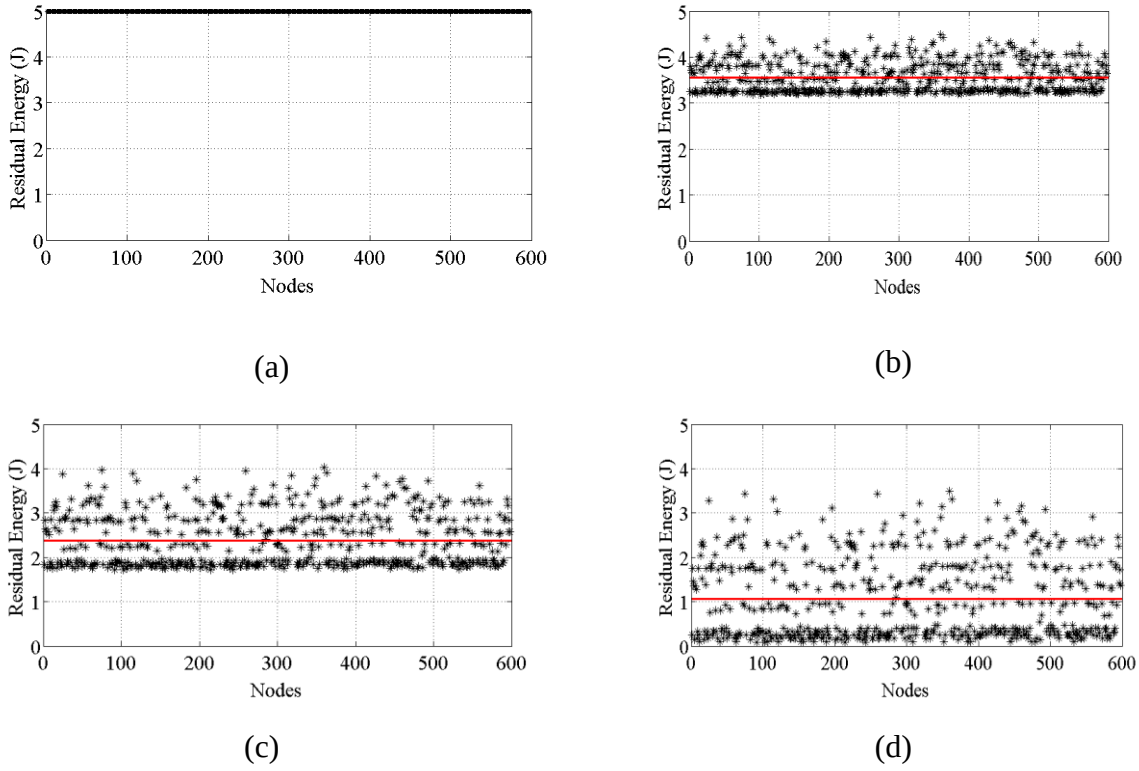

Fig. 5.5 Energy distributions for LBDD-ED-RD method ((a) start-up, (d) network death).

Fig. 5.5 (d) shows an improvement compared to Fig. 5.4 (d), which is that most of the nodes have reached closer to the death floor using LBDD-ED-RD. There is a high density of nodes just above the floor when any one node dies. The results show that locally constructing paths can be useful in implementing load balancing. These results encouraged the use of constructing paths taking local decision (based on high residual energy) in the proposed protocol. However, to implement load balancing to the fullest this dense streak of nodes should be on the floor.

To visualize the path formation and utilization better, an intensity map representation is shown in Fig. 5.6 for the basic DD, and in Fig. 5.7 for the LBDD-ED-RD. In both figures, the network is simulated first by placing sink and source nodes at the extreme ends (fixed configuration). The network is then simulated with random sink-source placement. The node which fails first in the simulation is indicated by a red circle. There are some nodes missing on the map, as the grid is for 625 nodes, and the proposed protocol is simulated for 600 nodes. From Fig. 5.7, it is evident that for the LBDD network more nodes were used in the path formation and were near death (black boxed pixel) when network failed. Whereas, for the basic DD (Fig. 5.6), the maps shows a few nodes near death while others are contrastingly at higher intensity. Another thing to notice here is the outskirts of the network. It can be seen that the path formation takes place mostly in the centre of the network. The nodes at the outskirts of the network are left with high energy and were not used properly.

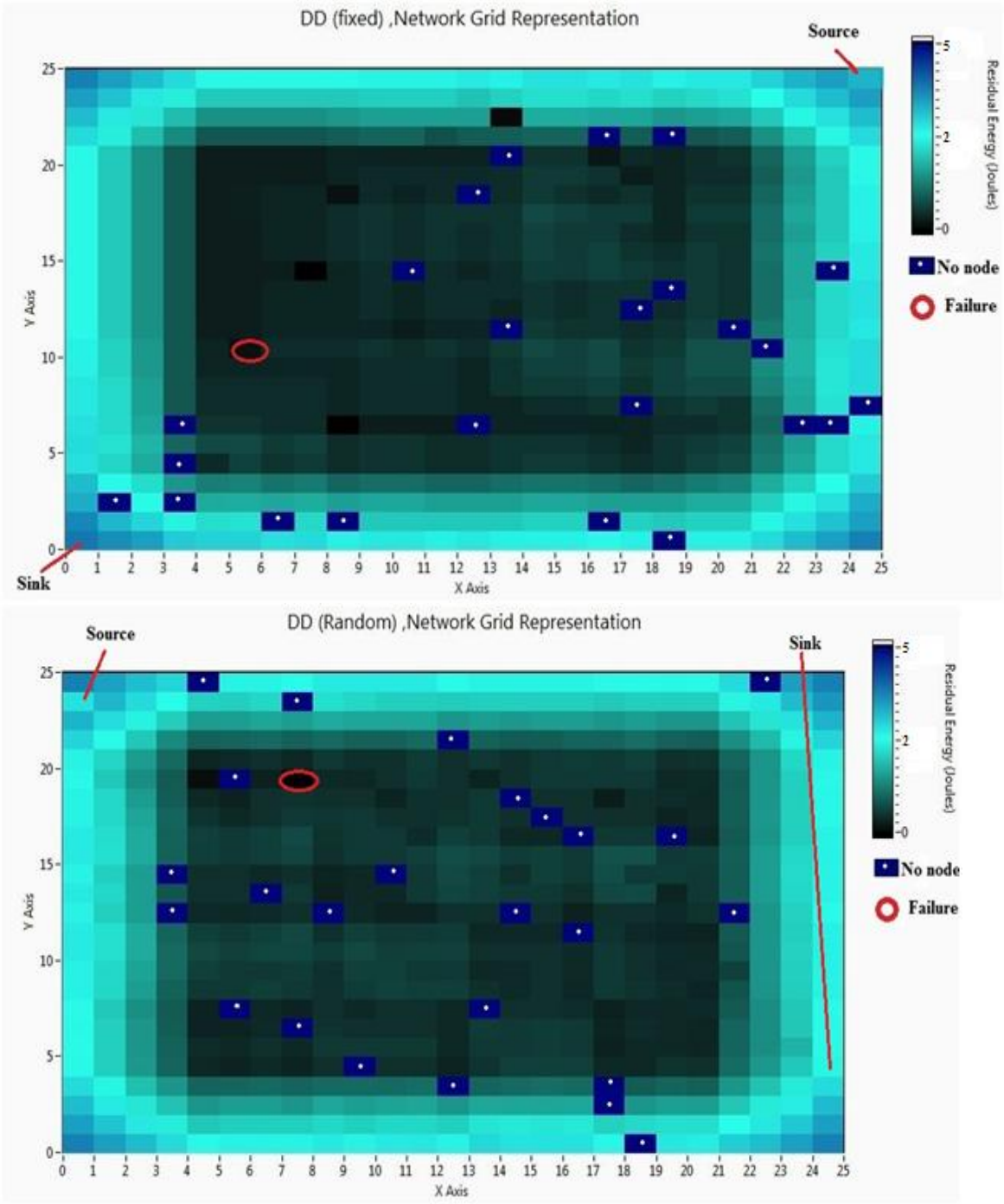

Fig. 5.6 Intensity maps for basic DD, using fixed and random sink-source placement.

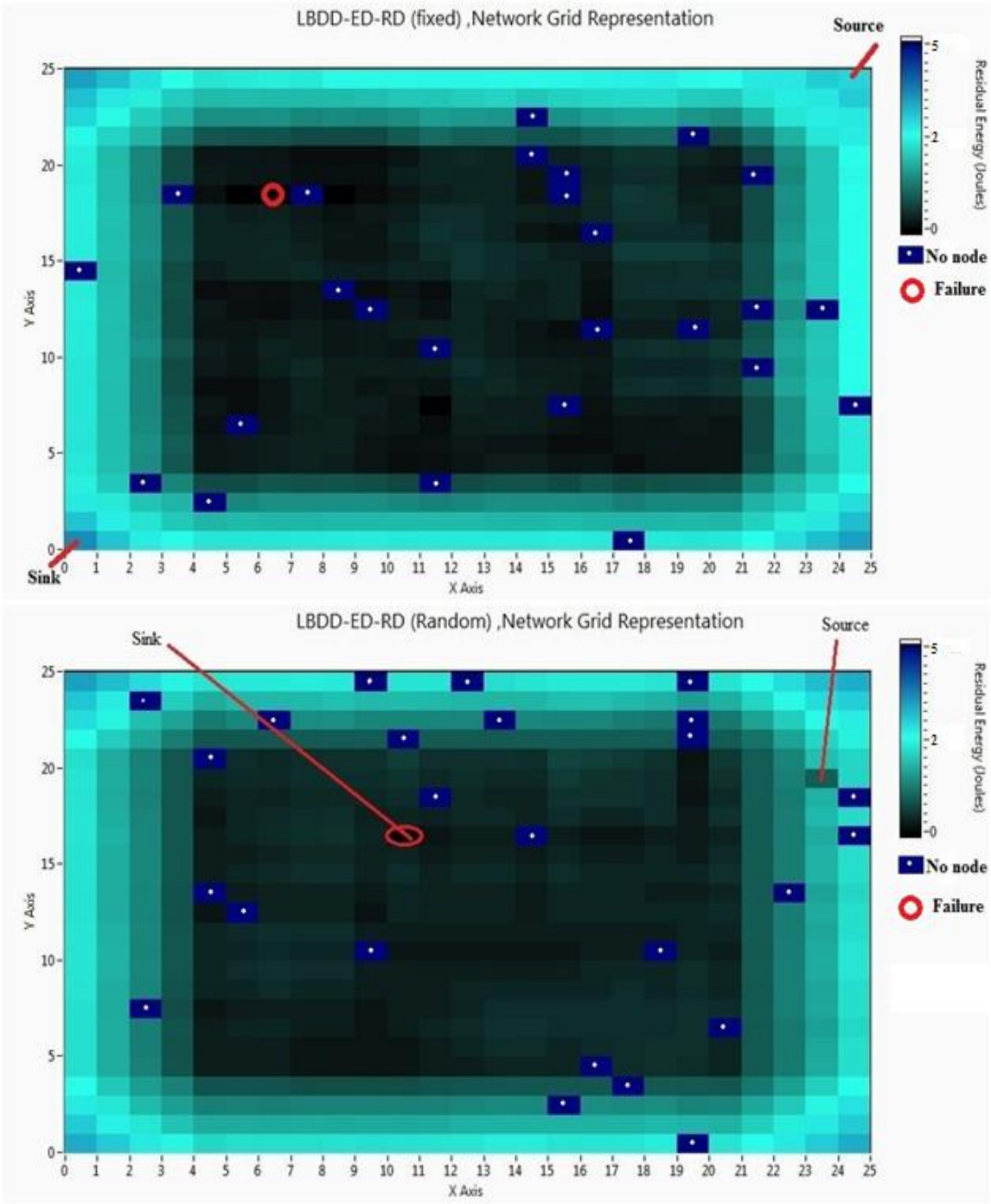

Fig. 5.7 Intensity maps for LBDD, using fixed and random sink-source placement.

5.3 Experiment: Multipath DD and other DD variants

In this experiment, the performance of a multipath routing protocol is compared with the performance of tested single path routing protocols. The method of multipath route construction based on cumulative residual energy (CRE) is applied to basic directed diffusion (DD) [7], and is called the multipath directed diffusion (MDD-CRE) protocol [47]. In this multipath protocol, paths are constructed taking local decisions and source node is allowed to select an optimal path for data transmission. In MDD-CRE protocol, each node receives energy updates about its neighbors using exploratory data and reinforced data phases of directed diffusion (just like LBDD-ED-RD protocol). During the reinforcement phase of the DD, the MDD-CRE method employs multipath route construction. The MDD-CRE protocol allows the sink node to send Reinforcement Message Packet (RMP) to all of its neighbors. The sink passes RMPs to its neighbors, with each RMP having a unique packet ID. These RMPs are then passed on by upstream nodes to one of their neighbors having the highest residual energy value. The RMPs eventually reach the source through multiple paths. The path constructed by the RMP is based on Cumulative Residual Energy (CRE) of the nodes along the path. The source node is given the control to select an optimal path from the set of constructed multiple paths. The CRE value for a constructed path with *n* nodes is given by:

$$CRE = \sum_{i=1}^{n} energy_{resd,i} \tag{5.3.1}$$

The source node selects the path *P* which has the highest CRE value:

$$Path\ for\ Data\ Transmission\ (Source) = max\ \{P_{CRE1}, P_{CRE2}, , \dots, P_{CREn}\} \tag{5.3.2}$$

By doing this, the source selects an optimum path which is constructed based on local decisions. The source node also estimates the CRE value of the used path using equation:

$$CRE = CRE - (HopCount * energy_{TX} + HopCount * energy_{RX}) \quad (5.3.3)$$

By dynamically updating CRE value, the source node keeps searching for a path with higher CRE and routes traffic using alternative paths. The method (MDD-CRE) can be compared with the basic DD, and other DD variant.

5.3.1 Energy variance

The energy variance results for the CRE based multipath method implemented on directed diffusion (MDD-CRE) are compared with DD, and LBDD-ED-RD (Fig. 5.8). It can be seen from Fig. 5.8 that the energy variance of the MDD-CRE method is significantly lower compared to the basic DD and the load balanced DD method. The MDD-CRE method has 85%, and 63% lower residual energy variance compared to the basic DD, and LBDD-ED-RD respectively. By applying multipath routing, the energy consumption across the network is minimized and load distribution across the network is improved. Constructing and selecting paths based on CRE allows the source to choose optimal paths in the network, and balance energy along the unused paths incorporating path-level load balancing. Results show the significance of multipath routing over the single path routing.

5.3.2 Network Lifetime

Network lifetime for different node densities is plotted for MDD-CRE, LBDD-ED-RD, and DD methods in Fig. 5.9. The MDD-CRE method has a longer network lifetime (134%, 42% longer) than basic DD and LBDD-ED-RD methods, respectively. The method of multipath route construction based on CRE when applied to basic directed

diffusion clearly outperforms all other tested variants of directed diffusion. Also, a network with fewer nodes survive longer compared to the one with high node density. Therefore, value of lifetime (number of packets passed before network died) changes considerably with N for DD and its variants.

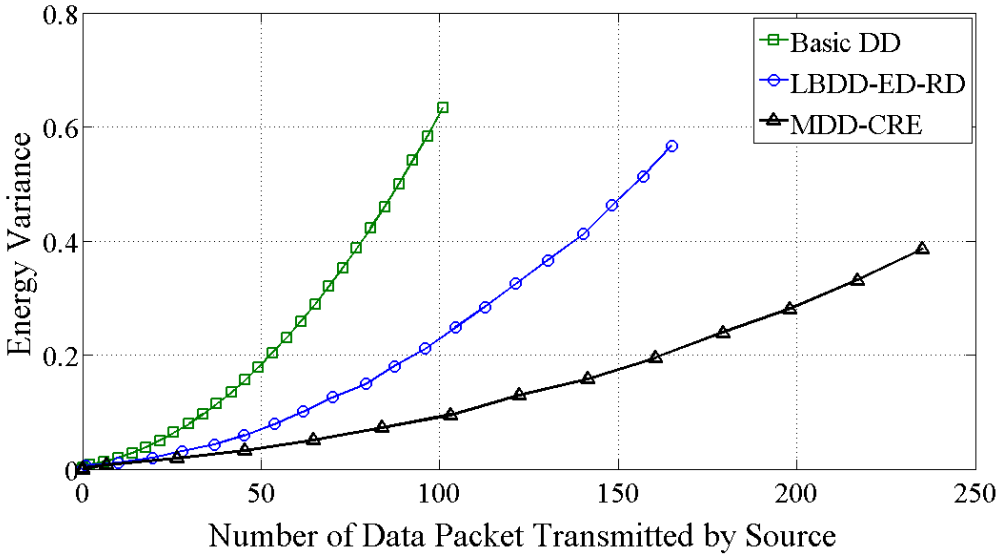

Fig. 5.8 Energy variance plot for MDD-CRE vs. other DD variants.

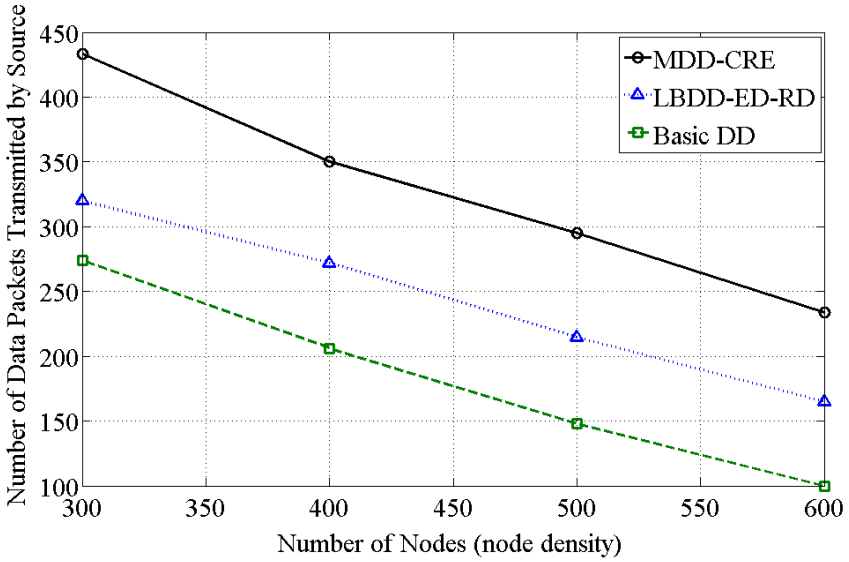

Fig. 5.9 Lifetime plot for DD, LBDD, and MDD-CRE.

5.3.3 Load Balancing

Energy distributions for the MDD-CRE method are represented by Fig. 5.10. It can be seen that the multipath route construction method based on CRE when implemented on directed diffusion incorporates an efficient load balancing. Most of the nodes go down at the same time towards the death floor, and before network death there is a dense streak of nodes (Fig. 5.10 (d)) almost on the floor. This shows that the proposed method incorporates better load balancing compared to the basic DD and LBDD methods (compared from Fig. 5.4 and Fig. 5.5). This is because of the CRE based path construction process which incorporates network level (constructing path based on high residual energy) as well as path level (choosing alternative paths high in CRE) load balancing.

From intensity maps of Fig. 5.11, it can be seen that the intensity contrast is deeper and more nodes have reached near death while being used. The load balancing cause considerable utilization of the paths in the centre of the network. Another thing to notice is that the path formation is concentrated in the centre of the network. This is encouraging because no matter where the source and sink are placed in the network, the load balancing tries to construct different paths around them. The outskirts of the network has the least share in constructing paths. These portions are unused may be because the route construction packets when reaches the corners of the network may not find a node to forward the packet and the packet is dropped there.

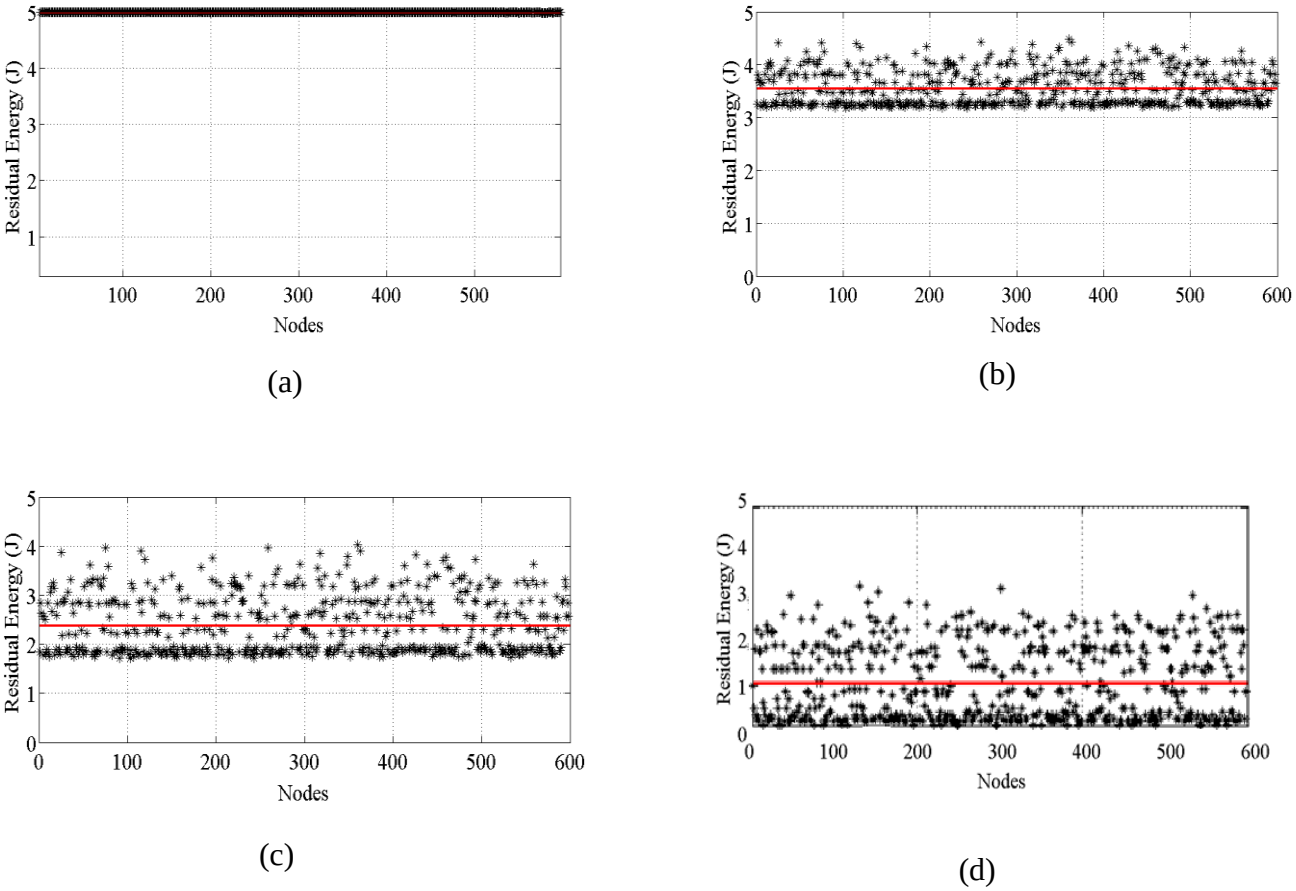

Fig. 5.10 Energy distributions for the MDD-CRE method ((a) start-up, (d) at network failure).

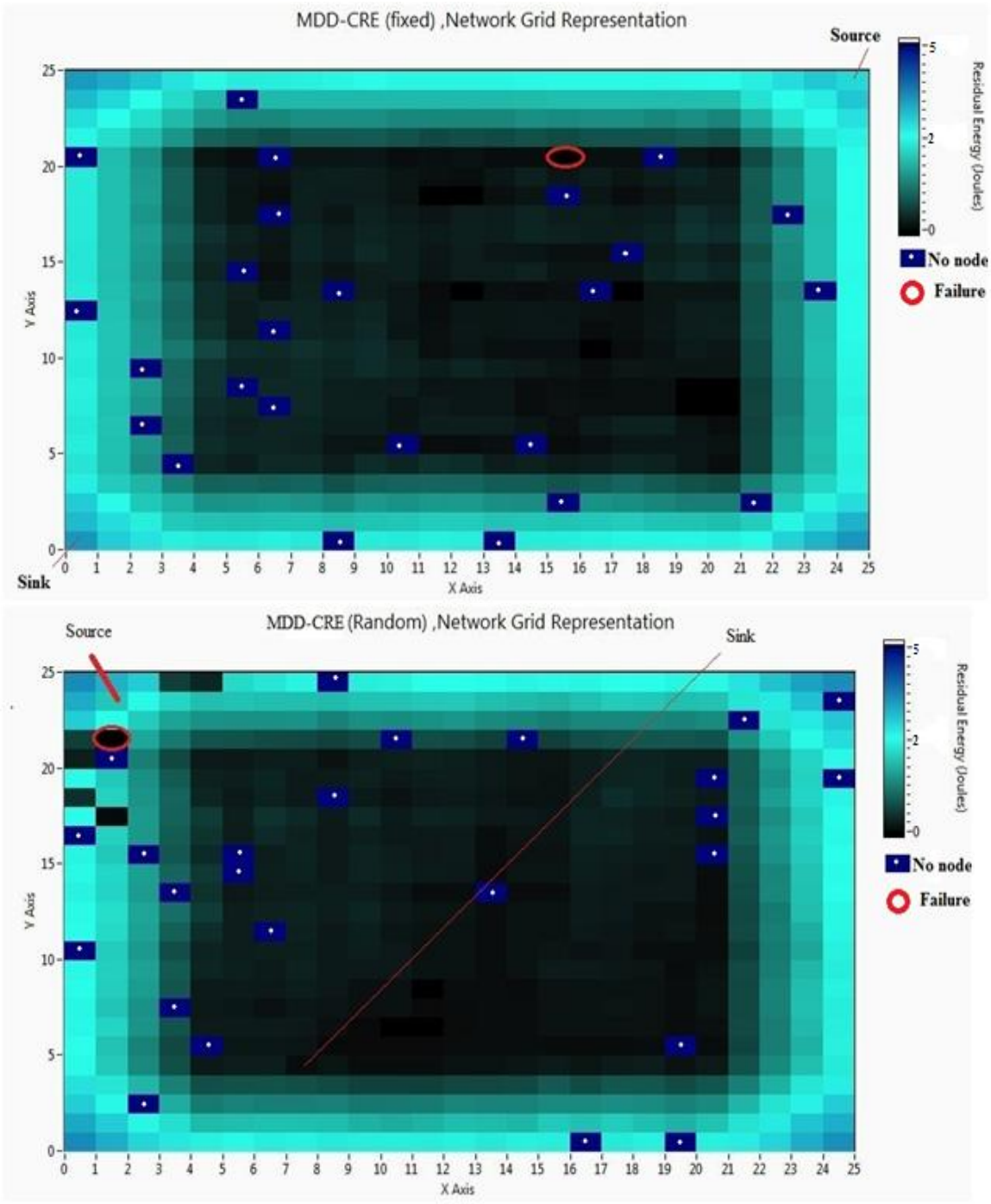

Fig. 5.11 Intensity maps for MDD-CRE, using fixed and random sink-source placement.

5.4 Experiment: Multipath DD based on CRE and Proposed Method based on CRE

The proposed multipath method based on CRE (M-CRE) can be compared with the proposed method which is implemented on directed diffusion (MDD-CRE). The MDD-CRE protocol has four phases (of DD), compared to M-CRE which has three.

5.4.1 Energy variance

The proposed protocol (M-CRE) clearly outperforms the MDD-CRE protocol. As shown in Fig. 5.12, the significant decrease in energy variance for the proposed protocol is because the protocol has three phases and it does not have to spend extra energy for a fourth phase as in MDD-CRE (exploratory phase). By merging route discovery and route construction in one phase, the M-CRE protocol achieves lower network energy consumption. In contrast, the MDD-CRE protocol uses an extra phase for exploring multiple paths, which degrades its performance compared to M-CRE. The energy conservation by integrating route discovery with route construction allows the M-CRE protocol to operate longer times and allows the load balancing to play its role more efficiently. Thus, we can see a significant deviation of energy variance for M-CRE right at the beginning of the simulation compared with MDD-CRE, and this continues till the time MDD-CRE cease to operate. The M-CRE has 68% lower energy variance compared to MDD-CRE. This improvement is remarkable compared to previous results and shows the dominance of the proposed multipath routing protocol.

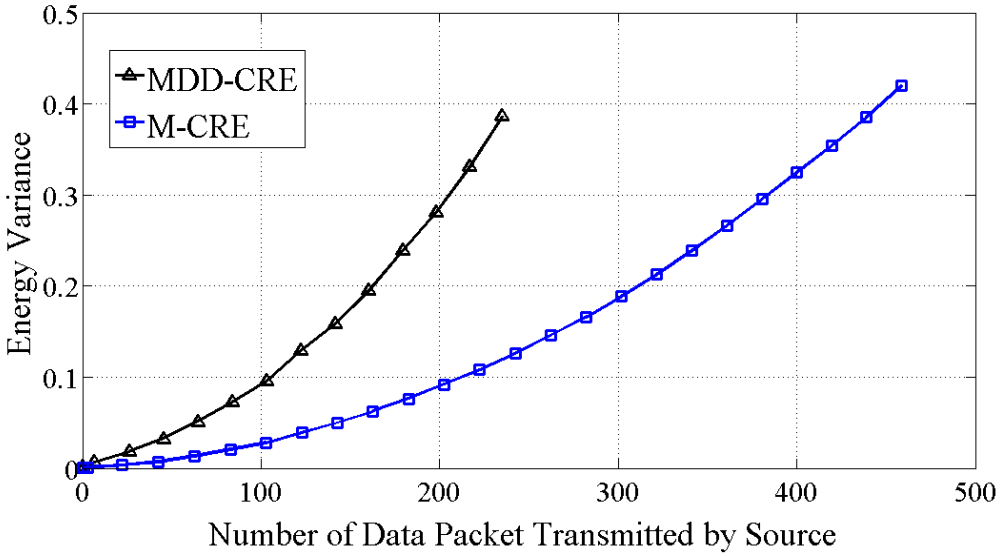

Fig. 5.12 Energy variance plot for M-CRE vs. MDD-CRE.

5.4.2 Network Lifetime

The network lifetime for the proposed multipath method based on CRE (M-CRE) is compared with the same method implemented on basic directed diffusion (MDD-CRE) in Fig. 5.13. The M-CRE method has a 96% longer network lifetime than MDD-CRE method. This big improvement in lifetime for M-CRE is because of two reasons. First, the method constructs paths which have the least probability of failure during data transmission. Secondly, energy savings due to the exemption of a fourth phase, which is not required by M-CRE, and is present in MDD-CRE. From the results it can be inferred that for MDD-CRE, as the node density increases, the lifetime of the network decreases. In contrast, for the proposed multipath protocol M-CRE, the lifetime of the network degrades smoothly with increase in network size. This shows the stability of the protocol with variation in network size.

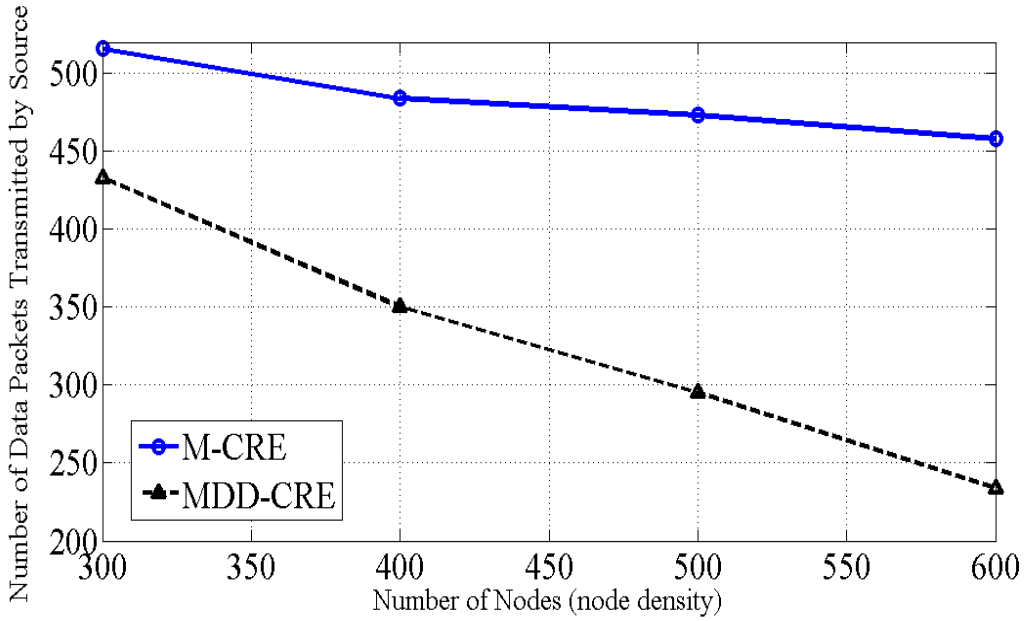

Fig. 5.13 The network lifetime plot for MDD-CRE and M-CRE.

5.4.3 Load Balancing

If we compare the distributions for the M-CRE method (Fig. 5.14) from that of MDD-CRE method (Fig. 5.10), we can see that the load balancing for M-CRE is better. A closer view of the Fig. 5.14(d) would give an idea that most of the nodes are touching the death floor at the same time when any one of them expires, and the average residual energy value (shown by straight bar) is very low. Thus, the M-CRE protocol achieves a better an intense load balancing compared to the same method implemented on DD (MDD-CRE).

From intensity maps of Fig. 5.15, it is clear that load balancing in M-CRE is much improved compared to the previous methods Fig. 5.11, Fig. 5.6, and Fig. 5.7. Also the random placement of the sink-source causes the usage to expand more towards the outskirts of the network, and more nodes are used.

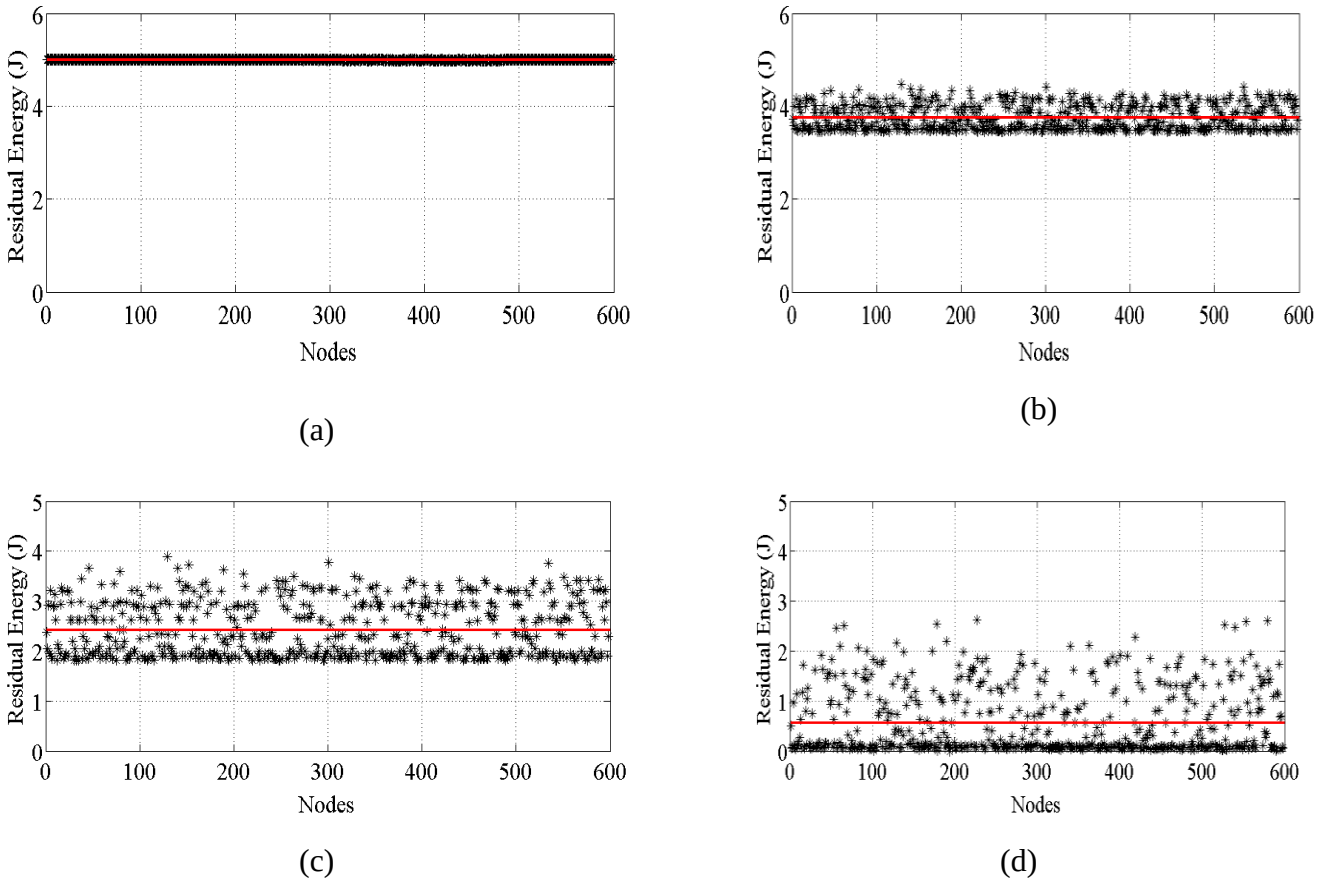

Fig. 5.14 Load balancing distributions for M-CRE method ((a) start-up, (d) at network failure).

The central portion of the network in both configurations has been utilized extensively and a large number of nodes have reached their death at the simulation stoppage time.

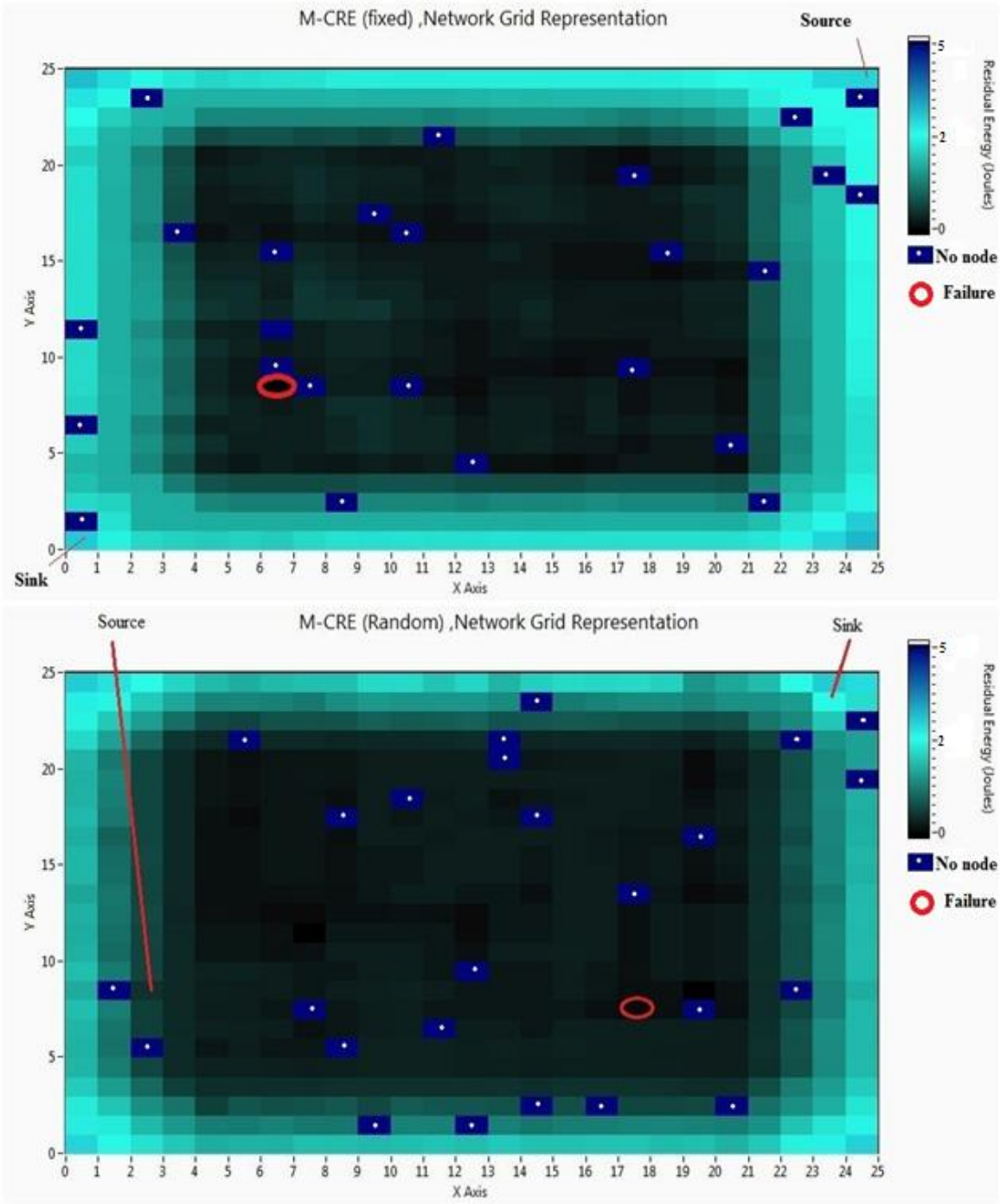

Fig. 5.15 Intensity maps for M-CRE, using fixed and random sink-source placement.

5.5 Experiment: Testing Proposed Methods (CRE versus Variance)

In this experiment, the proposed multipath method based on CRE (M-CRE) is compared with the proposed method based on variance (M-VAR). Both these protocols have almost similar operation except for the data transmission phase. In the data transmission phase the M-CRE selects a path which has the highest cumulative residual energy along the constructed path. However, it was pointed out earlier (in chapter 3) that such a method may choose a path with a weak node, which may fail during data transmission. To prevent failures occurring because of this reason, the proposed method is optimized (M-VAR) to construct paths based on residual energy variance of nodes along the path. The results are presented as follows:

5.5.1 Energy variance

The energy variance results for the two variants of the proposed multipath routing protocol are presented. The performance of M-VAR method is better compared to the M-CRE method, shown in Fig. 5.16. The M-VAR method has a 12% lower residual energy variance at the time of network death, compared to the M-CRE method. At the start both protocols have identical variance curves. This is because during the initial operation of the network, all nodes have enough residual energy to carry on data transmission through them and do not fail. However, as the network operates longer, the M-CRE protocol does not avoid constructing paths which may have a weak node on it and limits its performance. The M-VAR method avoids selecting paths which may have a weak node on it. It can be seen from Fig. 5.16 that this careful selection of paths lowers its energy variance compared to M-CRE. Energy variance curves for all the protocols are shown in

Fig. 5.17. The M-VAR method has 96%, 90%, and 72% lower residual energy variance compared to basic DD, LBDD-ED-RD, and MDD-CRE respectively.

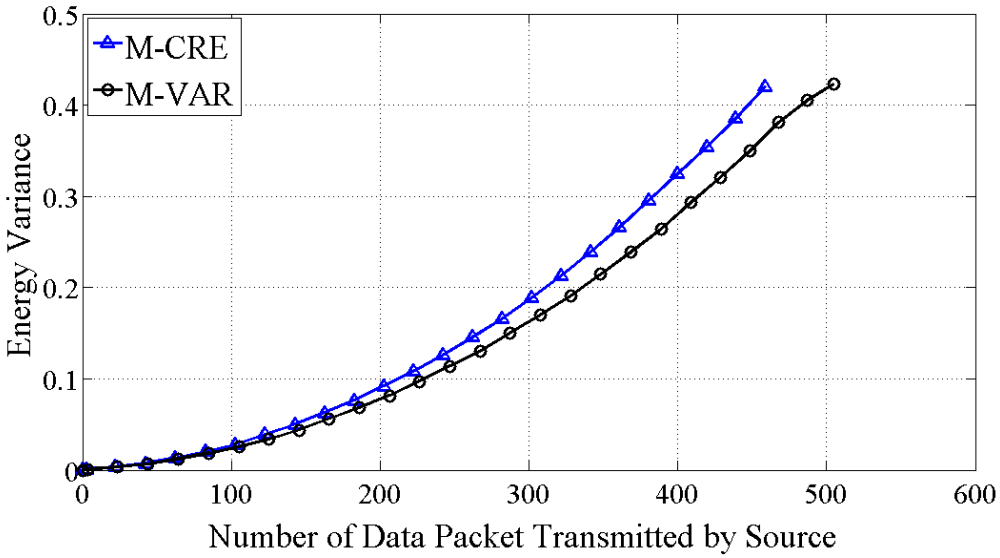

Fig. 5.16 Proposed multipath method based on CRE vs. the method based on Variance.

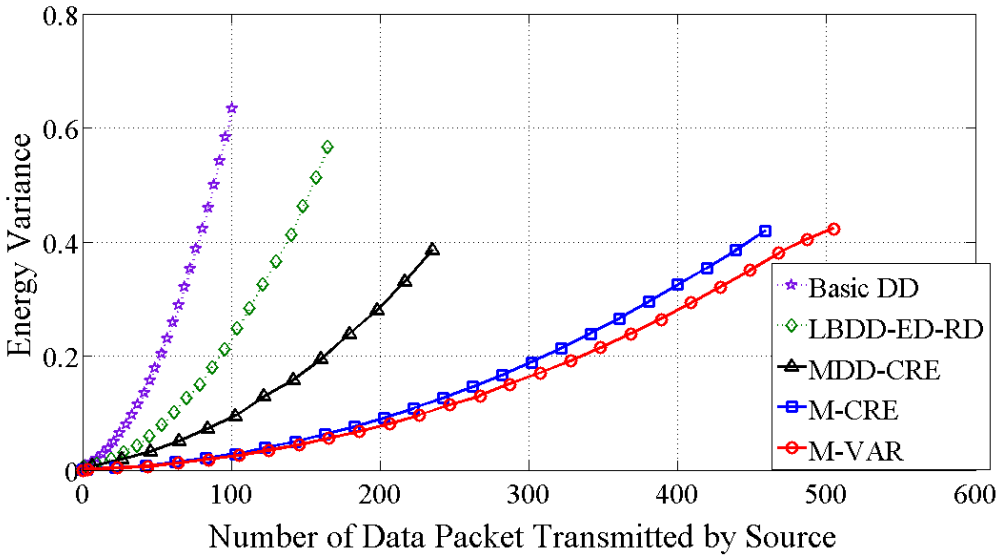

Fig. 5.17 Energy variance of different protocols.

5.5.2 Network Lifetime

Network lifetime for the two variants of the proposed method is shown in Fig. 5.18. The M-VAR is slightly superior in performance and has a longer lifetime compared to M-CRE. This improvement in the lifetime for M-VAR is because of the fact that the protocol tries to avoid paths which may have a weak node (which will die sooner), unlike M-CRE which cannot filter out such paths. The M-VAR method has 10% longer lifetime compared to M-CRE method. The M-VAR lifetime is 404% longer compared to the basic DD.

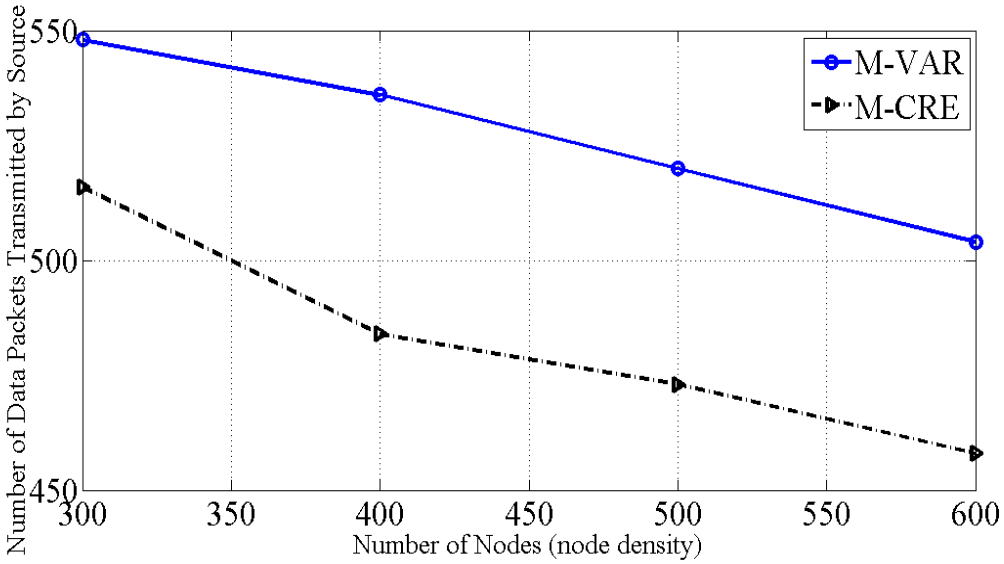

Fig. 5.18 Network lifetime plot for M-VAR and M-CRE.

An important thing to notice in Fig. 5.18 is that for multipath routing protocols the life of the network (packets transmitted before network death) varies slowly with network size (node density). This shows that the proposed multipath protocols are stable with variation in network size.

5.5.3 Load Balancing

For the proposed method of M-VAR, load balancing is relatively better than the M-CRE method. The average residual energy value for the M-VAR (Fig. 5.19(d)) is lower compared to the M-CRE average residual energy value (Fig. 5.14(d)). Also, a large number of nodes are sitting on the death floor right at the network failure, which shows the efficiency of load balancing (Fig. 5.19(d)).

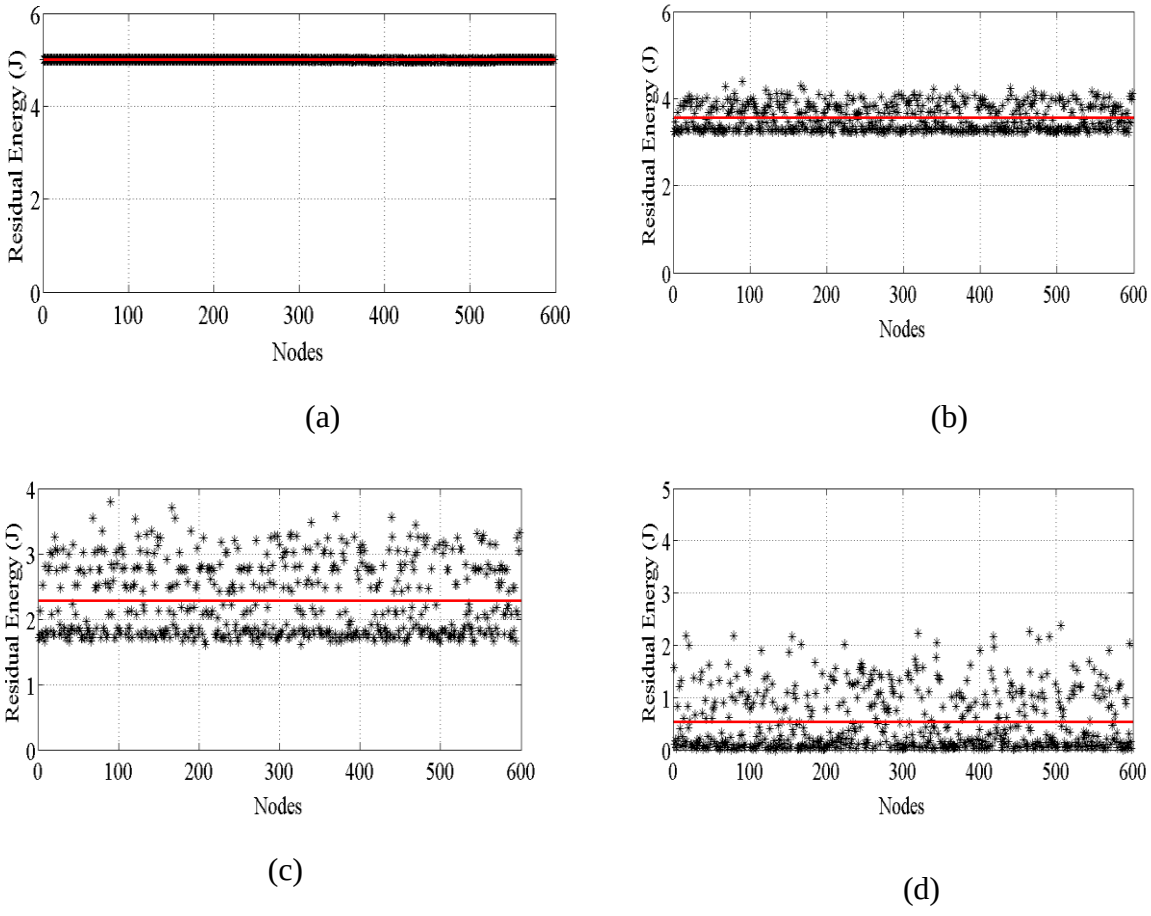

Fig. 5.19 Energy distributions for the M-VAR method.

Intensity map for M-VAR method is shown in Fig. 5.20. By comparing with other DD based schemes, it can be seen that the network has achieved higher degree of load balancing and large number of paths are constructed in the centre of the network.

From all intensity maps, it is seen that source and sink does not fail frequently. This is because they may not be involved in passing more data compared to an intermediate node used more often. From all the maps, it is clear that when source and sink are place on the extreme ends, the failure occurs in the centre, and when place randomly the failure is near the source or the sink.

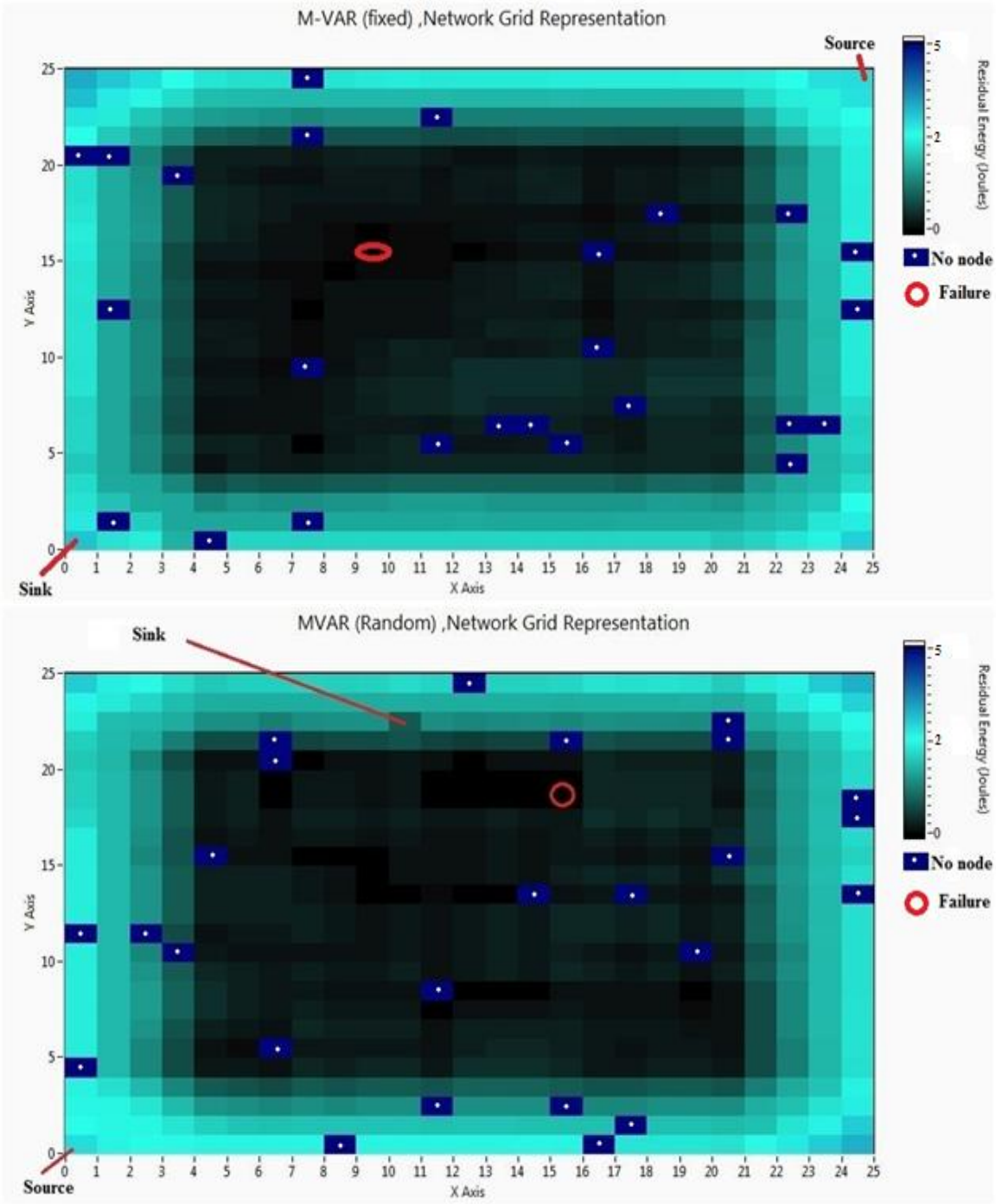

Fig. 5.20 Intensity maps for M-VAR, using fixed and random sink-source placement.

5.6 Cost Analysis and Multiple Paths

The proposed method can construct multiple paths during route construction phase. However, these paths are constructed at some energy cost. To select an optimum number of multiple paths, an experiment was conducted to see different costs associated with these multipaths.

In the experiment 600 nodes are placed in the network. The source and sink nodes are placed at the extreme ends to allow maximum number of hops in between for each path. By sending Route Construction Packets (RCPs) through different number of paths, their energy cost is measured. Energy costs for each RCP round are recorded and the recorded values for 10 experiments are averaged. This measured "energy cost" is plotted as a function of "number of paths", shown in Fig. 5.21. The cost of RCP is computed by considering the number of hops taken from the sink towards the source, using equation (5.6.2).

$$\text{Cost} = E_{Tx}(k, d) * hopCount + E_{Rx}(k) * hopCount \tag{5.6.1}$$

$$\text{Cost} = hopCount * (E_{elec} * k + \epsilon_{amp} * k * d^2 + E_{elec} * k) \tag{5.6.2}$$

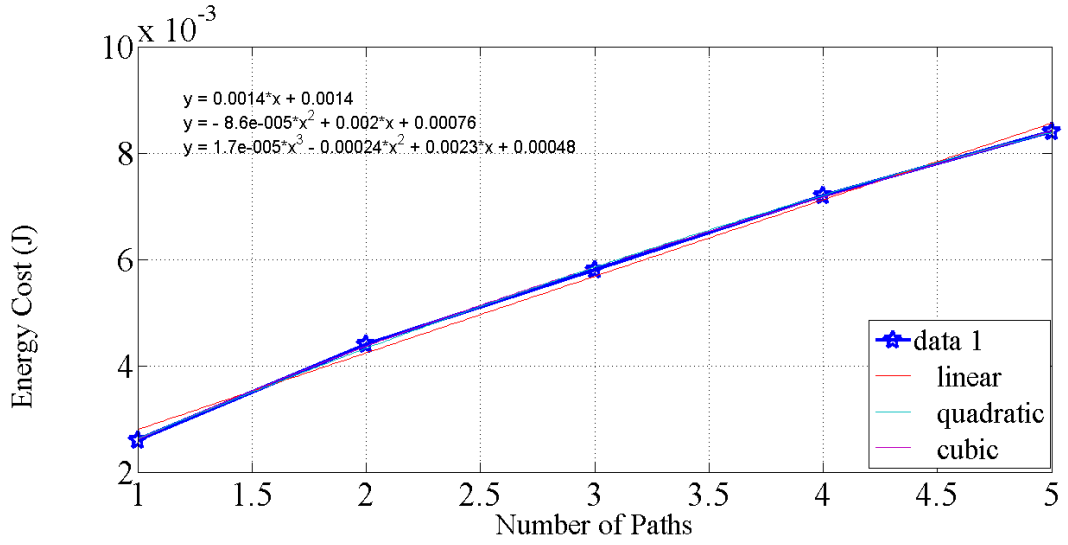

Fig. 5.21 Cost analysis for multiple paths.

The cost value can be associated simply with a linear curve and the expression for computing it is given by:

$$C(m) = 0.0014 * m + 0.0014 \quad (5.6.3)$$

Where, *C* is the cost of constructing *m* multiple paths.

Since the energy cost of sending the route construction packets increases linearly with the number of paths, the proposed algorithm was initially tested with two paths, to minimize energy consumption and test the protocol efficiency for the minimum number of multipaths. It was conjectured that using a 2-path route search mechanism, the energy consumption would be lower in the network, which would lower its energy variance.

It was found through experiments that a "2-path route construction" degrades the performance of the protocol compared to using n-multiple paths. In a 2-path route construction, the sink sends two route construction packets (RCPs) to two of its neighbors having highest residual energy. These RCPs then constructs two paths towards the source. Using a "2-path route construction" has two disadvantages. Firstly, the source node cannot choose an optimal path from a set of two paths. Secondly, when these two RCPs propagate towards the source in a dense network, there are chances of them getting dropped at some point. This would cause the network to take longer times to construct a path, increasing energy consumption, decreasing its lifetime.

In the "n-path route construction", the sink sends RCPs to all of its neighbors, which are then passed on by nodes to their neighbor with high residual energy. Multipath route construction allows the source to choose a path from a set of good paths and incorporate efficient load balancing. Also, by sending multiple RCPs each route construction phase, the load is distributed across the network. This is in contrast to the

case where RCPs propagate through 2-paths, and only the nodes along these paths dissipate energy for passing these RCPs, stressing only certain nodes in the network during route construction. The results for the M-CRE and M-VAR methods are shown in Fig. 5.22 and Fig. 5.23.

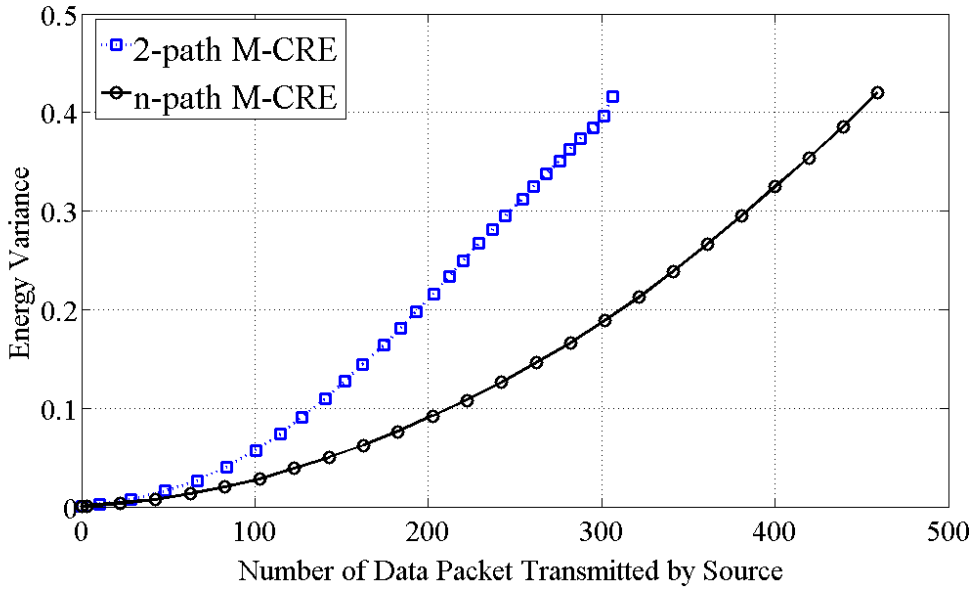

Fig. 5.22 Variance plots for M-CRE using 2-path and n-path route construction.

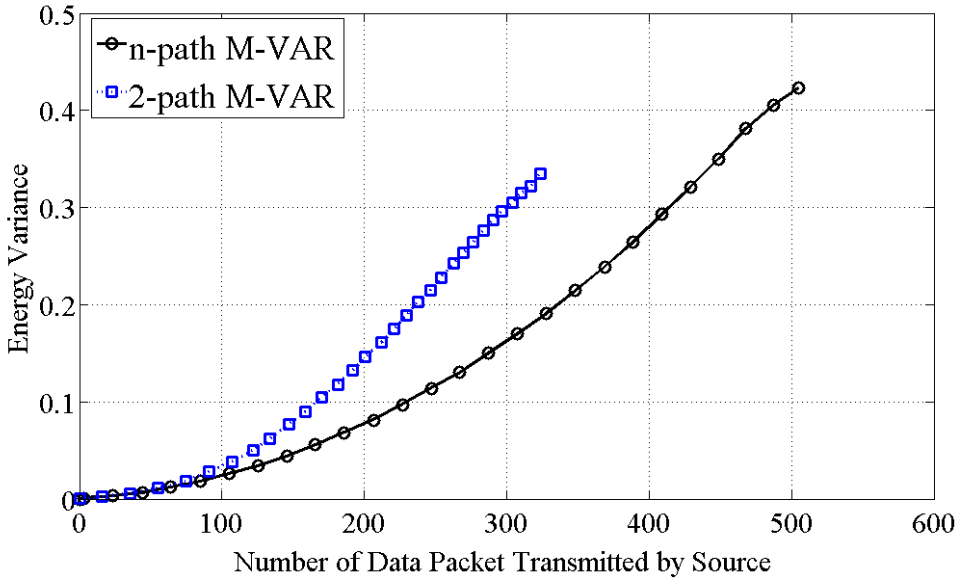

Fig. 5.23 Variance plots for M-VAR using 2-path and n-path route construction.

From results it is clear that for the proposed protocols (M-VAR and M-CRE), multipath route construction (n-path) maximizes network lifetime by incorporating efficient load distribution and minimizing energy consumption in the network. The cost of constructing multiple paths is balanced out by the load distribution RCP propagation offers. For the M-CRE protocol, using "n-path route construction" gives 52% lower energy variance compared to the "2-path route construction". The lifetime for "n-path" model is 51% longer than the "2-path" model. For the M-VAR protocol, using "n-path route construction" gives 47% lower energy variance compared to the "2-path route construction". The lifetime for "n-path" model is 54% longer than the "2-path" model.

5.7 Node Energy Consumption

To find the average energy consumed by the node to transmit a packet between the source and the sink node, the metric used is given as:

$$Node\ energy\ consumption = \frac{\sum_{i=1}^{N}\left(energy_{init,i} - energy_{resd,i}\right)}{N\sum_{j=1}^{S} data_j} \tag{5.7.1}$$

Where N is the total number of nodes in the network, $energy_{init,i}$ is the initial energy of node i, S is the number of sink in the network, and $data_j$ is the data packets received by sink j. The metric is used in [7][45], and computes energy efficiency of the protocol. The results for the proposed multipath protocol (M-VAR) and basic Directed Diffusion are shown in Fig. 5.24. It is clear from the results that the proposed multipath protocol has lower node energy consumption compared to the Directed Diffusion protocol. It can be seen that using the M-VAR protocol, a node can have stable lower energy consumption even when the network size varies. In contrast, for the Directed Diffusion the node

consumption increases with increase in network size. This shows the energy efficiency of the proposed multipath routing protocol.

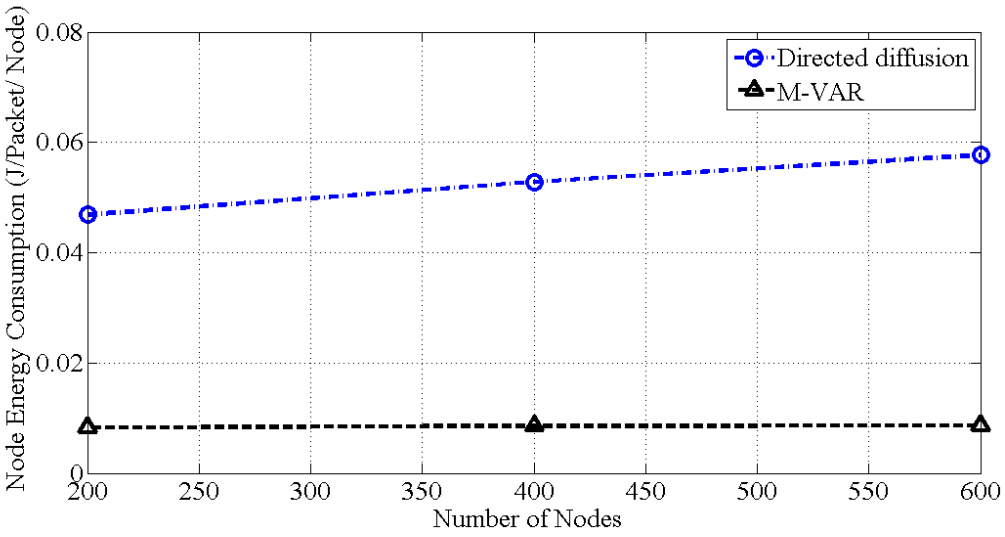

Fig. 5.24 Average node energy consumption (j=1).

5.8 Simulator verification

A lot of work has been put in to verify proper operation of the simulator. To do this, a trace file is generated by the Eclipse IDE when the simulation is run. This file is used to determine proper functioning of the simulator. All the events, data lists, flags, packet contents, transmissions and receptions by a node, and packet processing are printed and logged in this trace file. The trace files can be viewed to know how the network simulator is behaving and validate its operation. The trace file is also shown in Eclipse console while the simulation is running. An example of run-time trace file output in the Eclipse console is shown in Fig. 5.25.

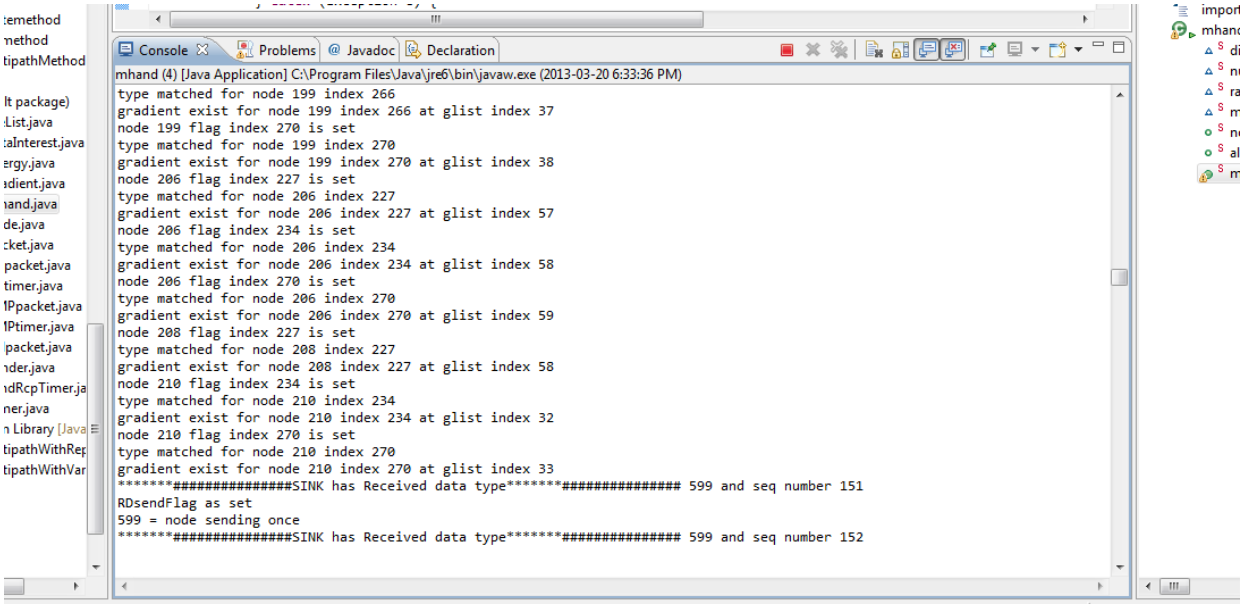

Fig. 5.25 Run time output of trace file in Eclipse console.

5.9 Summary

The experimental results have shown that the proposed method of multipath route construction based on variance (M-VAR) outperforms all other methods. The proposed algorithm M-VAR has lower residual energy variance (96%, 90%, 72%, 12% less) and longer network lifetime (404%, 205%, 115%, 10%) than basic Directed Diffusion, load-balanced Directed Diffusion (LBDD-ED-RD), multipath Directed Diffusion (MDD-CRE), and the proposed algorithm M-CRE, respectively.

Chapter 6

Conclusions and Future Work

In this thesis an energy-efficient multipath routing protocol is presented. The proposed protocol tries to rectify problems associated with single path data-centric routing protocols for WSNs. A problem with a single-path scheme is the premature network death, because of the burdening of a single path used. In the thesis, this problem of single path routing is analyzed for the basic Directed Diffusion [7]. Also, a solution, such as load balancing, to extend the lifetime of the network for such routing protocols, is presented. It is shown through experiments that load balancing in a single path routing can extend the lifetime of a network. The proposed multipath routing protocol develops on this work of load balancing. The proposed method of multipath route construction shares the incrementally accumulated knowledge of the constructed paths with the source node and allows it to select an optimal path. The proposed method uses alternative path routing and filters out a poor quality path (e.g., with weak node on it). The proposed method was also implemented on traditional single-path protocols such as Directed Diffusion to show its efficiency. In the following sections, contributions of the work and future directions are given.

6.1 Conclusions

The thesis presents a multipath routing protocol with two variants. Although, both variants perform well in terms of energy-efficiency, the method of route construction using variance (M-VAR) slightly outperforms the method based on Cumulative Residual Energy (M-CRE).

This work was developed by first analyzing the problems with traditional single-path routing protocols. An effort was made to make data-centric protocols, such as Directed Diffusion, energy-efficient. In this regard a novel method of load balancing was incorporated in the protocol. This load balancing method was implemented during the route construction phase of the Directed Diffusion (i.e., during the reinforcement phase), where each node selects a neighbor based on its residual energy (highest one). An improvement in the performance of load balance Directed Diffusion motivated the use of this idea to create a multipath routing protocol.

The proposed multipath routing protocol is data-centric and consists of three phases. The protocol uses multipath routing to construct multiple paths and uses only one of them for data transmission. By searching for multiple paths periodically and allowing the source node to select the best path, the protocol improves the energy-efficiency of the network, thus extending its lifetime. The goals achieved in doing this work are:

- Development of a multipath route construction algorithm. The algorithm develops a mechanism of energy updates in the network allowing each node to extract energy values of its neighbors from a radio transmission. Using this mechanism, each node can select a next-hop node based on residual energy.
- The protocol incorporates load balancing in the network, which extends network lifetime and minimizes energy consumption at the node. This load balancing is incorporated during route construction and path selection phases.
- The multipath protocol integrates route construction with route discovery, which helps in conserving energy. The paths constructed by the protocol are loop-free,

and have minimum latency (compared to looped paths). These paths are also good quality path (in terms of energy).

- To better visualize the network for its energy consumption, a visualization tool is developed in LabVIEW, which can be used for analyzing sensor network energy data.
- A Java based simulator is developed for analyzing data-centric protocols.
- The results show that the proposed algorithm M-VAR has lower residual energy variance (96%, 90%, 72%, 12% less) and longer network lifetime (404%, 205%, 115%, 10%) than basic Directed Diffusion, load-balanced Directed Diffusion (LBDD-ED-RD), multipath Directed Diffusion (MDD-CRE), and the proposed algorithm M-CRE, respectively.

6.2 Future Work

The following suggests future work to improve the routing protocol:

- In some cases, nodes are mobile. The proposed method considers stationary nodes. A mobility algorithm needs to be incorporated in the protocol to cater the neighbor update mechanism when nodes are mobile. This would extend the protocol for mobile nodes as well.
- The protocol constructs paths based on energy and does not considers the delay. The protocol needs to be refined for delay constrained applications. More research needs to be conducted to find out a balance between delay and energy based routing.
- The load balancing performed by the protocol needs to be maximized. In the results we could see that there still are some nodes with high residual energy in

the network. A technique which would take load balancing closer to the ideal load balancing needs to be developed.

- The protocol can be tested on a hardware platform to determine practical challenges and limitations.
- The multipath protocol can also be tested on a cross layer platform.
- The repair mechanism in the routing protocol needs to be exhaustively tested and characterized.
- The proposed protocol was tested with multiple sources in the network. However, there is still a need for comprehensive evaluation. The network needs to be tested with multiple sinks and sources.

References

[1] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA'02*, Atlanta, Georgia, 2002, pp.88-97.

[2] X. Jiang, G. Zhou, Y. Liu, Y. Wang , "Wireless sensor networks for forest environmental monitoring," in *2nd Int. Conf. on Information Science and Engineering*, 2010, pp. 2514-2517.

[3] http://www.alertsystem.org

[4] B. A. Warneke, and K. S. J. Pister, "MEMS for distributed wireless sensor networks," in *Ninth Int. Conf. on Electronics Circuits and systems*, 2002,pp. 291-294.

[5] S. C. Ergen, P. Varaiya, "On multi-hop routing for energy efficiency," *IEEE Commun. Lett.*, Vol. 9, Issue 10, pp. 880-881, 2005.

[6] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc. of the 5th ACM/IEEE MobiCom'99*, Seattle, WA, 1999, pp. 174-185.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proc. of ACM MobiCom'00*, Boston, MA, 2000, pp. 56-67.

[8] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy efficient communication protocol for wireless microsensor networks," in *33rd Annual Hawaii Int. Conf. on System Sciences*, 2000, pp. 3005-3014.

[9] Y. Xu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks," UCLA Computer Science Dept., Tech. Rep. UCLA/CSD-TR-01-0023, May 2001.

[10] Y. Chen, and Q. Zhao, "On the lifetime of wireless sensor networks," *IEEE Commun. Lett.*, vol. 9, no. 11, pp. 976–978, 2005.

[11] S. Wijedasa, S. Rizvi, and K. Ferens, "Load balancing algorithms for wireless sensor networks," in *Proc. Int. Conf. on Wireless Networks*, Las Vegas, Nevada, 2012, pp. 61-67.

[12] A. N. Eghbali, and M. Dehgan, "Load-balancing using multipath directed diffusion in wireless sensor networks," in *Proc. of 3rd Int. Conf. on mobile ad-hoc and sensor networks*, 2007, pp. 44-55.

[13] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," in *ACM mobile comput. and commun. Review*, vol. 5, no. 4, 2001.

[14] J. N. Al-Karaki, and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *IEEE Wirel. Commun.*, vol. 11, no. 6, pp. 6-28, 2004.

[15] I. F. Akyildiz, M. C. Vuran, O. B. Akan, and W. Su, "Wireless sensor networks: A survey revisited," *Computer Networks Journal (Elsevier)*, 2005.

[16] N. F. Maxemchuk, "Dispersity routing," in *Proc. ICC '75*, San Francisco, CA, 1975, pp. 41-10, 41-13.

[17] N. F. Maxemchuk, "Dispersity routing in high-speed networks," *Comp. Networks and lSDN Sys.*, vol. 25, no. 6, pp. 645-661, 1993.

[18] N. F. Maxemchuk, "Dispersity Routing on ATM Networks," in *Proc. IEEE Int. Conf. on Computer Commun.*, San Francisco, CA, 1993, pp. 347-57.

[19] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," in *Proc. of Conf. on Commun. Architecture and Protocols ACM SIGCOMM*, Zurich, 1991, pp. 43-51.

[20] J. Chen, P. Druschel, and D. Subranianian, "An efficient multipath forwarding method," in *Proc. of IEEE Int. Conf. on Computer Commun.*, San Francisco, CA, 1998, pp. 1418-1425.

[21] S. Murthy, and J. J. Garcia-Luna-Aceves, "Congestion-oriented shortest-multipath routing," in *Proc. of IEEE Int. Conf. on Computer Commun.*, San Francisco, CA, 1996, pp. 1028-1036.

[22] R. Ogier, V. Rutenburg, and N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths," *IEEE Trans. on Information Theory*, vol. 39, no. 2, pp. 443-455, 1993.

[23] G. D. Caro, F. Ducatelle, and L. M. Gambardella, "AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks," in *Proc. of Parallel Problem Solving from Nature VIII*, pp. 461-470, LNCS 3242, Springer, 2004.

[24] M. Roth, and S. Wicker, "Termite: A swarm intelligent routing algorithm for mobile wireless ad-hoc networks," *Springer SCI Series: Swarm Intelligence and Data Mining*, 2005.

[25] M. Radi, B. Dezfouli, K. A. Bakar, and M. Lee, "Multipath routing in wireless sensor networks: survey and research challenges," *Sensors J.*, vol. 12, no. 1, pp. 650–685, 2012.

[26] M. Z. Zamalloa, B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links," *ACM Trans. Sens. Netw.*, 2007, 3, doi:10.1145/1240226.1240227.

[27] M. R. Pearlman, Z. J. Haas, P. Sholander, S. S. Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc networks," *In Proc. of the 1st Annual Workshop on Mobile and Ad Hoc Networking and Computing*, Boston, MA, 2000, pp. 3–10.

[28] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: A survey," *IEEE Commun. Surveys and Tutorials*, Vol 11, 2009, pp52-73.

[29] J. Rehana, "Security of wireless sensor networks", *Seminar on Internetworking*, April 27, 2009.

[30] Y. W. Law, J. Doumen, and P. Hartel, "Benchmarking block ciphers for wireless sensor networks," in *IEEE Int. Conf. on Mobile Ad-Hoc and Sensor Systems*, 2004, pp. 447-456.

[31] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Commun. Surveys and Tutorials*, 2006.

[32] J. Zhang, Y. Lin, M. Lin, P. Li, and S. Zhou, "Curve-based greedy routing algorithm for sensor networks," Lecture Notes in Computer Science, vol. 3619, p. 1125, 2005.

[33] J. Zhang, Y. Lin, M. Lin, P. Li, and S. Zhou, "Curve-based greedy routing algorithm for sensor networks," *Lecture Notes in Computer Science*, vol. 3619, pp. 1125-1133, 2005.

[34] J. Deng, R. Han, and S. Mishra, "Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks," in *Proc. of the 2004 IEEE Int. Conf. on Dependable Systems and Networks* , 2004, pp. 637-646.

[35] P. C. Lee, V. Misra, and D. Rubenstein, "Distributed algorithms for secure multipath routing," in *Proc. of IEEE Int. Conf. on Computer Commun*, 2005, pp.1952-1963.

[36] S. Dulman, T. Nielberg, J. Wu, and P. Havings, "Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks," WCNC workshop, New Orleans, LA, Mar. 2003.

[37] B. Deb, S. Bhatnagar, and B. Nath, "ReInForM: Reliable Information Forwarding using multiple paths in sensor networks," In *Proc. of the 28th Annual IEEE Int. Conf. on Local Computer Networks*, Bonn, Germany, 2003; pp. 406–415.

[38] W. Lou, "An efficient N-to-1 multipath routing protocol in wireless sensor networks," In *Proc. of the 2nd IEEE Int. Conf. on Mobile Ad-hoc and Sensor System*, Washington, DC, 2005, pp. 672–680.

[39] W. Lou, Y. Kwon, "H-SPREAD: A hybrid multipath scheme for secure and reliable data collection in wireless sensor networks," *IEEE Trans. Veh Tech*., vol. 55, pp. 1320–1330, 2006.

[40] S. Li, R. K. Neelisetti, C. Liu, and A. Lim, "Efficient multi-path protocol for wireless sensor networks," *Int. J. Wirel. Mobile Netw*., vol. 2, no. 1, pp. 110–130, 2010.

[41] J. Ben-Othman, B. Yahya, "Energy efficient and QoS based routing protocol for wireless sensor networks," *J. Parall. Distrib. Comput*., vol. 70, pp. 849–857, 2010.

[42] H. Hassanein, and J. Luo, "Reliable energy aware routing in wireless sensor networks," *In Proc. of 2nd IEEE Workshop on Dependability and Security in Sensor Networks and Systems*, Los Alamitos, CA, 2006, pp. 54–64.

[43] R. C. Shah, and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Proc. of IEEE Wireless Commun. and Networking Conf.*, Orlando, FL, pp. 350-355, 2002.

[44] Z. Xiangbin, "An improved energy aware directed diffusion algorithm for wireless sensor network," In *Proc. of Asia-Pacific Conf. on Commun.*, 2007, pp. 385-388.

[45] Y. M. Lu, V. W. S. Wong, "An energy-efficient multipath routing protocol for wireless sensor networks," *Int. J. Commun. Syst*., vol. 20, pp. 747–766, 2007.

[46] Z. Wang, E. Bulut, and B. K. Szymanski, "Energy efficient collision aware multipath routing for wireless sensor networks," In *Proc. of the 2009 IEEE Int. Conf. on Commun*., Dresden, 2009, pp. 91-95.

[47] S. Rizvi, and K. Ferens, "Multipath route construction using cumulative residual energy for wireless sensor networks," In *Proc. of Seventh International Conference on Systems and Networks Communications*, Lisbon, pp. 71-76, 2012.