University of Alberta

A FAULT RECOVERY SCHEME FOR WIRELESS SENSOR NETWORKS

by

Ailin Zhou

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

©Ailin Zhou
Fall 2013
Edmonton, Alberta

Abstract

In wireless sensor networks, sensor nodes may fail due to energy depletion or physical damage. To recover the data of a failed node, we propose a fault recovery scheme which enables the remaining alive sensor nodes to use the redundant information with regard to the failed node to fulfill the recovery. The idea is similar to a level 4 RAID (Redundant Array of Independent Disks) in the sense that each sensor serves as the dedicated parity disk for its neighbors. We compare the network lifetime with and without recovery being involved, where the network lifetime is defined as the time interval from the point that a network starts operation to the point that a node failure is observed and can not be recovered. We explore the impact of parameters such as node failure probability, communication range, and node density on the network lifetime.

Acknowledgements

At the top of my thank you list is my supervisor Dr. Mario A. Nascimento. Throughout my study, he provided many learning opportunities for me. During our weekly meetings, he always has a way to help me organize my mind using his experience and expertise. It has been such a pleasure to work with this cool professor. It is my great honor to have Dr. Jörg Sander and Dr. Davood Rafiei as my committee members and Dr. Hong Zhang as the committee Chair. Prof. Antonio Loureiro gave me valuable feedback and suggestion during his visit to UofA last september. My sweet roommate Caitlin Cobb contributed so much of her time proofreading this thesis.

In the Chinese Zodiac, this year is my Year of Birth (Ben Ming Nian). Though I am not superstitious, it is indeed a life changing year for me. After going through all the dark days, never have I felt so powerful, fearless and determined in my life. I appreciate all the love and help my friends give me along the way. Without your support and encouragement, there is no chance I can make it. Special thanks to Joel Augustin and Darcy Matras, your kindness and trust are the reasons I choose to stay in Edmonton.

Last but not least, I want to thank my parents for bringing me to this world. Hope I will always be the daughter you can be proud of.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

A wireless sensor network (WSN) is made up of potentially large numbers of sensor nodes that use wireless communication protocol to communicate. The sensor nodes can gather, process and exchange sensed data in a large area to achieve various goals. Generally, the idea is to use autonomous sensor nodes to monitor areas of interest so that spatial and temporal data about the physical phenomenon can be collected without the need to actually be present in the target area. Each sensor node contains a partial view of the environment it covers. The collected data can be forwarded to certain energy-unconstrained base station(s) or sink(s) on a regular basis or on-demand. Hence, the advent of WSNs redefines the data gathering process and significantly reduces the need of human involvement.

We have seen the proliferation of WSNs in a wide range of applications in recent years. WSNs have been adopted to facilitate scientific research as well as military and civil applications. For instance, environmental scientists may build a sensor network to monitor snow processes in high mountains or climate change in forests. The military force would have a WSN that can detect intruders and report any abnormal observations. More applications in seismic/acoustic/medical data-gathering [5], wildlife/habitat/civil structure monitoring are used.

Hardware technology advances enable the massive production of small and cheap sensors. Sensor nodes come along with some limitations such as strict energy constraints and limited computational capacity due to their small sizes and considerations of cost-effectiveness. Subject to application specific requirements, sensors may vary in size, computational ability, power capacity, properties to be measured, etc. However, sensors do share similar architecture and composition modules. A sensor node is mainly made up of four components: sensing module, processor, radio module and power unit, as shown in Figure 1.1. The sensing module samples the analog sensor signal and uses an A/D converter to digitize the signal to achieve the goal of mapping physical phenomenon such as temperature, pressure and light to quantitative measurement. The processor processes the digitized data and completes the necessary manipulation. This data is often referred to as “raw data” and needs to be relayed back to the sink. The radio module needs to establish digital channels in which sensors are able to communicate with each other. To achieve such a goal, this component usually adopts a radio transceiver for both receiving and transmitting data. The power unit is often battery-operated and is used for supplying power to all other units.

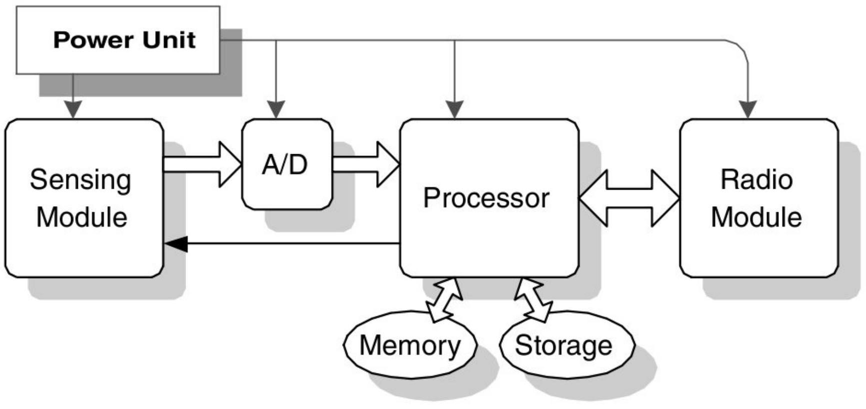


Figure 1.1: Typical architecture of sensor nodes [18]

Equipped with the radio module, the sensors can be integrated into networks of any size. With proper routing algorithms a sensor can disseminate its measured data to any other sensor or to the sink for further data analysis. Moreover, the ability to communicate with other nodes in the network can facilitate the cooperation among nodes to achieve a much more effective monitoring goal. Modern WSN applications are often in the size of hundreds or thousands of sensor nodes.

Based on their different functions, the WSNs can be generally put into four categories: monitoring, detection, prevention and response. Depending on the characteristics of the deployed nodes, the sensor networks can also be classified as homogeneous or heterogeneous networks. Homogeneous sensor networks consist of identical nodes with the same capability and functionality while the heterogeneous networks assign different roles to nodes with different capabilities. Heterogeneous networks facilitate the advent of hierarchical or cluster-based topology and algorithms in which sensor nodes form clusters and send their data only to the corresponding cluster head. In terms of mobility of the nodes, WSNs come in two main forms: static or mobile. For some applications static sensor nodes would suffice, while for other applications, mobile nodes are more favorable. According to [18], if sensor nodes can adjust their positions, the monitoring capability can be improved and the required transmission power can be reduced. Depending on how the network is organized and how the nodes relay sensed data, a network can be either multi-hop or single hop. Multi-hop communication is the mainstream communication mode due to the requirement of low energy cost and scalability.

Wireless sensor networks have attracted the attention of many researchers in recent years. There are some common research challenges. Failures in wireless sensor networks can occur for various reasons. Unlike wired networks, WSNs suffer from resource constraints due to the wireless communication channel and the basic nature of the nodes. The scarcity of the wireless channel limits the bandwidth of each sensor and the inherent error-prone nature of the channel leads to some failure in the communication process. The fact that sensor nodes have strict energy constraints and that they are often deployed in large-scale, remote and harsh environment renders the solution of recharging batteries impossible in most circumstances. Once the energy depletes, the node inevitably fails. Moreover, the hostile and unpredictable environment makes the sensors vulnerable. The sensors can be destroyed by an external event at any time. Since the deployment fields are sometimes unattended, an immediate replacement of a failed sensor node is not always feasible.

Therefore, sensor nodes may fail due to energy depletion or physical damage after the deployment of nodes. When a node failure takes place, the sensed data stored in this node will be lost ever since. The user can no longer have access to all the valuable information the sensor node has collected before its failure, which may mislead the user's understanding of this network. To solve the problem of node failure and data loss, a fault tolerant and recovery scheme is in urgent need for securing the sensed data within the network.

1.2 Overview of our scheme

The goal of our work is to maintain high data availability and guarantee the completeness of query results when sensor node failure and data loss take place. To achieve such goal, we propose to incorporate a fault tolerant, more specifically, fault recovery scheme into WSNs. RAID (redundant array of independent disks) is the original storage technique, which distributes data and utilizes redundant disks for data recovery. Similar to the idea of RAID, we propose to let each sensor node store some redundant information about all its direct neighbors. Within each sensor node the storage unit is partitioned into two distinct sections. One section stores normal sensed data and the other section stores the redundant information about its neighbors' sensed data. For example, node A, whenever it updates its own sensed data, a copy of the update is sent to A's direct neighbors so that the neighbors can recalculate and update their information with regard to node A. In this way, the sensed data of A is preserved in more than one place and therefore a recovery of A's sensed data is possible by taking advantage of its neighbors which contain the redundant information about A. Our scheme is quite different from previous works in WSN research. We do not aim to minimize the energy consumption of data dissemination or to provide a mechanism for tolerating link failure. Instead, we are focusing on the data management level to achieve fault tolerance. It is obvious that our scheme imposes the communication overhead of constantly broadcasting data updates to neighbors and the additional message transmission cost when a recovery process is initiated. However, we argue that it is worthwhile to invest a little bit in exchange for a much longer period of complete query results, just like we pay our health insurance fees for the long-term benefits. Figure 1.2 shows the topology of a network which is composed of 10 sensors. The edge between each pair of nodes indicates that the connected nodes are within the communication range and thus can transmit and receive messages to/from each other. In this topology, should node 1 fail, our scheme is able to recover node 1's sensed data by taking advantage of nodes that are within any of the three rounded rectangles.

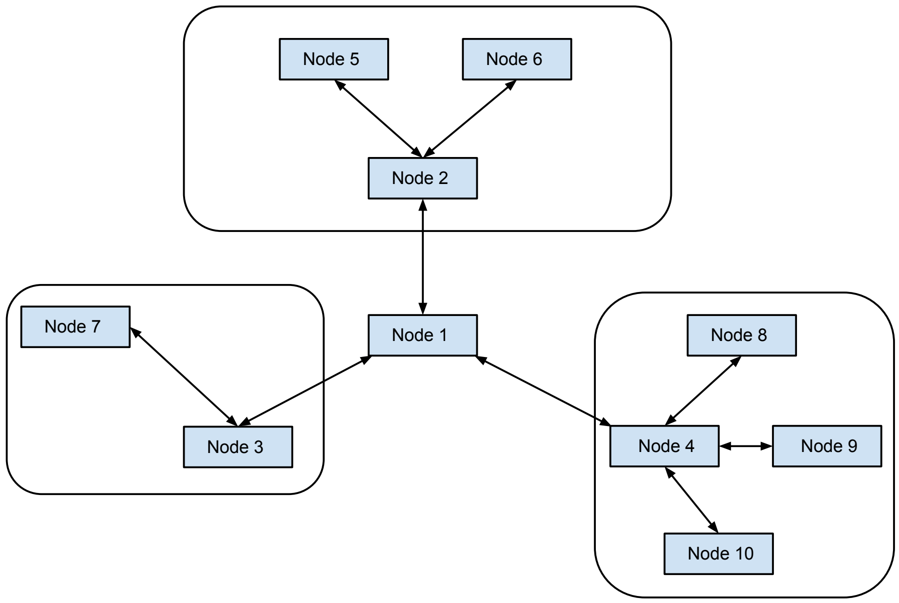


Figure 1.2: Schematic diagram of the proposed recovery scheme

1.3 Thesis contributions

In this thesis project we focus on proposing a fault recovery scheme in the context of wireless sensor networks. By incorporating data redundancy into the network our scheme can recover the sensed data of an already failed node and therefore return the correct query results as if there were no failure in the network. More specifically, the contributions of this thesis work lie in the following aspects:

- adapting the idea of RAID to wireless sensor networks
- enabling WSNs to tolerate a moderate rate of node failure without showing any query result discrepancy

- presenting the communication energy consumption model of the proposed scheme
- conducting extensive experiments which show the performance gain of network lifetime under different parameters

To the best of our knowledge, this is the first attempt to achieve in-network node failure recovery for wireless sensor networks.

1.4 Thesis outline

Chapter 2 reviews recent literatures on fault tolerance in WSN research and the general idea of RAID. Chapter 3 introduces the methodology of our scheme including the underlying assumptions, the energy consumption model, and the network lifetime definition. A detailed description of the simulation environment and experimental setup is presented in Chapter 4. Chapter 5 presents the experimental results and a detailed analysis of them. A brief conclusion and possible future work directions are given in the final chapter.

Chapter 2

Related Work

In this chapter, we start reviewing existing research efforts in the field of fault tolerance in wireless sensor networks. Then a brief introduction to RAID (redundant array of independent disks) follows.

2.1 Fault tolerance

Fault tolerance has been intensively studied in computing science for more than half a century. In general, fault tolerance is the ability to maintain a desired level of functionality in the presence of failures. In the context of wireless sensor networks, both sensor nodes and data transmissions are prone to failures. The sensor nodes can fail due to energy depletion, hardware faults, physical damage, and so on. The data transmissions depend heavily on the quality of the communication channels, which can be affected by environment conditions, physical obstacles, and signal interference. Such error-prone nature requires that fault tolerance should be considered at the system design level for many WSN applications. Thus, lots of research works on incorporating fault tolerance into WSNs has been proposed. Researchers address the fault tolerance issue at different layers of the network protocol stack, e.g., communication layer or application layer.

Many researchers focus on designing fault tolerant routing protocols. According to a survey by Alwan and Agarwal [2], fault tolerant routing techniques can be classified into two main categories: retransmission and replication. Retransmission means that if the sender can not receive an acknowledgment packet from the sink before a pre-defined timeout, the sender will retransmit the sensed data to the sink. This approach is quite popular since the packet loss rate in WSNs is higher than in traditional networks. Two popular replication mechanisms are multipath routing [20, 14, 13] and erasure coding [40, 41]. In the former approach multiple copies of sensed data are transmitted over multiple routing paths so that the data can successfully reach its destinations, so long as one path is free from node failures along the way. Thus, the multipath routing protocols are more resilient to node failures at the expense of increased overall traffic. In [13], the authors study the trade-off between the increased traffic and the fault tolerance ability of the network. The scheme proposed in [14] is a multipath variant of the original directed diffusion paradigm [19] and the authors believe that the use of multipath routing provides a viable alternative for energy-efficient recovery from failures in WSNs. Erasure coding is another replication approach aiming at enhancing fault tolerance in WSNs. The basic idea is to add K parity fragments to the M data fragments to have a total of $M + K$ fragments. Then the $M + K$ fragments are divided into sub-packets and transmitted over multiple paths. The sink can reconstruct the original data when at least M out of the $M + K$ fragments have been successfully transmitted to the sink. The most widely used erasure coding algorithms in WSNs are Reed-Solomon codes [30] and Rateless codes [7]. Deng et al. propose a light-weight routing mechanism for tolerating node failures [10]. This routing mechanism can dynamically repair routing paths between sensor nodes to sink by selecting a new path which circumvents the failed node. When a sensor node detects that its parent node has either failed or lost access to the sink, this node notifies all its child nodes about this failure, asks its neighboring nodes for their connection information, and chooses one of the neighboring nodes as its new parent node based on the information received.

Liu et al. [23] survey fault tolerance algorithms/protocols in application layer of WSNs. They study how fault tolerance is addressed in the following five categories of applications: node placement, topology control, target and event detection, data gathering and aggregation, and sensor surveillance. In each category, representative research works that aim at achieving fault tolerance in application layer are discussed.

Some research works about fault tolerance management can be found in [3, 32, 42, 27]. Yu et al. [42] survey existing fault management approaches for WSNs and classify them into three phases: fault detection, fault diagnosis, and fault recovery. They consider fault detection as the phase where the network system properly identifies the unexpected failure. According to their classification, there are two main types in the current fault detection techniques: centralized and distributed fault detection. Fault diagnosis is considered as the phase where the network distinguishes certain detected faults from other irrelevant or spurious alarms. The last stage, fault recovery, is the phase where the network is reconstructed or reconfigured in order to mitigate the impact of detected failures. Another survey on fault management is presented by Paradis and Han [27]. In this survey, they follow the taxonomy of different fault tolerant techniques used in traditional distributed systems [37], i.e., fault prevention, fault detection, fault isolation, fault identification, and fault recovery. Then they summarize and compare existing fault tolerant algorithms/protocols for each category. Saleh et al. in [32] come up with a general framework for fault tolerance in WSN that can be used to guide the design and development of fault tolerance solutions. The proposed framework has two major modules: System Management module and System Operations module. System Management consists of four submodules: Defining Roles and Structures, Generating FT Schedule, Assignment of Roles, and Executing FT Schedule while System Operations consists of three submodules: Fault Discovery, Cost Assessment, and Fault Containment and Recovery. The two major modules interact via a Analysis and Refinement component. The authors then apply the framework to CRAFT, a checkpoint/recovery scheme for data collection and dissemination in WSNs [31]. They also use the framework to evaluate a few existing fault tolerant schemes and conduct a comparative study based on the fault tolerance requirements they define. The authors in [3] propose a self-managing fault management mechanism for WSNs to deal with fault detection and recovery. They start with the definition of a fault model which describes different types of faults within the network. Then a hierarchical structure is proposed to distribute fault management tasks and assign different roles to the sensor nodes so that the sensor nodes can take appropriate reaction to fulfill their fault management goals. Their simulation results prove that the proposed mechanism is more efficient than the scheme proposed in [38], in terms of energy consumption and response time.

Most of the existing fault management or fault tolerance techniques in WSNs simply isolate the failed or malfunctioning nodes in the communication layer and ignore the data of the failed node as in [10, 25, 26, 36]. In these papers, fault tolerance is achieved in a sense that the networks can still fulfill the sensing tasks in the presence of failures. However, these approaches do not deal with the recovery of the failed node. Once a node has failed, the data stored in the failed node is lost forever with these approaches. We argue that the data of the failed node is valuable. Thus, in this thesis project we propose a fault recovery scheme to recover the data after node failures take place in WSNs.

Similar to our goal, Chessa and Maestrini [8] present a fault recovery mechanism to cope with node failures in single hop WSNs. They came up with a solution to recover data after a node failure by distributing redundant information of sensed data among sensor nodes. More specifically, they proposed to partition the memory of sensor nodes into two parts, one for storing its own sensed data and the other for storing redundant data used for recovery. By keeping redundant data of other sensor nodes this scheme is able to recover data loss after a node failure. The redundant concept is similar to our work. However, their mechanism can only deal with single node failure within single hop WSNs, whereas our work can be applied to multi-hop WSNs for more than one node failure.

2.2 RAID

First proposed by Patterson et al. [28], the term RAID was originally the acronym for Redundant Array of Inexpensive Disks. Later on manufacturers redefined the term to represent Redundant Array of Independent disks. In the original paper, the authors came up with the idea of RAID to solve the bottleneck issue of I/O performance in order to catch up with the increasing speed of CPUs and memories. The authors made it clear that RAID can be achieved in both hardware and software implementation. The paper described five different organizations of RAIDs and analyzes their relative cost as well as performance, which laid the foundations of modern architecture from RAID 1 to RAID 5.

Nowadays, RAID is typically referred to as a storage virtualization technology that can replicate and distribute data among an array of disk drives while being accessed by the operating systems as one single logical drive. Depending on different reliability and performance requirements, RAID can be categorized into several "levels" which specify how the data is accessed and how the redundancy is maintained. RAID levels and their corresponding data formats are standardized by the Storage Networking Industry Association (SNIA) in the Common RAID Disk Drive Format (DDF) standard[1]. A brief description of each standard RAID level follows.

RAID 0 represents a striped array with no parity. It divides sequential data into several blocks so that different blocks can be written to different physical disk drives. This level does not aim to provide any redundancy since it utilizes block-level striping without parity or mirroring. The merit of RAID 0 is that several physical disks can be combined to make a larger virtual disk and the I/O performance will be better than using a single disk. One drawback is that the more disks added to the array, the more likely that disk failure and data loss can happen.

RAID 1 can be described as a mirrored array. It requires at least two disks and the data is always written identically to all the disks. Read operation can be completed by accessing any of the disks in the array since the disks contain the same data. By simply mirroring the data with no parity or striping techniques, the fault tolerance is achieved in a sense that as long as one disk in the array is properly functioning, the data can be accessed.

RAID 2 adopts bit-level striping so that no two consecutive bits in the original data stream are written to the same disk. Hamming code is used in this level for error correction. This level is the only original level proposed in [28] that is not currently used.

[1]http://www.snia.org/tech_activities/standards/curr_standards/ddf

RAID 3 is a byte-level striped array with a non-rotating parity disk. The non-rotating parity disk is a special disk that is dedicated for storing parity results of other data disks within the array. This level uses byte-level striping so that no two consecutive bytes are on the same disk. Besides, the hamming-code parity used in RAID 2 is replaced by bitwise "exclusive or" parity of the corresponding byte.

RAID 4 is a block-level striped array with a non-rotating parity disk for storing the parity of other data disks. Since the data is stored in block-level and each disk operates independently, the I/O requests on different disks can be handled concurrently. This level can tolerate at most one physical disk failure. In our work, we choose this RAID level to incorporate data redundancy and achieve fault recovery since it strikes a balance between space efficiency and data redundancy. Fig. 2.1 shows the diagram of a RAID 4 array composed of three data disks and one parity disk. RAID 4 is often referred to as a dedicated parity scheme since it utilizes a

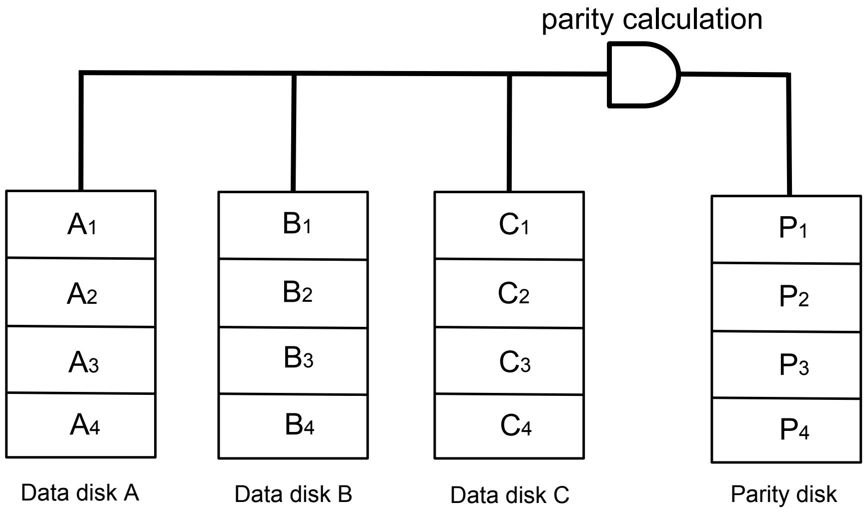


Figure 2.1: RAID 4 with three data disks and one parity disk

single disk for storing the parity information only. In this figure, each row within the disk represents a data block. The data block of the parity disk stores the parity results for all the data blocks on the data disks that are in the same row. Denote the parity calculation as $\oplus$, for data block i, within this array we have the following relations:

$$P_i = A_i \oplus B_i \oplus C_i \tag{2.1}$$

Whenever a write is performed on any of the data disks, the parity calculation component recalculates and updates the corresponding blocks on the parity disk. Should any of the data disk fails, the remaining data disks, together with the parity disk, can be utilized to reconstruct the data of the failed disk. For instance, if data disk A fails, we can recover its data according to Eq. 2.2:

$$A_i = P_i \oplus B_i \oplus C_i \tag{2.2}$$

RAID 5 is a block-level striped array with rotating parity disks. It is similar to RAID 4 except that now there is no more dedicated parity disk in the array. Instead, the parity information is distributed evenly across the array, that is, each disk rotates and takes partial responsibility for the dedicated parity disk in RAID 4. This level can only tolerate one disk failure.

RAID 6 is designed in a similar way as RAID-5, but with dual rotating parity physical disks. Thus, this level can tolerate at most two physical disk failures so as to achieve high availability. Two distinct parity are calculated and distributed evenly across the array. This RAID level eliminates the potential risk of data loss when one disk in RAID 4 or RAID 5 has already crashed and not yet been replaced and recovered.

Chapter 3

Methodology

The motivation behind our research is to achieve high data availability and fault tolerance in the context of node failure. Our scheme is more suitable for application scenarios with the following characteristics:

- each sensor node plays the same role in the network
- the data each sensor gathers is of equal importance to the operation of the network
- the data correlation or redundancy between neighboring nodes is rather small

3.1 Assumptions

A few assumptions about the system have been made to simplify the problem.

To make our method more general, the first assumption we made is that the topology of the network is flat, which means that there are no superior nodes, hierarchical structures or cluster heads that are in charge of other nodes within their domain. All the nodes are considered as having the same significance and are equipped with the same amount of storage space, computation resources, communication capacity and initial energy supply. Even though networks with clusters, such as LEACH [16], do have a better performance in terms of minimizing energy consumption, we try to make our scheme as generic as possible and investigate the performance gain as well as the energy consumption overhead under a generic network topology setting.

In our scheme, the sink serves as an intelligent agent which spreads the queries in each round, stores the whole network topology and chooses the best recovery candidate based on the topology. The sink is assumed to be located at the center of the deployment field. The sink is also considered as being constantly charged by reliable electricity sources and thus has unlimited power supply.

We also assume that all the nodes communicate with a fixed communication range since typical cheap sensors are not equipped with sophisticated communication electronics that can adapt the sensor's transmission range according to different scenarios. In other words, the sensors we consider are not able to adjust the transmitting power towards different destination nodes.

We assume that sensor nodes are stationary after being deployed. Even though mobile WSNs have their merits, such as the sensor coverage and network connectivity can be improved, the mobility of nodes always complicates the design and analysis of WSNs. For instance, if the nodes are mobile, the network topology may change radically even if only a small portion of nodes change their positions, causing problems for routing. The majority application scenarios of wireless sensor networks are static networks so far [11] and mobile nodes are only used in special circumstances for achieving specific goals [1]. Since our method relies heavily on the neighborhood relationships within the network, it is impossible to tolerate topology changes caused by node mobility.

We assume that the initial radio state of all the sensor nodes are on and we do not put the radio to sleep at any time. One may argue that not including wake/sleep scheduling is not realistic for real-life applications. Though sleep/wake scheduling has been considered as an effective mechanism to reduce energy consumption and prolong network lifetime, it may bring up additional communication overhead for periodically exchanging synchronization information between neighboring nodes to achieve the precise synchronization. It may also incur additional transmission delays while waiting for neighboring nodes to wake up. Neither of these two characteristics is desirable for our scheme. After all, the purpose of this thesis project is to tackle the problem of possible node failure, thus the sleep/wake scheduling is not our concern.

We assume that the edges between connected nodes are bi-directional. By bi-directional, we aim to convey that the nodes can not be considered as connected unless they can both transmit/receive messages to/from each other. If bi-directional edges do exist between a pair of nodes, we assume that there is a link between them. We also assume that in our scheme there is no link failure taking place during which messages are dropped. The primary reason for making this assumption is that our recovery scheme requires successful transmissions of sensed data from a node to its neighbors.

Since our approach depends heavily on location of sensors and neighborhood relationships between sensors, we assume that both the location and the neighborhood information is known to all the sensors. This assumption implies that each sensor node has a list of all the neighbors that reside within its communication range. The sink stores the whole network topology. In real-life deployment, this can be achieved during the system initialization period.

3.2 System model

We consider a WSN composed of large numbers of sensor nodes. The sensor nodes do continuous and periodic sensing and data collection at their locations. Then the sensor nodes pack the data into messages and transmit these messages back to the sink on demand, i.e., when the sensors receive queries originating from the sink. In the experiment, we assume that there is one single sink located at the center of the deployment field, however, this can be easily generalized to other cases. Since our scheme is proposed for maintaining data availability in the context of node failure, the nodes are considered as error-prone. A node can be in either an alive or a failed state. The transition from an alive state to a failed state is one-way and irreversible. When one node fails, the remaining alive sensor nodes are able to cooperate and recover the sensed data of the failed node by utilizing the redundant information stored in the alive nodes. Therefore, the remaining alive sensor nodes can successfully handle queries with regard to the failed node, as if there were not any node failure. Moreover, if a new sensor is added into the network to replace the failed node, reconstructing the data of the failed node and migrating the data to the replacing node can be potentially achieved in a similar way as described in the proposed recovery scheme.

In our proposed scheme a lot of operations are conducted by the sink in a centralized manner, including issuing queries, neighborhood relationship management, best recovery candidate selection, recovery initialization, and possibly the recovery itself. The advantage of this design is the ease of implementation and less communication needed from node to node. None of the sensor nodes rely on the global knowledge of the network topology. They only need to be aware of who their neighbors are. This centralized design decreases the message exchanges between sensor nodes and can therefore reduce the energy consumption.

During our simulation, at the beginning of each round, the sink generates a query message specifying the target query area. The query message is in the format of [$Center$, $Radius$] where $Center$ represents the coordinates of query center. The query message is then flooded to the whole deployment field. Upon receiving the query message, each sensor calculates the Euclidean distance d between its own location and $Center$. If d is no greater than the specified $Radius$, this node determines that it is within the query area and will respond to this query. Fig. 3.1 shows several nodes which are depicted as small disks and the query area as specified in a query message. In this case, node i will respond to this query, since $d_i \leq Radius$ and node j will not respond as $d_j > Radius$.

We aim to incorporate in-network data redundancy to achieve node failure recovery. The main idea is about properly preserving the sensed data of one node in its neighboring nodes so that in case this node fails, the neighboring nodes still have access to sufficient information for recovering the data of the failed node. To achieve this goal, we propose to partition the storage unit of a sensor node into two separate sections, one for storing its own periodically collected data and the other for storing the redundant information with regard to the sensed data of all its neighbors. Fig. 3.2 shows an example of storage unit partition in a simple sensor network which consists of six sensor nodes. The sink is ignored in this figure. Each rectangle represents a sensor node. The edge between each pair of nodes indicates

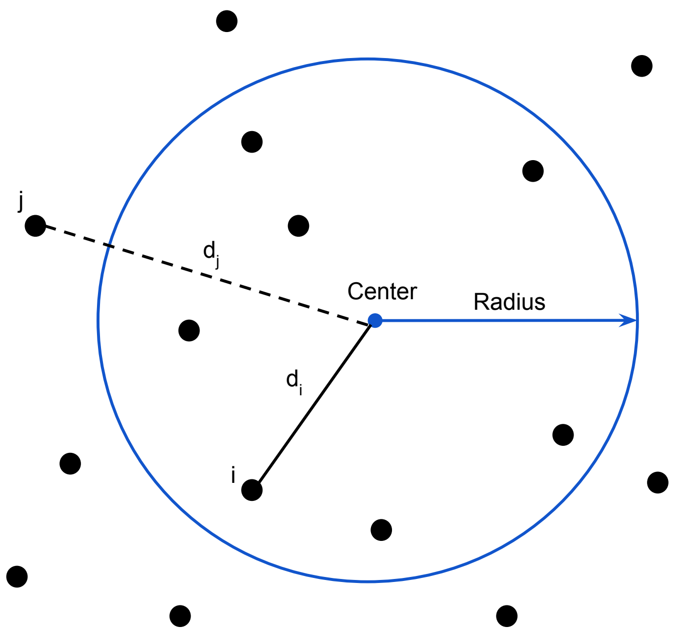


Figure 3.1: Nodes and the query area

that the connected nodes are within communication range and can transmit/receive messages to/from each other.

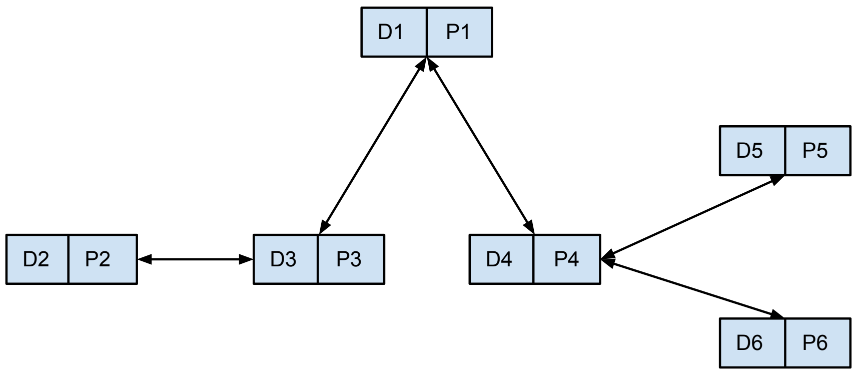


Figure 3.2: Storage unit partition in sensor nodes

As shown in Fig. 3.2, the storage unit of sensor nodes are partitioned into D_i and P_i where i represents the node ID. D_i is the same as the storage unit of common sensors and the data it stores is named $SensedData$. P_i is responsible for storing the redundant information of all this node's direct neighbors and the data it stores is called $ParityData$. Each time a node samples new data, the node not only updates its D_i section but also sends a copy of the newly sensed data to all its direct neighbors so that the neighboring nodes can update their P_i sections by calculating the parity of all the received data. Analogous to the idea of RAID 4 [28], each sensor node in our scheme now serves as the dedicated parity disk for the array composed of all its direct neighbors.

Different from the definition of parity in mathematics, which refers to the evenness or oddness of an integer, the parity in our context is the result of bitwise exclusive or operation (denoted as $\oplus$) of all the binary inputs. Therefore, the parity result depends on all the inputs. Exclusive or, or abbreviated as XOR, is a widely used logical operation in computing science. If there are two boolean inputs, the output of XOR is true *iff* one input is true. If there are more than two boolean inputs, the output of XOR is true *iff* an odd number of inputs is true. Parity has the property that for the same bit, if one single error or an odd number of errors take place in the input, the parity result of this bit will be incorrect. This property makes parity a popular error detection scheme. In our scheme the P_i section of a node stores the parity result of this node's direct neighbors. If we simply use D_i and P_i to represent the data stored in these sections, based on the topology shown in Fig. 3.2, we can define the following XOR relations for nodes with more than one neighbor:

$$P_1 = D_3 \oplus D_4 \tag{3.1}$$

$$P_3 = D_1 \oplus D_2 \tag{3.2}$$

$$P_4 = D_1 \oplus D_5 \oplus D_6 \tag{3.3}$$

Suppose D_1 contains data 10010110 and D_2 contains data 00101111. By conducting the bitwise XOR operation as defined in Eq. 3.2 we can calculate the value of P_3 in this way:

$$\begin{array}{rl} & 10010110\ \ D_1 \\ \underline{\text{XOR}\ \ } & \underline{00101111\ \ \text{D}_2} \\ & 10111001\ \ P_3 \end{array}$$

Using the property of XOR, should any input value get lost, the lost input can be easily rebuilt by conducting XOR operation on all the remaining input values and the former XOR output value. Assume that in this case node 1 fails and we need to reconstruct D_1. From Eq. 3.2 and Eq. 3.3 we can see that D_1 is preserved twice in P_3 and P_4 respectively. We can reconstruct D_1 by using either of the two means as specified in Eq. 3.4 and Eq. 3.5:

$$D_1 = P_3 \oplus D_2 \tag{3.4}$$

$$D_1 = P_4 \oplus D_5 \oplus D_6 \tag{3.5}$$

Assume we adopt Eq. 3.4 to fulfill the recovery, by subjecting P_3 and D_2 to the XOR operation:

$$\begin{array}{r} 10111001\ \ P_3 \\ \underline{\mathrm{XOR}\ \ 00101111\ \ \mathrm{D}_2} \\ 10010110 \end{array}$$

the data we can recover is 10010110, which is exactly the data stored in D_1.

As shown in Eq. 3.4 and Eq. 3.5, in order to recover the $SensedData$ of a specific node, sometimes there is more than one option. The average number of options is proportional to the network density and communication range. However, the total number of failed nodes increases as time goes by so that not all the D_i at the right hand of the equations are accessible. Therefore, not all the options are suitable for conducting a successful recovery. The good news is that our recovery scheme works when there is at least one valid option for a specific failed node. This means that as long as one neighbor of the failed node, denoted as A, is alive and all the direct neighbors of A survive, the recovery can be successfully completed.

3.3 Recovery initialization

Prior to proceeding with the recovery, the sink needs to figure out which nodes are expected to respond in each round. All the nodes that are expected to respond to the current query forms a responding node set. When the sink does not receive the query response from a node that belongs to the responding node set, this node is considered as failed. Note that we assume that the sink has the great picture of the network topology. With range query structures like R-tree [15] and its variants [4, 33], it is guaranteed that we can find which nodes reside within the query range in $O(logN)$ time theoretically. In our simulation, we did not implement such data structure because it is not necessary for the size of our simulation. Instead, we adopt a passive failure detection mechanism. Our experimental setup ensures that the alive nodes within the responding node set will generate the response and the response message is scheduled to arrive at the sink in the next round. In other words, the first time that the sink misses the response of a specific node, the sink determines that this node has failed. Thus, in the following sections, we treat the time when we observe a missing response as the time when the corresponding node failure happens. Once a node is considered as failed, it will be removed from the responding node set since the sink should not expect its responses any more.

When a node failure has been observed, denoted as node F, the sink will initiate the recovery process immediately according to Alg. 1. The main goal of this process is to find the best recovery candidate.

From line 1 to 5, all the alive neighbors of the failed node F are considered as recovery candidates. They are included in a recovery candidate set R.

From line 6 to 18, the algorithm traverses R to find all the valid candidates to form the valid candidate set R'. We define a recovery candidate j as valid if all the direct neighbors of j, other than the one which we aim to recover, are in the alive state by the time we initiate the recovery process.

The last part of the algorithm traverses R' and chooses the candidate with the minimum vertex degree, denoted as R_{parity}, as the best recovery candidate to fulfill the recovery task. The problem with a high degree recovery candidate is obvious in the sense that we need more information about this candidate's neighbors. The recovery candidate with the minimum vertex degree in R' guarantees that there is less communication overhead incurred for requesting the data of this candidate's neighbors. The best recovery candidate is referred to as R_{parity} node since the recovery process needs the $ParityData$ of it to fulfill the recovery.

Algorithm 1 Choose the best recovery candidate R_{parity}

```
Require: node $F$.alive==FALSE
 1: for every direct neighbor $i$ of $F$ do
 2:     if $i$.alive == TRUE then
 3:         $R = R \cup i$
 4:     end if
 5: end for
 6: for every $j$ in $R$ do
 7:     Valid=FALSE
 8:     for every direct neighbor $k$ of $j$ do
 9:         if $k$.alive == FALSE and $k \neq F$ then
10:             Valid=FALSE
11:         else
12:             Valid=TRUE
13:         end if
14:     end for
15:     if Valid==TRUE then
16:         $R' = R' \cup j$
17:     end if
18: end for
19: Min = INFINITY
20: for every $l$ in $R'$ do
21:     if $l$.degree $<$ Min then
22:         Min = $l$.degree
23:         $R_{parity} = l$
24:     end if
25: end for
```

The direct neighbors of the R_{parity} node, excluding F, are considered as R_{data} nodes. Their $SensedData$, together with the $ParityData$ of R_{parity} node, will be utilized to reconstruct the data of F. In Fig. 3.3, a partial snapshot of a real simulation run in our experiment is presented. The dots represent the uniformly distributed sensor nodes and the edges between nodes indicates that the connected nodes can receive/transmit messages from/to each other. Now assume that the node within the circle is the failed node F. According to the topology, six sensor nodes, from node 1 to node 6, form the recovery candidate set R, as marked by triangles. Assume that all the recovery candidates in R are valid at this point. Node 3 has the minimum degree of 4 among all nodes in R. Therefore, node 3 is chosen as the best recovery candidate R_{parity}. Its three direct neighbors, i.e., node 4, 7, and 8, are considered as R_{data} nodes. The data of F can be recovered as specified in Eq. 3.6 after receiving all the necessary responses.

$$D_F = P_3 \oplus D_4 \oplus D_7 \oplus D_8 \tag{3.6}$$

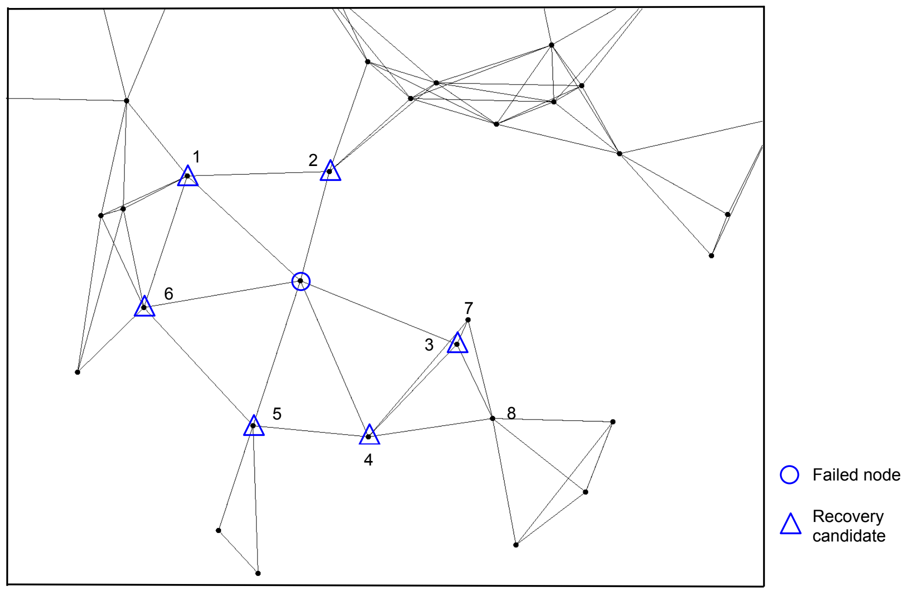


Figure 3.3: Recovery candidates in a simulation run

3.4 Centralized and localized recovery approaches

After the sink initiates the recovery and chooses R_{parity}, the actual recovery process begins. The recovery process can take place either at the sink or in the R_{parity} node. If the process takes place at the sink, we name it centralized recovery. If it takes place in the R_{parity} node, we name it localized recovery. The only difference between centralized and localized recovery is where the actual recovery takes place and where the data packets are transmitted. For centralized recovery, all the R_{parity} nodes and R_{data} nodes send their responses back to the sink. In contrast, for localized recovery, the sink only initiates the recovery and all the other operations are done at the best recovery candidates locally. The actual recovery process is slightly different for both recovery approaches:

- For centralized recovery, if a R_{data} node is within the query area, the sink has already got its query responses and stored its $SensedData$. Therefore, it is not necessary for the sink to request for its data. Otherwise the sink needs to send request messages to R_{data} nodes that are not within the query area, asking for their $SensedData$. After receiving all necessary responses, the sink completes the recovery calculation. Alg. 2 gives the pseudocode for this process.

- For localized recovery, the R_{parity} node will be notified by the sink that it has been chosen as the best recovery candidate and a recovery is needed based on it. Then the R_{parity} node sends request messages to all R_{data} nodes and waits for their responses. The recovery calculation takes place in the R_{parity} node and now the R_{parity} node can respond to queries on behalf of the failed node. The process is shown in Alg. 3.

In general, the centralized recovery approach is more suitable for queries that are interested in raw sensed data while the localized recovery approach is a better idea for aggregate queries. Aggregate queries are more concerned about the summary or representative values of the raw data instead of details of the raw data. When processing such queries, sensors merge and condense their data into high

Algorithm 2 Centralized recovery at the sink

```
Require: $R_{parity}$ has been chosen
 1: send request message to $R_{parity}$ node for its $ParityData$
 2: for every $R_{data}$ node $i$ do
 3:     if $i$ outside of query area then
 4:         send request message to $i$ for $SensedData$
 5:     end if
 6: end for
 7: if all response messages have arrived then
 8:     Result = $ParityData$
 9:     for every $R_{data}$ node $i$ do
10:         Result = Result $\oplus$ $SensedData[i]$
11:     end for
12: end if
```

Algorithm 3 Localized recovery in the R_{parity} node

```
Require: this node has received the notice for initiating the localized recovery
 1: send request message to every $R_{data}$ node for $SensedData$
 2: if all response messages have arrived then
 3:     Result = $ParityData$
 4:     for every $R_{data}$ node $i$ do
 5:         Result = Result $\oplus$ $SensedData[i]$
 6:     end for
 7: end if
```

quality data packets in order to reduce the volume of data that needs to be transmitted to the sink. Therefore, aggregate query processing can result in a lower energy consumption and a prolonged network lifetime. If a centralized recovery approach is adopted, whenever a recovery process is initiated, R_{parity} node and R_{data} nodes need to transmit their raw data back to the sink. This is strongly against the concept of aggregate query which aims at reducing the transmission of raw data. The localized recovery approach solves the problem by having all the R_{data} nodes send their data to R_{parity} node. Then the whole recovery process is completed in the R_{parity} node so that no raw data transmission to the sink is required. After the recovery, R_{parity} has both its own sensed data and the sensed data of the failed node. Then the aggregate query can be processed as it is.

Fig. 3.4 shows an example of the aggregation process. Note that this is not the whole network topology, instead it is an aggregation tree rooted at node 1 where data aggregation takes place at the non-leaf nodes. The aggregate function we choose is to find both the minimum value min and the maximum value max of the sensed data. The corresponding aggregation results are shown in the format of [min, max] right beside each node. In Fig. 3.4(a) all the data is available and node 1 has no problem getting the correct aggregation results. Then in Fig. 3.4(b) node 6 has failed and without the recovery process the data of node 6 is lost. Thus the root gets the max value of 8, which is not the maximum value among what all the sensors have gathered. Assume that node 5 is chosen as the R_{parity} node and has completed the recovery process for node 6, as shown in Fig. 3.4(c), node 2 can aggregate the data of node 4, 5 and the failed node 6, leading to thes correct aggregation results at node 1.

3.5 Modeling energy consumption

Energy constraint is one of the most fundamental challenges in WSN research. Researchers have come up with many energy models [12, 45, 17, 34] that are used to evaluate network lifetime or compare different algorithms or protocols in network design and analysis. With the help of energy models, we can have an analytical and

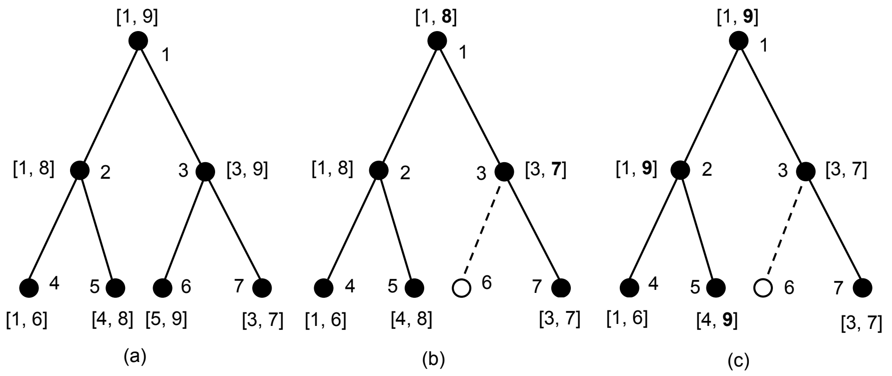


Figure 3.4: Aggregation process in a routing tree where: (a) every node is alive, (b) node 6 failed and no recovery has been achieved, (c) node 6 failed and the recovery process has taken place at node 5

quantitative analysis of how much energy the network consumes as time goes by. In general energy cost has three categories: sensing, processing and communication. The energy cost for sensing and processing are often considered as constant values. For communication, the energy consumption for both receiving and transmitting packets are considered. The cost for receiving is due to the receiver electronics while the cost for transmitting is due to both transmitter electronics and radio frequency (RF) transmit power. In [29] the author gives a simple yet reasonable model for calculating the transmitting and receiving energy cost for a single bit:

$$E_{tx} = \alpha_{11} + \alpha_2 \times d^n \tag{3.7}$$

$$E_{rx} = \alpha_{12} \tag{3.8}$$

where E_{tx} and E_{rx} represent the energy it takes to transmit and receive one bit respectively, α_{11} and α_{12} indicate the energy cost caused by transmitter electronics and receiver electronics respectively, α_2 represents the energy radiated via the power amplifier, d is the distance from the source node to the destination node, n is the path loss exponent. The value of α_2 depends on the path loss exponent n. In Heinzelman's dissertation [16], she used 87 meters as the "cross-over distance" for the path loss exponent. If d is smaller than the cross-over distance, she chose $n = 2$ and $\alpha_2 = 10\ pJ/bit/m^2$. Otherwise $n = 4$ and $\alpha_2 = 0.0013\ pJ/bit/m^4$ were chosen. In our simulation the communication range never exceeds 87 meters so we use $n = 2$ and $\alpha_2 = 10\ pJ/bit/m^2$. We also use the following values as provided in [16]: $\alpha_{11} = \alpha_{12} = 50\ nJ/bit$.

Eq. 3.7 implies that the radio wave scatters as it propagates towards the destination node and the average received signal power is modeled as inversely proportional to the n^{th} power of the distance. As a result, in order to achieve the same amount of received signal (as required by signal-to-noise ratio), we need to enlarge the transmission power to the n^{th} power of the distance.

Note that in our energy consumption model, we neglect the energy cost for sensing as well as processing and focus on the cost for communication only. Since we are assuming that the sink has unlimited power supply, our energy consumption model does not take the energy cost of the sink into consideration. In our implementation of the recovery scheme, the energy cost of all the sensor nodes in every simulation round falls into the following categories:

- receiving query messages and transmitting query responses, denoted as E_{query}
- sending and receiving sensed value updates to/from neighbors, denoted as E_{update}
- receiving recovery request messages and transmitting the corresponding responses, denoted as $E_{recovery}$

To come up with the energy consumption cost model, a few notations are used to facilitate the representation. The notations along with the corresponding description and default values are listed in Table 3.1. All the energy related notations are defined from the perspective of sensor nodes. Note N_{in} is related to a specific simulation run, N_{data} and N_{out} are associated with the chosen R_{parity} node for a recovery process in a specific simulation run, L_{resp2} depends on which round the message is generated in. The explanation and equations for calculating notations that do not have default values follow.

Since we assume that the sensor nodes always transmit messages using a fixed transmission power, $E_{tx_{node}}$ can be calculated according to Eq. 3.7 using the communication range $W = 60$ as the distance:

$$E_{tx_{node}} = \alpha_{11} + \alpha_2 \times W^2 = 86\ nJ \tag{3.9}$$

Notation	Description	Value
X	Dimension of the deployment area on X axis	500 m
Y	Dimension of the deployment area on Y axis	500 m
N	Total number of the deployed nodes	400
N_{in}	Number of nodes within the query area	-
N_{nb}	Average number of neighbors per node	Eq. 3.10
N_{hop}	Average number of hops from sensors to sink	Eq. 3.14
N_{data}	Number of R_{data} nodes in a recovery process	-
N_{out}	Number of R_{data} nodes outside of the query area	-
W	Communication range	60 meters
$E_{tx_{node}}$	Energy used to transmit a bit to a sensor node	$86\ nJ$(Eq.3.9)
$E_{tx_{sink}}$	Average energy used to transmit a bit to the sink	Eq. 3.15
$E_{rx_{node}}$	Energy used to receive a bit from a sensor node	$50\ nJ$ [16]
$E_{rx_{sink}}$	Energy used to receive a bit from the sink	$50\ nJ$ [16]
L_{query}	Size of the query message	256 bits
L_{resp1}	Size of the response message to the query	64 bits
L_{resp2}	Size of the response message to the recovery request	-
L_{update}	Size of the sensed value update message	64 bits
L_{req}	Size of the recovery request message	64 bits

Table 3.1: Notations used in the energy consumption model

If N nodes are uniformly distributed on a deployment field with dimension $X \times Y$, the average number of neighbors, N_{nb}, can be estimated by Eq. 3.10:

$$N_{nb} = \frac{N\pi W^2}{XY} - 1 \tag{3.10}$$

Note we do not implement the multi-hop routing protocol between sensor nodes and the sink since designing an adaptive and fault-tolerant routing protocol is beyond our scope. Instead, we present an analytical analysis of the energy consumption of multi-hop routing and include it in the cost model. To calculate the energy cost of a multi-hop routing path, one needs to know how many hops on average, N_{hop}, are needed to send the data from the source node to the sink. Then the multi-hop routing can be considered as N_{hop} hops of direct communication between neighboring nodes. N_{hop} can be estimated as the quotient of the distance between source node and the sink, D_{sink}, divided by the average length of the orthogonal projection of the direct communication paths onto the direction of D_{sink}, denoted as D_{proj}. That is, the following equation holds true:

$$N_{hop} = \frac{D_{sink}}{D_{proj}} \tag{3.11}$$

An example of the orthogonal projection is shown in Fig. 3.5. In this example, node 1 can send its data to the sink via node 2 and node 3. The path from node 1 to node 2, D_{12}, is now projected onto the direction from node 1 to the sink and the projection is denoted as $D_{12_{proj}}$. For large size WSNs, the distance from nodes within the

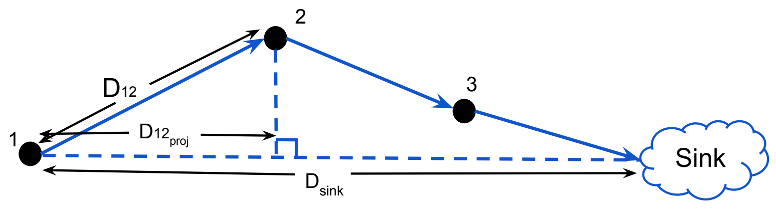


Figure 3.5: Projection of a direct communication path on the direction towards the sink

query area, R_{data} nodes and R_{parity} nodes, to the sink can be approximated as the distance between the center of the query area to the sink. Given the coordinates of query area center (Q_x, Q_y) and the coordinates of sink (S_x, S_y), this approximate distance D_{sink} is:

$$D_{sink} = \sqrt{(S_x - Q_x)^2 + (S_y - Q_y)^2} \tag{3.12}$$

In [9], D_{proj} is given as:

$$D_{proj} = \frac{2W}{3} \cos \frac{\pi}{2N_{nb}} \tag{3.13}$$

Then the average number of hops between sensors to the sink can be approximated as:

$$N_{hop} = \frac{\sqrt{(S_x - Q_x)^2 + (S_y - Q_y)^2}}{\frac{2W}{3} \cos \frac{\pi}{2N_{nb}}} \tag{3.14}$$

The average energy used to transmit a bit to the sink, $E_{tx_{sink}}$, can be calculated as the total energy cost of N_{hop} hops of direct communication:

$$E_{tx_{sink}} = N_{hop} \times (E_{tx_{node}} + E_{rx_{node}}) - E_{rx_{node}} \tag{3.15}$$

The $-E_{rx_{node}}$ part is due to the fact that the source node does not need to receive any message.

After explaining all the notations, we begin to derive the energy cost for each category. For E_{query}, in each round the sink generates a query message and floods the message to all the sensors within the deployment field. Thus, all the nodes (N) need to pay the price of receiving the query message ($E_{rx_{sink}} \times L_{query}$). On the other hand, only nodes within the query area (N_{in}) will respond to the sink and pay the price of transmitting the response messages ($E_{tx_{sink}} \times L_{resp1}$). Therefore, we have the model for E_{query} as shown in Eq. 3.16:

$$E_{query} = N \times E_{rx_{sink}} \times L_{query} + N_{in} \times E_{tx_{sink}} \times L_{resp1} \tag{3.16}$$

For E_{update}, in each round every node sends the update message containing the sensed value collected in this round to all its immediate neighbors, resulting in a transmission cost of $E_{tx_{node}} \times L_{update}$. At the same time, one node receives N_{nb} times of update messages which have been sent from its neighbors in the previous round. By summing up the transmitting and receiving energy cost, we have the model for E_{update}:

$$E_{update} = N \times E_{tx_{node}} \times L_{update} + N \times N_{nb} \times E_{rx_{node}} \times L_{update} \tag{3.17}$$

In the rounds when a recovery process takes place, we have another cost $E_{recovery}$ to be taken into account. If we denote E_{data} as the total cost at R_{data} nodes and denote E_{parity} as the total cost at R_{parity} node, the total energy cost for recovering a failed node is:

$$E_{recovery} = E_{data} + E_{parity} \tag{3.18}$$

As discussed in section 3.4, there are two different recovery approaches. For the centralized approach, both R_{parity} node and R_{data} nodes interact with the sink. They receive the request messages from the sink and send the requested information back to the sink. Note only R_{data} nodes that reside outside of the query area receive the request messages. The E_{data} and E_{parity} for this approach are given in Eq. 3.19 and Eq. 3.20 respectively.

$$E_{data} = N_{out} \times E_{rx_{sink}} \times L_{req} + N_{out} \times E_{tx_{sink}} \times L_{resp2} \tag{3.19}$$

$$E_{parity} = E_{rx_{sink}} \times L_{req} + E_{tx_{sink}} \times L_{resp2} \tag{3.20}$$

For the localized approach, there are N_{data} number of R_{data} nodes and each of them receives from and responds to the chosen R_{parity} node. E_{data} is shown in Eq. 3.21. The R_{parity} node receives the notification from the sink, broadcasts the recovery request message to R_{data} nodes, receives all the responses from the R_{data} nodes, as shown in Eq. 3.22.

$$E_{data} = N_{data} \times E_{rx_{node}} \times L_{req} + N_{data} \times E_{tx_{node}} \times L_{resp2} \tag{3.21}$$

$$E_{parity} = E_{rx_{sink}} \times L_{req} + E_{tx_{node}} \times L_{req} + N_{data} \times E_{rx_{node}} \times L_{resp2} \tag{3.22}$$

3.6 Network lifetime

The limited energy supply of sensor nodes results in the critical need for energy efficient protocols and prolonged network lifetime. After all, a network that can easily run out of energy is not practical for real-life deployment. In WSN research, network lifetime has long been considered as one of the most important parameters for evaluating WSN and WSN algorithms. Many research efforts have been spent on maximizing network lifetime [22, 43, 35]. Other important parameters, such as connectivity and coverage, are also related to network lifetime. While some authors use the term "network lifetime" without giving the exact definition, most of the authors do make it clear how the network lifetime is defined in their work. As the design of WSN depends heavily on the specific application requirements, the definition and metrics for estimating network lifetime is also application specific.

In [11], the authors give a comprehensive classification of network lifetime definitions presented in research works. They classify the existing definitions based on the metric(s) each definition considers. The metrics lie in the following aspects:

- the number of alive nodes
- sensor coverage
- connectivity
- application quality of service requirements

Each of the definitions has its own merit to be considered. For example, since the primary goal of sensor networks is to cover and monitor the deployment field, associating network lifetime with how the region of interest is covered by sensor nodes seems to be natural. Two common definitions under this category are α-coverage and k-coverage. α-coverage considers the time duration during which at least α portion of the region of interest is covered by at least one sensor [44], while k-coverage network lifetime requires that the area of interest is covered by at least k sensors [21].

Sensor coverage is an important aspect of WSN applications. However, sufficient sensor coverage must be complemented with satisfactory connectivity in order to transmit data from source to the sink. Thus some metrics of network lifetime also take the connectivity of the network into consideration. Connectivity has been used as a network lifetime metric in wireless ad hoc networks in which there is no notion of sensor coverage. Hence, in wireless ad hoc networks the network lifetime is sometimes defined as the time that the size of the largest connected component drops below a specified threshold [6]. In the context of WSNs, connectivity can be measured in terms of the packet delivery ratio at the sink node [39]. Many researchers propose to combine sensor coverage and connectivity in the definition of WSN network lifetime, e.g., in [39] the network lifetime is considered as the continuous operational time of the system before either the coverage or the packet delivery ratio drops below a specified threshold.

Coverage and connectivity are both essential properties for evaluating WSNs. However, we aim to propose a fault recovery scheme which can reconstruct the data of a failed node with 100% confidence. Neither coverage nor connectivity can measure the level of redundancy and fault tolerance in our scheme. Thus, they are not suitable metrics for the network lifetime definition.

Another common category of network lifetime definition is associated with the number of alive nodes. For a network composed of n sensor nodes, it can be described as n-of-n lifetime where the network lifetime ends as soon as the first node fails or k-of-n lifetime where the network lifetime represents the time during which at least k out of n nodes are alive. The advantage of this definition is its simplicity. Due to its simplicity, one may argue that it does not yield a realistic and meaningful estimation of network lifetime. However, as authors in [24] point out, this kind of strict definition is only applicable to WSN applications in which all nodes are of equal importance and critical to the network operation. Our scheme is just proposed for applications with such characteristics. Thus, our definition is a variant of this category.

By taking the specific goal of our scheme into account, we define network lifetime as: the time interval from the point that a WSN starts operation to the point that a node failure is observed and can not be recovered. This definition captures the fact that our proposed scheme can overcome node failures for a period of time and the sensors can respond to queries continuously, as if there were no node failures in the network. The recovery process can be transparent to the end users and the user will not notice any discrepancy in the query results before the network lifetime ends.

Chapter 4

Experimental Setup

4.1 Sinalgo framework

In our experiments, we use the network simulator Sinalgo (Simulator for Network Algorithms), which is developed by Distributed Computing Group at ETH Zurich, Switzerland[1]. Sinalgo differs from other network simulators such as OMNeT++[2] and ns-3[3] in a sense that it is designed to facilitate the verification and prototyping process of network algorithms instead of simulating details of network protocol stack or communication channel. Some of the features that make Sinalgo attractive to us include:

- Quick prototyping
- High performance
- Straight forward extensibility

Sinalgo supports two modes, synchronous and asynchronous mode. Synchronous mode is based on rounds and events could take place in parallel within such rounds. Asynchronous mode is discrete event driven. Since our recovery scheme does not require a discrete event driven feature, we have decided to choose the synchronous mode. The calling sequence of a synchronous simulation in Sinalgo is shown in Alg. 4. At the beginning of each round, we randomly disable selected nodes using

[1]http://disco.ethz.ch/projects/sinalgo/index.html
[2]http://www.omnetpp.org/
[3]http://www.nsnam.org/

a predefined node failure probability p to reflect that nodes are error-prone at any round, as shown from line 3 to line 7 in Alg. 4.

Algorithm 4 Calling sequence in Sinalgo

```
1:  while round number $r <$ threshold do
2:      $r = r + 1$
3:      for every sensor node $i$ do
4:          if node $i$.alive == true and RNG$< p$ then
5:              $i$.alive == false
6:          end if
7:      end for
8:      handle global timers
9:      adjust according to mobility/connectivity/interference models
10:     for every node in the simulation do
11:         gather all the messages this node has received
12:         handle timers that fire in this round
13:         process the arriving messages
14:     end for
15: end while
```

4.2 Models

Sinalgo adopts the idea of models to achieve high extensibility of the simulation environment. Models are used to describe the characteristics of nodes. There is one global model, i.e., message transmission model, which is an instance owned by the whole simulation framework. For each node, it has five instances of node specific models: distribution, connectivity, mobility, interference, and reliability model. Node specific models are instances that a node will have upon creation and different nodes can have different implementation of models. The main function of each model is listed in Table 4.1 and a detailed explanation of these models follows.

Transmission model This model is applicable to all nodes and all messages transmitted. In our experiments, we set a constant time delay of 1, which indicates that it takes one round for a message to arrive at its target.

Model	Description
Transmission	How long it takes a message to arrive at its destination
Distribution	Initialize the positions of the nodes
Connectivity	If two nodes can be considered as connected
Mobility	How the nodes change positions
Interference	Whether simultaneous transmissions would interfere
Reliability	The possibility that one message may be dropped

Table 4.1: Different models and their functions in Sinalgo

Distribution model We use a uniform node distribution model and a grid node distribution model in our experiments. The uniform model randomly scatters the nodes in the simulation area while the grid model places the nodes on the intersection points of a grid which covers the entire deployment area. In order to calculate the position of each node, both models need the dimensions of the deployment area as the input. The grid model also takes the total number of nodes into consideration while the uniform model utilizes a random number generator to initialize the node positions.

Connectivity model As an important property of wireless networks, connectivity has been investigated by many researchers. Several factors can affect the connectivity of a network including but not limited to network density, communication range, initial deployment. In our simulation, the Unit Disk Graph (UDG) is used as the connectivity model. UDG is widely used in WSN topology design as a geometric graph theory model. A pair of nodes are considered as connected *iff* the distance between them is no greater than a certain specified threshold. Figure 4.1 (a) and (b) are the graphic representation of the UDG model for uniform node distribution and grid node distribution, respectively. In this figure, the diameter of the disk around each node represents the distance threshold. Thus, two nodes are connected *iff* there is intersection between the corresponding disks.

Mobility model This model defines how the nodes move. For example, a node may move according to a direction or move completely randomly. As discussed in

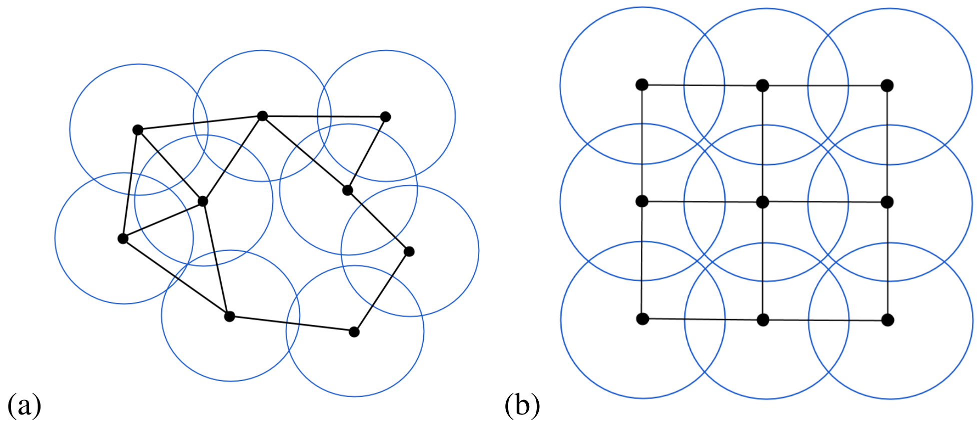


Figure 4.1: UDG model for (a) uniform node distribution and (b) grid node distribution

section 3.1, our scheme relies on static topology. Thus, no mobility model has been utilized.

Interference model This model decides if the received message should be dropped due to the interference caused by either environmental noise or message collisions at this node. Our experiments assume that there is no interference.

Reliability model This model is used to simulate a lossy network in which messages will be dropped for whatever reason. Since we assume that all the transmissions are reliable, we incorporate the ReliableDelivery model which does not drop any message.

4.3 Message types

There are eight different types of messages involved in the simulation. The message types, senders, receivers and functions are shown in Table 4.2.

UpdateMsg Our scheme requires that when a node captures some new data, a copy of the newly captured data is sent to all its immediate neighbors. This is done by using UpdateMsg messages. In each round, a node deals with UpdateMsg twice. When a node updates its own sensed value, it also broadcasts a UpdateMsg message to all its neighbors. Then this node needs to gather all the UpdateMsg messages it

Message Type	Sender	Receiver	Function
UpdateMsg	Sensor	Sensor	Broadcast updates to neighbors
QueryMsg	Sink	Sensor	Specify the query area and type
QueryResp	Sensor	Sink	Contain the query response
ReqParityMsg	Sink	R_{parity}	Ask for $ParityData$
ReqParityResp	R_{parity}	Sink	Contain $ParityData$
ReqDataMsg	Sink /R_{parity}	R_{data}	Ask for $SensedData$
ReqDataResp	R_{data}	Sink /R_{parity}	Contain $SensedData$
NotifyMsg	Sink	Sink /R_{parity}	Notify the receiver to initiate a localized recovery

Table 4.2: Different message types in our simulation

has received from its neighbors and conduct XOR operations and append the calculation results in the parity section. Then the UpdateMsg messages are discarded.

QueryMsg The query area for a simulation run is set prior to any message transmission. In each round of the simulation, the sink will send a QueryMsg message to all the nodes specifying the center and the radius of the query area. When the sensors receive this message, they will calculate to decide if they are within the query area. In our experiments, we specify that the QueryMsg message asks for the sensed values which the sensors have collected in the most recent simulation round.

QueryResp Once a node has received a QueryMsg and has decided that it is indeed within the query area, this node will immediately generate a QueryResp message and send this message back to the sink. Sensors include the sensed values they have collected in the most recent simulation round in the QueryResp message.

The next five types of messages are directly associated with the recovery process. The message exchange diagram of both centralized and localized recovery approach is shown in Fig. 4.2.

ReqParityMsg This type of message is only applicable to the centralized recovery approach. When the sink is aware of a node failure and has chosen one neighbor

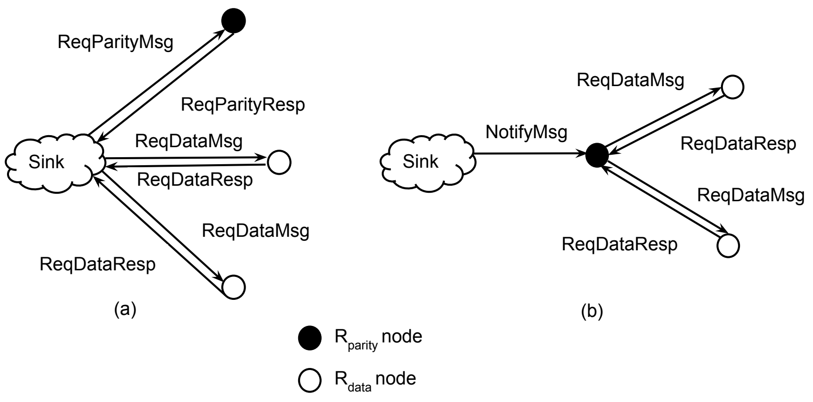


Figure 4.2: Message exchange in the recovery process for (a) centralized approach and (b) localized approach

of the failed node as the R_{parity} node, the sink needs to send a ReqParityMsg message asking for the $ParityData$ and deliver it to R_{parity}.

ReqParityResp Upon receiving the ReqParityMsg from the sink, the sensor node will send a ReqParityResp message containing all its $ParityData$ to the sink to facilitate the data recovery process.

ReqDataMsg In the centralized recovery approach, the sink needs to send a ReqDataMsg message to each R_{data} node that resides outside of the query area for their $SensedData$. In the localized recovery approach, it is the R_{parity} node which sends this type of message to all the R_{data} nodes.

ReqDataResp When the R_{data} node receives the ReqDataMsg message, it will generate a ReqDataResp message containing all its $SensedData$ as a response and send it back to the sender of the corresponding ReqDataMsg message.

NotifyMsg This type of message applies to the localized recovery approach only. When the sink decides which node is chosen as R_{parity} node for the recovery, the sink sends a NotifyMsg message to the R_{parity} node to notify that a localized recovery is needed.

Key <Sensor>	Value <Data>
2	1,5,3,2,9,3,6,...
...	...
15	4,8,6,0,1,3,5,...
...	...
42	8,3,6,4,2,1,6,...
...	...
58	7,6,3,7,1,9,2,...

Table 4.3: Excerpt from DataTable

4.4 Data structure at the sink

There are two types of nodes in our simulation: sink and sensors. The sink is a super node with enough storage and computing capacity. As the core component of the proposed fault recovery scheme, the sink maintains two data structure: a hash map named DataTable for storing the data of the received QueryResp messages and a hash map NeighborTable for storing the network topology.

In the centralized recovery approach, the sink needs the $SensedData$ of the R_{data} nodes. In case some of the R_{data} nodes are within the query area, they have already sent their sensed values of each round in the QueryResp messages. Thus, a DataTable is maintained to keep track of such values so that the sink does not need to send new requests to these R_{data} nodes. In general, DataTable is used to store the $SensedData$ of the nodes that are within the query area, based on the QueryResp messages originating from these nodes. The hash map entries can be presented as <Sensor; Data>pairs. When the sink receives a QueryResp message, the sink will update Data of the corresponding message sender in the DataTable by appending the sensed values wrapped in the QueryResp message to the end of the original Data. Assume that the sensed values are integers, an excerpt of the DataTable is given in Table 4.3.

NeighborTable is incorporated so that the sink can have a complete graph of the network topology which can be used for the recovery process. It can be represented as <Key; Value> pairs. Key stores sensor node objects while Value is a class which contains Degree and NodeList. Degree is simply the size of NodeList and NodeList

Key <Sensor>	Value <Degree;NodeList>
2	7;<15, 34, 60, 73, 91, 94, 98>
...	...
15	3;<2, 42, 58>
...	...
42	6;<15, 29, 58, 64, 94, 95>
...	...
58	5; <15, 42, 64, 94, 95 >

Table 4.4: Excerpt from NeighborTable

is the vector composed of the key's immediate neighbors. More specifically, the hash map entries can be presented as <Sensor; <Degree; NodeList>>. Table 4.4 is an excerpt from the NeighborTable built in one simulation run.

Here is a simple example of how to use both hash maps to fulfill a centralized recovery. Assume that the sink detects node 15 has just failed. The sink searches NeighborTable and knows that node 15 has three neighbors, i.e., node 2, 42, 58. Assume that all these nodes are valid recovery candidates. According to Alg. 1, node 58 is chosen as the best recovery candidate since it has the lowest degree among all the recovery candidates. Now we need the $SensedData$ of node 42, 64, 94 and 95 and $ParityData$ of node 58 to fulfill the recovery of Node 15. Then the sink sends a ReqParityMsg to Node 58 and Node 58 wraps all its $ParityData$ in the ReqParityResp message. Meanwhile, the sink looks up in the DataTable for tuples of node 42, 64, 94 and 95. If any tuple is missing, the sink generates ReqDataMsg to that node requesting for its $SensedData$. By XORing $SensedData$ of node 42, 64, 94, 95 and $ParityData$ of node 58 we can recover the data of node 15.

Chapter 5

Results and Analysis

As discussed in section 3.2, our recovery scheme is able to prolong the network lifetime by incorporating data redundancy into wireless sensor networks. Therefore, we are interested in to what extent the proposed scheme can extend the network lifetime under different scenarios and network settings. On the other hand, our scheme inevitably requires more message transmissions for both the recovery process and the synchronization of sensed values between sensor nodes and their neighbors, which impose additional communication overhead. Thus we conduct experiments and use the energy cost model as defined in section 3.5 to investigate the energy cost for each round. In section 5.6 the effect of various initial energy supply on the network lifetime is discussed. To the best of our knowledge there is no other approach that we could use in order to make a fair comparison with respect to the our experimental results. We only use the no recovery scheme for comparison.

Four key objectives guide the design of the following experiments:

- showing the trend of network lifetime as the node failure probability increases
- investigating the relations between network lifetime and different network parameters
- showing how the size of the query area affects the network lifetime
- presenting the energy cost according to the energy consumption model in section 3.5

As defined in section 3.6, we consider the time interval from the point that a WSN starts operation to the point that a node failure is observed and can not be recovered as the network lifetime. Since Sinalgo adopts the concept of round to achieve clock synchronization within the network, we use the number of rounds to indicate the network lifetime. The first round a node failure takes place and can no longer be recovered is considered as the end of the network lifetime.

In each section from 5.1 to 5.4, we vary one parameter and fix the others to investigate the impact of the varied parameter. For a specific parameter setting, each point in the figure represents the average result of S different simulation runs using distinct seeds. We choose $S = 20$ for section 5.1- 5.4. Since the network lifetime is a random variable, a 95% confidence interval of the S samples is included to give an estimate of the true network lifetime. For a fixed parameter setting, we denote the observed network lifetime of the i^{th} simulation run as $T_{i_{rec}}$ and $T_{i_{no}}$ for the recovery scheme and no recovery scheme respectively. The average network lifetime using this parameter setting is defined as the sample mean of the S runs. The average network lifetime for the recovery scheme, denoted as T_{rec}, is given in Eq. 5.1 and the average network lifetime for no recovery scheme, denoted as T_{no}, is given in Eq. 5.2.

$$T_{rec} = \frac{1}{S}\sum_{i=1}^{S} T_{i_{rec}} \tag{5.1}$$

$$T_{no} = \frac{1}{S}\sum_{i=1}^{S} T_{i_{no}} \tag{5.2}$$

We also define a performance gain ratio, G, to indicate how much performance gain the recovery scheme can have over no recovery scheme:

$$G = \frac{T_{rec}}{T_{no}} \tag{5.3}$$

The sensor deployment field is in the size of $500\ m \times 500\ m$. A sink is located at the center of the deployment field and the sensors are created using the models as specified in section 4.2. A few notations used in this chapter are listed in Table 5.1. In each section from 5.1 to 5.4, we vary each of the first four parameters according to the given range and fix the other three parameters using their default values.

Notation	Description	Values
p	The probability that one sensor would fail in one single round	[0.05%, 0.5%] Default: 0.1%
W	Communication range of sensors	[40, 80] m Default: 60 m
N	Total number of deployed nodes	[200, 800] Default: 400
Q	The percentage of the deployment field that is covered by the query	[5%, 30%] Default: 10%
N_{in}	Number of nodes within the query area	Eq. 5.7
N_{nb}	Average number of neighbors per node	Eq. 3.10
L	The number of rounds until the first node failure takes place	-
P_{fail}	The probability that at least one failure happens within the query area in each round	Eq. 5.4
P_{valid}	The probability that a recovery candidate is considered as valid	Eq. 5.6

Table 5.1: Notations used in the experiments

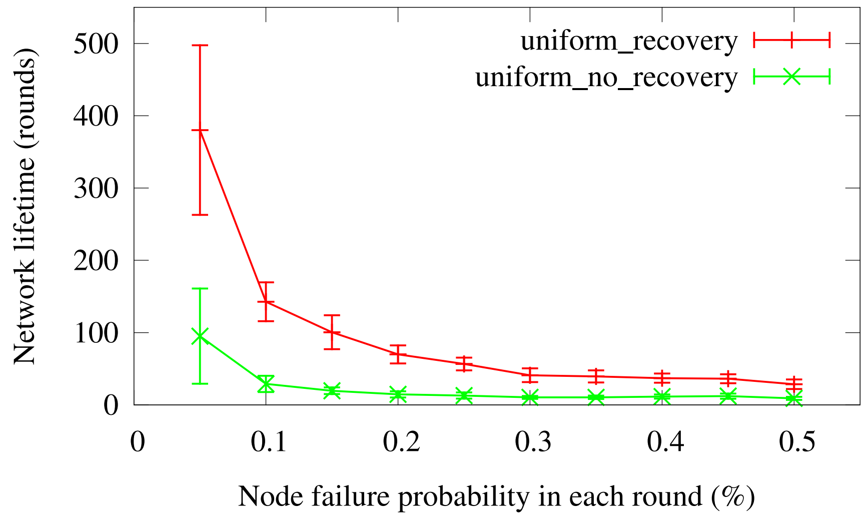
5.1 Node failure probability

Since the motivation of proposing a fault tolerant scheme is to cope with node failure, we want to investigate how the scheme performs under different node failure probability ratios. The node failure probability, p, indicates the probability that one sensor node would fail due to whatever reasons in one single round. The value of node failure probability reflects the reliability of the sensor nodes and the environmental conditions. Even though the numerical value of node failure probability is difficult to be measured in real WSN applications, we use it as a conceptual indication of the error-prone nature of the WSNs. For poor quality sensors or extremely harsh environment, the node failure probability should be relatively high. We consider whether one sensor node fails as a Bernoulli trial with the probability p to observe node failure. The probability distribution of node failure among different nodes is *i.i.d.*

To investigate the trend of network lifetime and performance gain ratio under various node failure levels, we vary the node failure probability in each round from

0.05% to 0.5% with 0.05% as the increment. The total number of deployed nodes N, the communication range W and the query size percentage Q are set to be their corresponding default values.

The network lifetime of uniform node distribution and grid node distribution is shown in Fig. 5.1(a) and (b). For both node distributions, as the node failure probability increases, the network lifetime of both schemes drops.



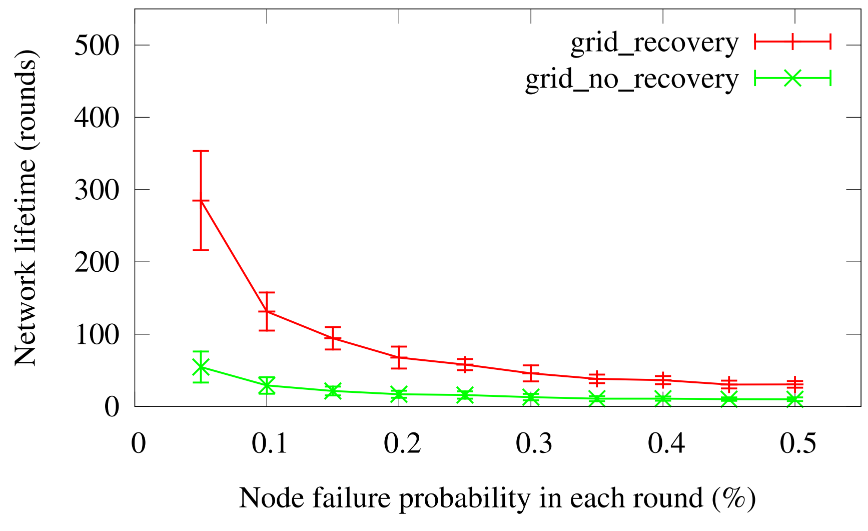


Figure 5.1: Network lifetime under different node failure probability for (a) uniform node distribution and (b) grid node distribution

For no recovery scheme, the probability that at least one node failure takes place within the query area during one single round, P_{fail}, is given by Eq. 5.4:

$$P_{fail} = 1 - (1-p)^{N_{in}} \tag{5.4}$$

For this experiment, N_{in} is considered to be fixed since neither the query size nor the network density changes. When we increase p in the range of (0, 1), the resulting P_{fail} increases monotonically. This implies that within a round, there is a higher possibility that at least one node failure may take place. The number of rounds until the first node failure occurs, L, is a geometrically distributed random variable. The expected value of L is given by Eq. 5.5:

$$E[L] = \frac{1}{P_{fail}} \tag{5.5}$$

Therefore, when P_{fail} increases, $E[L]$ decreases, which indicates that the first node failure is expected to come earlier when the node failure probability becomes larger. Thus, we observe a shorter network lifetime for no recovery scheme.

For our recovery scheme, the network lifetime depends on when a failed node can no longer be recovered. The recovery algorithm as defined in section 3.3 builds a recovery candidate set R and a valid candidate set R'. How p varies will not affect the composition of R. However, the size of R' is associated with p. For a specific recovery candidate C_i in R, there are two requirements that C_i must meet in order to join R': first, C_i is alive and second, all the direct neighbors of C_i, except the one we aim to recover, are alive. Thus, the probability that C_i is considered as valid, P_{valid}, is given in Eq. 5.6.

$$P_{valid} = (1-p)^{N_{nb}} \tag{5.6}$$

When p increases, P_{valid} decreases monotonically, which means that each recovery candidate experiences a more difficult time being chosen as a valid candidate. As a consequence, R' has a higher chance to be an empty set, resulting a shorter network lifetime. Therefore, the network lifetime of the recovery scheme is also expected to decrease as the node failure probability increases.

Since the network lifetime of both schemes is expected to decrease, the performance gain ratio is worth investigating. As shown in Fig. 5.2, with uniform node distribution, the maximum value of G is achieved when $p = 0.15\%$ and afterwards

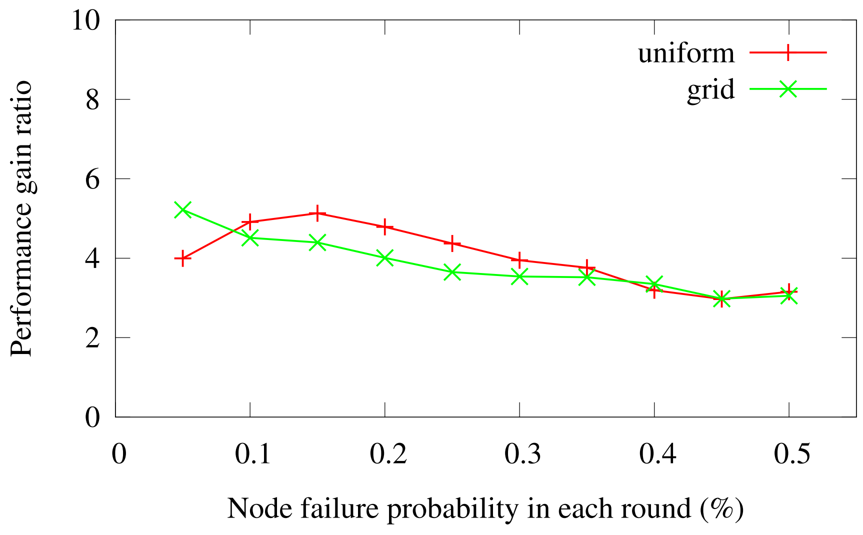


Figure 5.2: Performance gain ratio under different node failure probability

G drops. For grid node distribution, G monotonically decreases as p increases. This means that for $p \geq 0.15\%$ the lifetime performance drop of the recovery scheme is greater than the no recovery scheme, which indicates that the increase of p has a relatively larger impact on the recovery scheme. G decreases by 42.1% and 42.8% from the maximum value to the minimum value for uniform and grid node distribution, respectively. For the worst case, i.e., when the node failure probability is as large as 0.5%, the recovery scheme can still achieve a network lifetime three times as long as that of no recovery scheme.

5.2 Communication range

Given the same number of deployed sensor nodes, the communication range W has a large impact on the connectivity of the network. Controlling the communication range directly affects the number of neighbors a node can communicate with. Since our proposed recovery scheme depends on the neighborhood relations, it is worthwhile investigating how the network lifetime gain will be influenced by W. The first problem is how to set a reasonable range of W to conduct the experiment. A network composed of 400 nodes with a W, which is below 40 meters, has a relatively high chance of not being connected, while a network with a W larger than

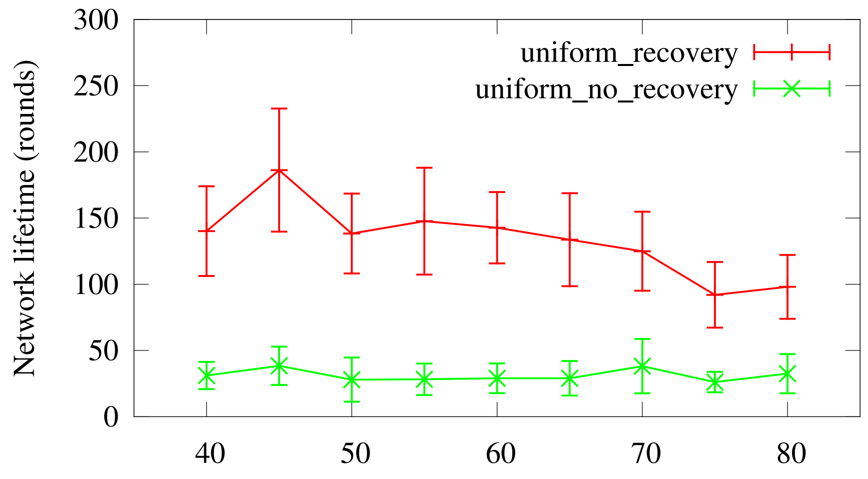
Communication range	N_{nb} of uniform dist.	N_{nb} of grid dist.
40	7.04	7.41
45	9.18	7.41
50	11.57	11.01
55	14.20	17.85
60	17.10	17.85
65	20.24	17.85
70	23.63	21.09
75	27.27	24.09
80	31.17	30.95

Table 5.2: Average number of neighbors using various communication range

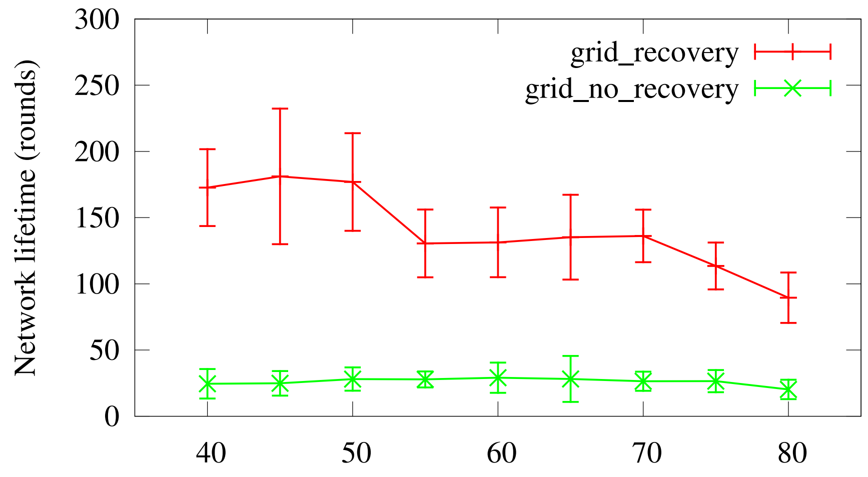
80 meters yields a large average number of neighbors (more than 30). Thus, in this experiment, we vary W from 40 meters to 80 meters with 5 meters as the increment. The other parameters, N, p, and Q are set using their default values.

As shown in Eq. 3.10, for a randomly and uniformly distributed WSN, the average number of neighbors per node is proportional to W^2 given that the total number of nodes and the dimension of deployment field are fixed. For grid distribution, the average number of neighbors does not grow linearly with W^2 because sometimes increasing W does not necessarily create more edges between nodes. Table 5.2 shows the average number of neighbors per node for uniform and grid node distribution as a function of W. Note that the numerical results of uniform node distribution are obtained from the calculation according to Eq. 3.10 while the results of grid node distribution come from our simulation. For uniform node distribution, since we use different seeds for every simulation run, the node positions of two simulation runs are never the same. For grid node distribution, the node positions are fixed as a 20×20 grid and changing W affects the neighborhood relations only.

Differently from section 5.1, both the size of recovery candidate set R and the size of valid candidate set R' can be affected by W. The increase of W brings more recovery candidates as N_{nb} increases. For a recovery candidate C_i in R, the probability that C_i is considered as valid is the same as defined in Eq. 5.6. As N_{nb} increases, P_{valid} decreases monotonically, which indicates that it is more difficult for C_i to be considered as a valid candidate. Thus, a larger W brings more recovery candidates, which has a positive effect on the recovery process. However, each recovery candidate has a lower probability to be valid, which has a negative impact on the network lifetime. The total effect of a larger recovery candidate set R and a lower P_{valid} for each recovery candidate depends on which factor plays a dominant role.



(a)



(b)

Figure 5.3: Network lifetime using different communication range for (a) uniform node distribution and (b) grid node distribution

The network lifetime for uniform and grid node distribution is shown in Fig. 5.3 (a) and (b), respectively. From Fig. 5.3 (a) we can see that when W varies from 40 to 45 meters, the network lifetime of the recovery scheme increases by 46 rounds, which equals to a 32.9% increase. The corresponding N_{nb} varies from 7.04 to 9.18. This performance improvement indicates that at this stage, increasing N_{nb} has an overall positive impact on prolonging the network lifetime since the positive effect of a larger R outperforms the negative effect of a lower P_{valid}. After this the network lifetime remains quite smooth from 50 to 70 meters and the corresponding N_{nb} is in the range from 11.57 to 23.63. In this period, the increase of N_{nb} does not significantly affect the network lifetime since the two factors offset each other. When W goes beyond 70 meters, a performance drop of 32 rounds takes place, indicating that the negative effect of a decreasing P_{valid} is dominant.

The analysis of the experiment results leads us to believe that there exists a certain threshold for N_{nb}, below which increasing W can have better lifetime performance and above which increasing W may even result in an opposite effect. More specifically, for this parameter setting, when N_{nb} is below 9.18, more neighbors means more recovery candidates and all candidates have a fairly large probability to become valid candidates. When N_{nb} goes beyond 9.18, even though increasing W indicates more recovery candidates, each recovery candidate experiences a more difficult time being chosen as the valid recovery candidate. In short, the average number of neighbors is a double-edge sword which can bring more recovery candidates with lower probability to become valid candidates. Thus the average number of neighbors should be carefully chosen. Under this parameter setting, a N_{nb} less than 20 seems to be suitable.

For grid node distribution, the network lifetime of the recovery scheme has a similar trend as shown in uniform node distribution. Note that for $W = 40$ and $W = 45$, the connectivity of the network is identical since N_{nb} is both 7.41. The same holds for $W = 55$, $W = 60$, and $W = 65$. Since the node distribution is fixed, the same connectivity implies the same network lifetime performance. When W is in the range of 40 - 50 meters, the network lifetime achieves the highest values. The corresponding N_{nb} is no greater than 11.01. After that the network lifetime decreases by nearly 50 rounds when W increases from 50 to 55 meters. The corresponding N_{nb} increases from 11.01 to 17.85, which can be considered to be a significant increase. As previously discussed, the considerable increase in N_{nb}

brings more recovery candidates which are less likely to be considered as valid, resulting in the lifetime decrease. The same performance degradation is observed from $W = 70$ to $W = 80$, which comes with a non-negligible increase of N_{nb}. We do not recommend the average number of neighbors go beyond 20 under current parameter setting.

The network lifetime of no recovery scheme does not change significantly as W increases. As discussed in section 5.1, P_{fail} is given by Eq. 5.4. In this experiment, both p and N_{in} are considered as fixed. Thus, increasing W will not affect the lifetime of no recovery scheme.

The performance gain ratio, G , as a function of the communication range, W, is shown in Fig. 5.4. In general, G decreases as we enlarge W for both distribu-

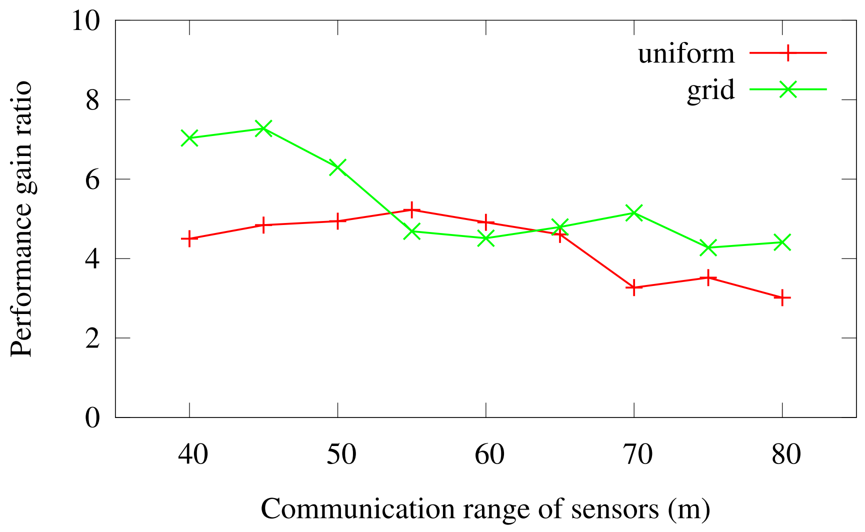


Figure 5.4: Performance gain ratio for different communication range

tions. For grid node distribution, the recovery scheme achieves a lifetime at least four times as long as that of no recovery scheme and G decreases by 39.2% in the best case to the worst case scenario. For uniform node distribution, the lifetime of recovery is at least three times as long as that of no recovery scheme and G decreases by 42.3% it the best case to the worst case scenario.

5.3 Node density

We are very interested in how the network topology affects the network lifetime and the performance gain ratio G. Typically, the topology determines which node is allowed to communicate with which other nodes. Since we already made a few assumptions such as we are dealing with flat WSNs and sensors are static after deployment, the problem of topology control is all about node density and communication range. In the previous section we discussed communication range and in this section we will explore the impact of node density.

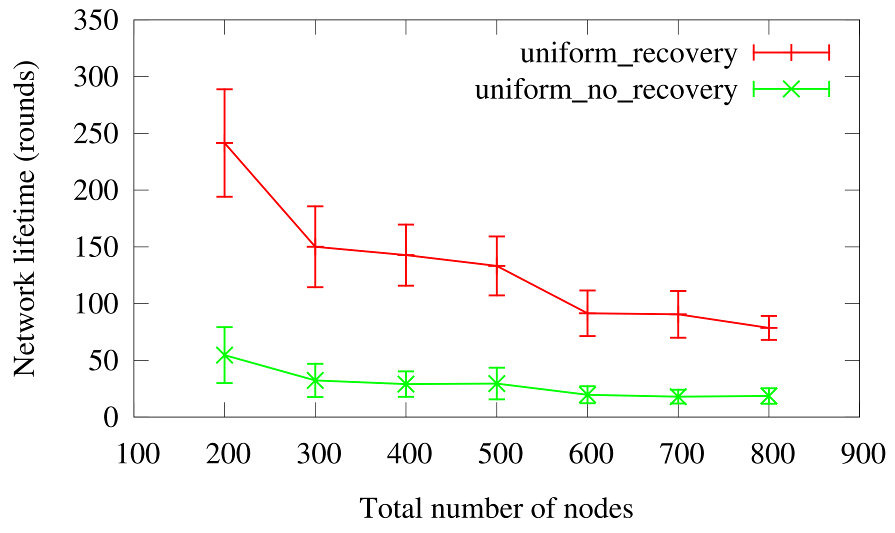
In this experiment, p, W and Q are set to be their default values. We vary the total number of deployed nodes from 200 to 800, with the increment being 100. If the total number of nodes is below 200, for a uniformly distributed network with $60\ m$ as the communication range, there is a higher chance that the network is not connected. If the total number of nodes exceeds 800, the average number of neighbors a node has will be greater than 32 for both node distributions. We consider any scenario beyond this point to be too dense for a real WSN application, since an extremely high density will cause problems for MAC and routing protocols. The results for uniform and grid node distribution are shown in Fig. 5.5 (a) and (b) respectively. In Fig. 5.5 (a) we can see that as the total number of nodes N increases, the network lifetime of both no recovery scheme and the recovery scheme decreases. For no recovery scheme, this trend is anticipated, since no recovery scheme considers the first node failure as the termination of network lifetime. For uniform node distribution in a given deployment field, the number of nodes that reside within the query area, N_{in}, is proportional to the total number of nodes, N, and the percentage of the deployment field covered by the query, Q, as shown in Eq. 5.7:

$$N_{in} = N \times Q \tag{5.7}$$

For grid distribution, the previously indicated relations holds true approximately. As shown in Eq. 5.4, when N_{in} increases, P_{fail} increases monotonically, which results in a lower expected value of L. Thus, the first node failure is expected to come earlier when the network becomes denser.

For the recovery scheme, the average number of neighbors N_{nb} increases as the

(a)



(b)

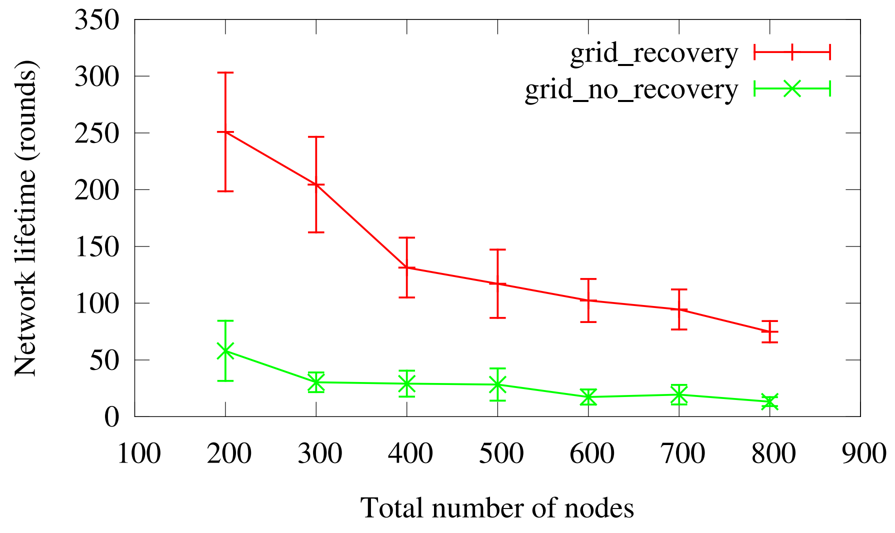


Figure 5.5: Network lifetime achieved with different total number of nodes for (a) uniform node distribution and (b) grid node distribution

network gets denser. For uniform node distribution, N_{nb} increases linearly with N as defined in Eq. 3.10 and for grid node distribution, N_{nb} increases monotonically with N. The N_{nb} of both node distributions with different total number of deployed nodes is given in Table 5.3.

Similar to the discussion in section 5.2 , a larger N_{nb} can not guarantee a longer lifetime for recovery scheme since each candidate within the recovery candidate set has a less likelihood to be valid. According to Eq. 5.6, a larger N_{nb} yields a smaller P_{valid}. However, the drop of P_{valid} is not the only reason why the recovery

Total number of nodes	N_{nb} of uniform dist.	N_{nb} of grid dist.
200	8.05	7.16
300	12.57	10.85
400	17.10	17.85
500	21.62	18.08
600	26.14	25.12
700	30.67	32.17
800	35.19	32.39

Table 5.3: Average number of neighbors using various total number of nodes

candidates are less likely to be considered as valid. Note P_{valid} of different recovery candidates can not be considered as independent. As the network becomes denser, there is a greater chance that two or more recovery candidates share some neighboring nodes in common. Once a common node fails, all the associated recovery candidates are considered as invalid and therefore can not be used to initiate the recovery process. Consider the example in Fig. 5.6. In this case, assume we are

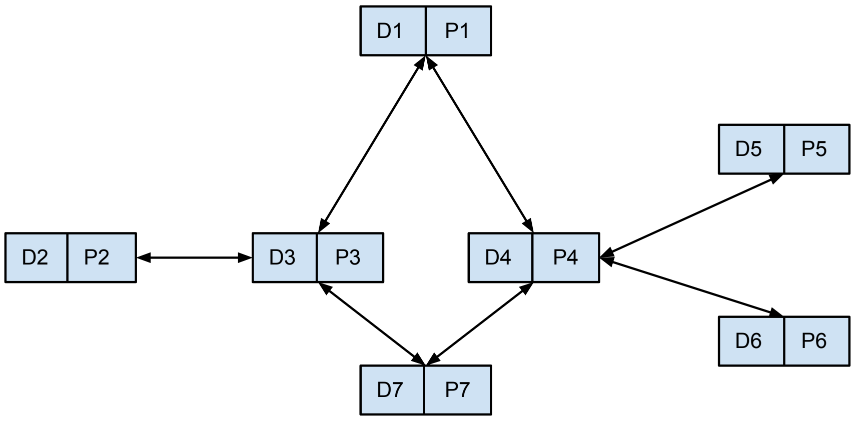


Figure 5.6: An example of common node shared by recovery candidates

trying to recovery node 1. Node 1 has two recovery candidates: node 3 and node 4. Different from the topology shown in Fig. 3.2, in this case, recovery candidate node 3 and node 4 share the same neighbor: node 7. Should node 7 have failed, neither node 3 nor node 4 can be considered as a valid candidate for the recovery. The denser the network is, the more likely that this scenario can be observed. Thus, apart from the decrease of P_{valid}, recovery candidates are less likely to meet the requirements of valid candidates as we increase the network density.

The performance gain ratio obtained when we increase the network density is shown in Fig. 5.7. We can see that for uniform node distribution, G does not sig-

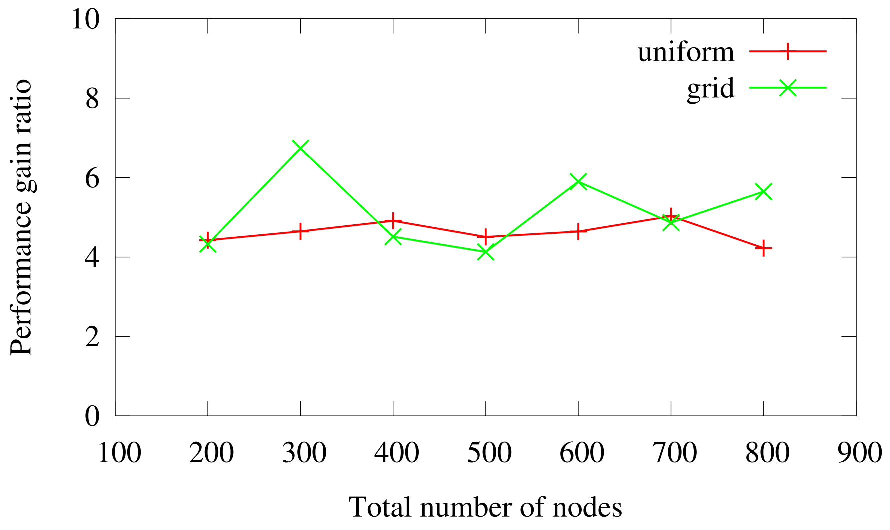
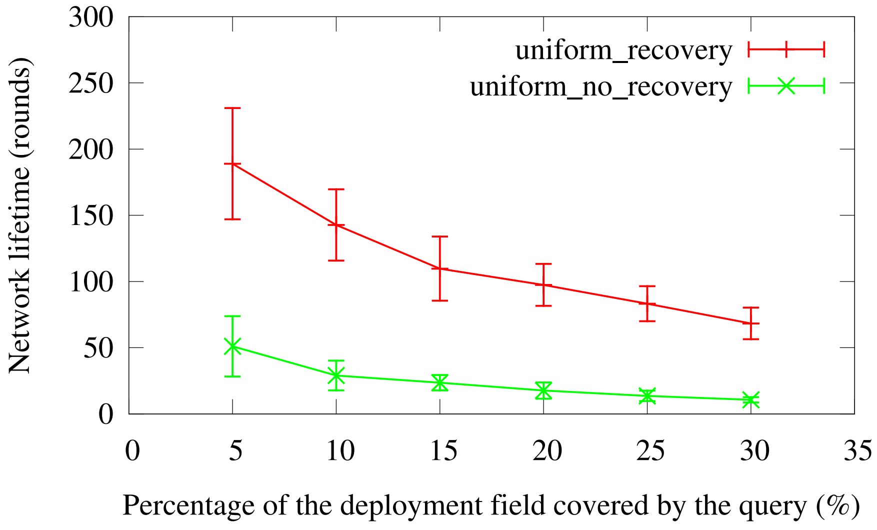


Figure 5.7: Performance gain ratio under different total number of nodes

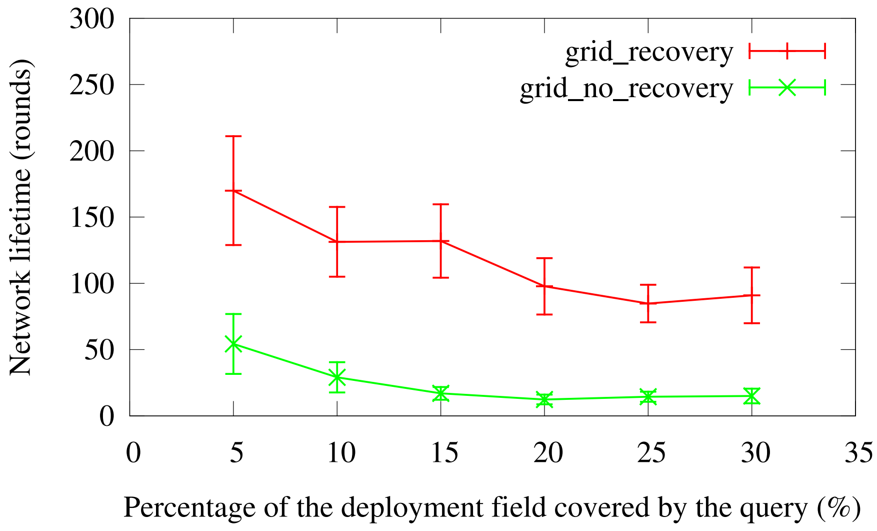
nificantly change as N increases. Our recovery scheme can achieve a lifetime four times as long as that of no recovery scheme regardless of the network density for uniformly distributed WSNs. The performance gain ratio for grid node distribution shows greater variance. This is due to the fact that the average number of neighbors of the grid node distribution does not grow linearly as the density increases. For example, from $N = 300$ to $N = 400$, G drops obviously and the corresponding N_{nb} has a significant increase from 10.85 to 17.85. From $N = 400$ to $N = 500$, G does not change significantly and the corresponding N_{nb} has a minor increase from 17.85 to 18.08. A 38.7% drop of G is observed from the best case to the worst case for grid node distribution.

5.4 Query size

Since our definition of network lifetime is associated with observed node failures within a given query area, one may question if the size of the query area matters to the outcome. In this experiment, we aim to explore how the query size will affect the lifetime gain of our scheme. We vary the percentage of the deployment field that is covered by the query from 5% to 30%. The total number of nodes is set to be 400 with a communication range of 60 meters. The node failure probability is set to be 0.1%. The network lifetime is shown in Fig. 5.8.



(a)



(b)

Figure 5.8: Network lifetime achieved using different query size for (a) uniform node distribution and (b) grid node distribution

From Fig. 5.8 we can see that for both node distributions, the network lifetime of both recovery scheme and no recovery scheme decreases as the query size increases. As shown in Eq. 5.7, N_{in} increases as we enlarge query size Q. Therefore, the network lifetime of no recovery scheme is expected to decrease since the first node failure is expected to come earlier, as defined in Eq. 5.4 and Eq. 5.5. For the recovery scheme, given this parameter setting where N_{nb} and p are fixed, the total number of recovery candidates and P_{valid} for each recovery candidate will not be affected by the query size. Thus, the probability that each node failure is recoverable stays the same no matter how the query size varies. However, as N_{in} increases, the probability that all the nodes are recoverable decreases. Thus, the first non-recoverable node failure is expected to co me earlier and a shorter network lifetime is observed for the recovery scheme when the query size increases.

The performance gain ratio of network lifetime for different query sizes is shown in Fig. 5.9. Combined with Fig. 5.8, we can conclude that even though the absolute value of network lifetime decreases as we enlarge the query size, the relative performance gain ratio does not. More specifically, the value G of uniform node distribution increases steadily. The result implies that the network lifetime degradation of no recovery is faster than that of recovery scheme. In other words, the proposed recovery scheme is more robust to the change of query size. In the worst case scenario, the recovery scheme can still achieve a lifetime 3.70 and 3.13 times as long as no recovery scheme for uniform and grid node distribution, respectively.

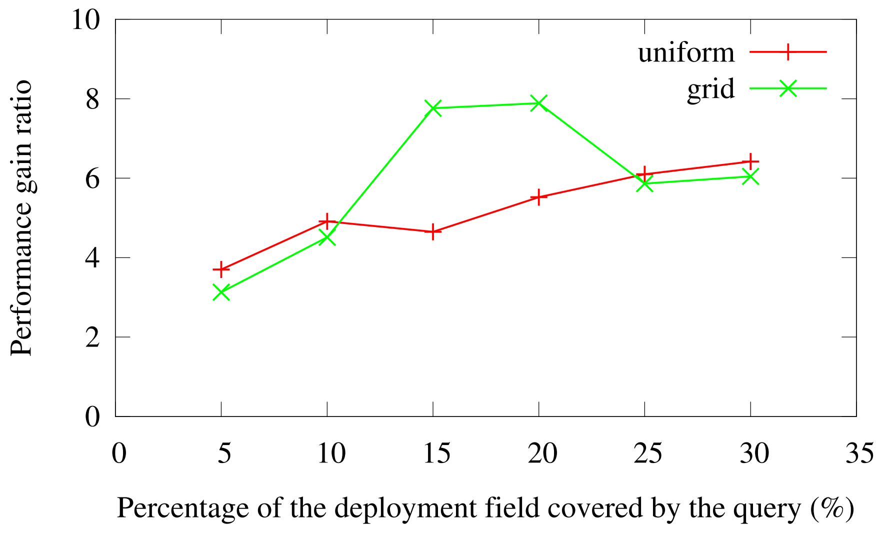


Figure 5.9: Performance gain ratio for different query size

5.5 Communication overhead

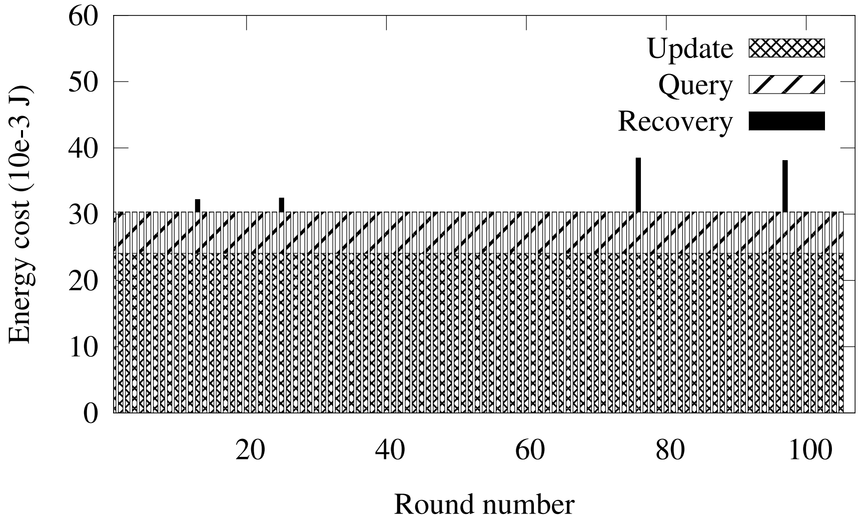
In section 3.5 we define the energy consumption model for the proposed recovery scheme. In this section, we use this model to calculate the energy cost of E_{update}, E_{query} and $E_{recovery}$ for each round. For the experiments in this section, p, W, N, Q are all set to be the default values.

As discussed in section 4.3, the query asks for the sensed values which the sensors have collected in the most recent simulation round. Table 3.1 lists the parameters we need for the calculation. Note that the QueryMsg message has a fixed size of 256 bits. The QueryResp message and the UpdateMsg message has a fixed size of 64 bits, since we assume that the size of the sensed values in one round is 64 bits. All the recovery request messages, including ReqParityMsg, ReqDataMsg, and NotifyMsg, have a fix size of 64 bits too. The size of the recovery request response message, i.e., ReqParityResp and ReqDataResp message, grows linearly with the round number. Denote the round number as T, given that the size of the sensed value in one round is 64 bits, the size of the recovery request response message in the T^{th} round is:

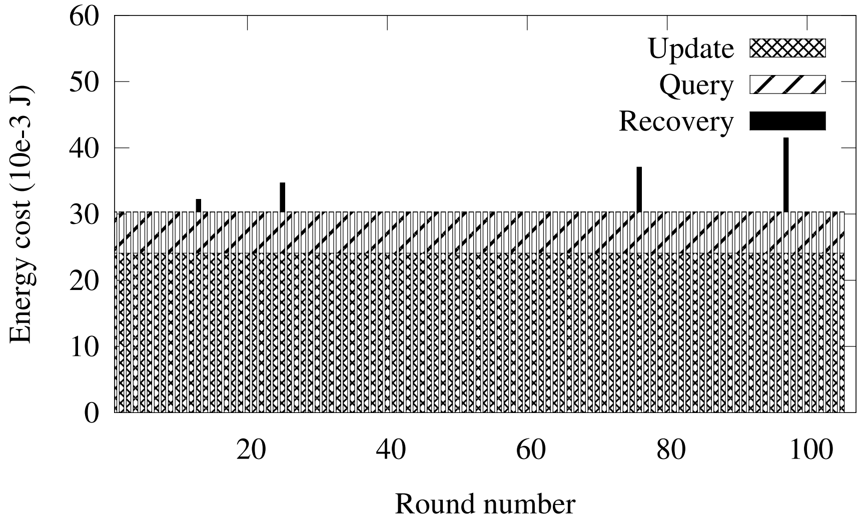
$$L_{resp2} = 64T \tag{5.8}$$

For simplicity, we neglect the fact that a node does not send the update messages to its neighbors or respond to the QueryMsg message after its failure.

We begin with the explanation of a specific simulation run. In one run, the center of the query area is located at coordinates [331, 359]. Five node failures take place in rounds 14, 26, 77, 98 and 107. The failure that takes place in round 107 is the first non-recoverable node failure. Thus, the network lifetime for no recovery scheme and the recovery scheme is 14 rounds and 107 rounds, respectively. E_{query} and E_{update} can be calculated according to Eq. 3.16 and 3.17. For the centralized recovery, only R_{data} nodes outside of the query area need to send the responses to the sink. In this run, the number of R_{data} nodes that reside outside of the query area is 4, 2, 3, 2 for rounds 14, 26, 77 and 98, respectively. For the localized recovery, note that in order to fulfill the recovery, all the R_{data} nodes need to send all their $SensedData$ to the R_{parity} node. The number of R_{data} nodes for rounds 14,



(a)



(b)

Figure 5.10: Energy cost composition in each round for (a) the centralized recovery approach and (b) the localized recovery approach

26, 77 and 98 is 15, 19, 10, 13, respectively. Fig. 5.10 (a) and (b) shows the energy cost composition for the centralized recovery and localized recovery approach, respectively, for this specific simulation run.

From Fig. 5.10 (a) and (b), we can see that in the beginning, $E_{recovery}$ only accounts for a small proportion of the overall energy cost. As the round number increases, L_{resp2} increases accordingly, resulting in an increase of $E_{recovery}$. The recovery scheme regularly pays a cost E_{update} to keep redundant information within the network for the recovery. E_{update} can be expensive, which is three times more

Run number	Total energy cost of E_{query} (mJ)	Total energy cost of E_{update} (mJ)	Ratio
1	669.07	2577.41	3.85
2	704.85	2625.59	3.73
3	1296.93	5082.56	3.92
4	1259.11	4432.19	3.52
5	667.57	2625.59	3.93
6	1309.04	4673.07	3.57
7	577.75	2240.18	3.88
8	652.66	2770.12	4.24
9	933.20	3275.96	3.51
10	923.96	3589.11	3.88
11	576.16	2384.71	4.14
12	575.18	2553.32	4.44
13	532.63	1878.86	3.53
14	462.82	1686.16	3.64
15	1384.88	5154.83	3.72
16	1239.21	4913.95	3.97
17	940.51	3878.16	4.12
18	987.58	3781.81	3.83
19	721.74	3107.35	4.31
20	768.48	3227.79	4.20
Average	859.17	3322.93	3.87

Table 5.4: Total energy cost for the query and the update in different simulation runs

than E_{query} in this case. However, without the proposed scheme, there is data loss as early as round 14.

To make the results more generic, we conduct 20 simulation runs to compare the total energy consumption of E_{update} and E_{query}. $E_{recovery}$ is ignored because the other two categories are two orders of magnitude greater than $E_{recovery}$. We sum up the energy cost of each category till the network lifetime of the proposed scheme ends. The results are shown in Table 5.4. As shown in this table, the energy cost for updating information is 3.87 times as expensive as the energy cost for the query on average. The communication overhead is an inevitable drawback of the proposed scheme. For mission critical WSN applications, we argue that it is still worthwhile to invest in the extra message transmissions to achieve higher data availability.

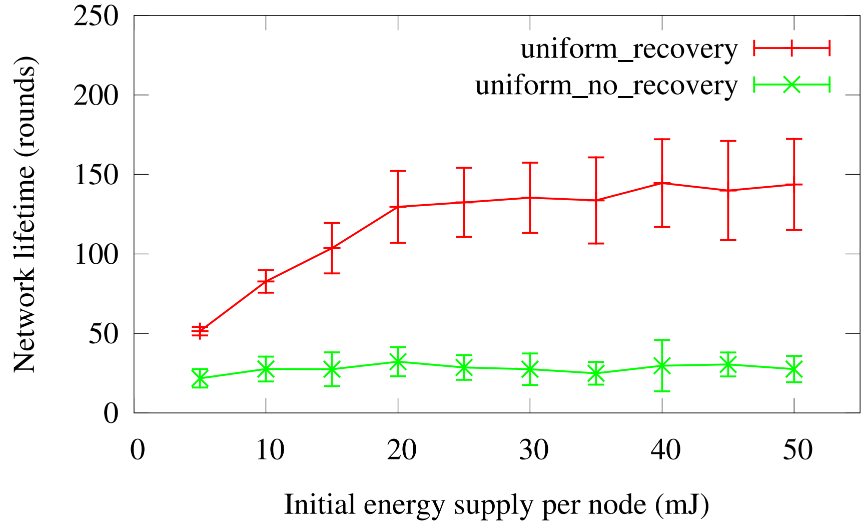
5.6 Initial energy supply

As discussed in the previous section, the communication overhead caused by the proposed recovery scheme is not trivial compared to the query cost. Given the same amount of initial energy, the network which adopts the recovery scheme will run out of energy faster than the network with no recovery scheme involved. In sections from 5.1 to 5.5, we did not take any energy constraint into account. This implies that there is infinite initial energy supply for sensors in those experiments. As one may argue, infinite energy supply is not realistic. In this section, we aim to investigate how different initial energy supply levels affect the performance of the recovery scheme and no recovery scheme. The parameters, p, W, N, and Q are set to be their default values. Under such parameters, we set the range of initial energy supply per node to be [5mJ, 50mJ]. This setting is pessimistic in the sense that batteries would typically have a much larger charge[1] making the energy overhead of our approach be even less noticeable when comparable to the inherent probability of failure. Now the reason for a node failure is two folds: either due to energy depletion or due to the node failure probability p. The network lifetime achieved for uniform node distribution and grid node distribution is presented in Fig. 5.11 (a) and (b), respectively.

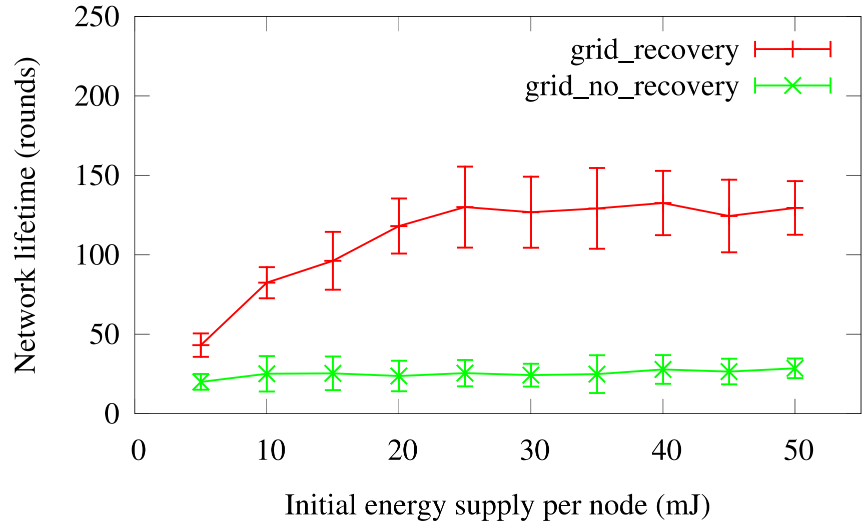
For both node distributions, different levels of initial energy supply do not affect the network lifetime for the no recovery scheme. The moment that the first node failure takes place, which is considered as the termination of network lifetime for the no recovery scheme, the sensors have not suffered from energy depletion in most cases, even when the initial energy supply per node is as low as 5mJ. Thus, the node failure resulting from energy depletion can barely affect the network lifetime for the no recovery scheme. The trend of network lifetime is therefore the same as if there were no energy constraint.

For the recovery scheme, when the initial energy supply varies in the range of [5mJ, 20mJ], it can have a large impact on the network lifetime. Decreasing the initial energy supply can shorten the network lifetime significantly. In this range,

[1]http://www.allaboutbatteries.com/Energy-tables.html



(a)



(b)

Figure 5.11: Network lifetime with various initial energy supply for (a) uniform node distribution and (b) grid node distribution

sensors typically run out of energy before the first node failure caused by p takes place, resulting in an early termination of network lifetime. Therefore, the network lifetime can not achieve as high values as the cases when there is infinite energy supply. When sensors are charged with more than 20mJ initial energy, the energy can usually last longer than the first non-recoverable node failure caused by p. Thus, varying the initial energy supply in the range above 20mJ will have very limited impact on the network lifetime.

The performance gain ratio as a function of the initial energy supply is shown in Fig. 5.12. The trend of both node distributions is very similar. Within the range

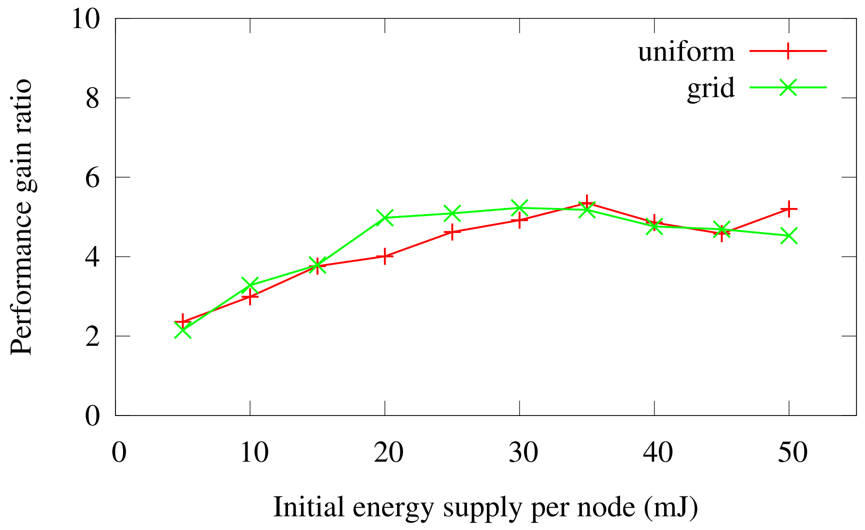


Figure 5.12: Performance gain ratio with various initial energy supply

[5mJ, 20mJ], G achieves rather small values and increasing the initial energy supply leads to the increase of G. After 20mJ, G remains quite stable no matter how the initial energy supply varies. This result indicates that if the initial energy supply is very limited, the performance gain is not as substantial as the cases with unlimited energy supply.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In recent years, the proliferation of various wireless sensor network applications resulted in an increasing research interest in WSNs. Sensor nodes come along with limited energy supply due to their small sizes and the cost-effectiveness consideration. When the energy depletes, a node inevitably fails. The fact that sensors are often deployed in harsh and unattended areas renders them more vulnerable to physical damage resulting from the environment. Moreover, recharging or replacing the failed nodes is not feasible in most cases. Usually, the sink issues queries for gathering the sensed values from the sensor nodes. When a node fails, the query responses from the sensor nodes will be incomplete, which may mislead the user. Without any fault recovery mechanism, the user can not have access to the sensed data of the failed node ever since it failed. We argue that the data stored in the failed node is valuable. Therefore, it is desirable for a WSN to have some fault tolerance, more specifically, fault recovery ability so that the network can provide the most accurate information to the user in the presence of node failures.

In this thesis project, we proposed a fault recovery scheme to recover the data after node failures take place in WSNs. Our scheme is good for mission critical applications where the sensed values of each sensor node matter and the system designer is willing to pay for some communication overhead in exchange for higher data availability. We designed the recovery scheme in a generic way so that it can be integrated with other WSN protocols/algorithms. Our work differs from previous research efforts which isolate the failed nodes in the communication layer and ignore the data of the failed node.

We used the network lifetime to indicate the performance of the proposed scheme. However, in WSN research, there is no universal definition of network lifetime. Researchers use different metrics, e.g., number of alive nodes, coverage, and connectivity, to define the network lifetime. In our work, in order to measure to what extent the recovery scheme can tolerate node failures, we defined the network lifetime as the time interval from the point that a WSN starts operation to the point that a node failure is observed and can not be recovered.

As authors in [42] have pointed out, there is always a trade-off between the complexity of fault tolerance design and the resource consumption of sensor networks. By adding the fault recovery feature into WSNs, the proposed scheme causes extra communication overhead. In order to investigate the energy cost, an energy consumption model was presented in section 3.5. Using this model, the communication cost of querying, updating redundant information and the recovery process can be quantified.

We used the network simulator Sinalgo to test the proposed recovery scheme. We conducted a series of experiments to show the network lifetime gain of the proposed recovery scheme under different parameters: node failure probability, communication range, node density and query size. We also compared the network lifetime of the proposed scheme with no recovery scheme. Since the proposed scheme relies heavily on the neighborhood relations, the average number of neighbors N_{nb} is an important factor. The experimental results lead us to believe that for the recovery scheme, there is a turning point for N_{nb}, below which increasing N_{nb} will yield better lifetime performance and above which increasing N_{nb} will result in opposite effect. Even in the worst case scenario, the network lifetime of the proposed scheme can still achieve at least three times as long as the lifetime of no recovery scheme.

6.2 Future work

As shown in section 3.5, we have not implemented a fault tolerant multi-hop routing protocol between sensor nodes and the sink. Instead, we considered a multi-hop routing path as N_{hop} hops of direct communication between neighboring nodes along the path and calculated the sum of all the direct communication energy cost. This analytical analysis does not take some realistic aspects into account and can only give an estimation of the energy consumption. For example, as the number of failed nodes increases, the average number of hops needed for transmitting a data packet back to the sink should increase accordingly, since the optimal path may not be available due to node failures along the path. Thus, a fault tolerant multi-hop routing implementation is needed so that we can sum up the energy cost of each sensor node on a routing path to get the exact energy cost for this routing path.

Our work assumes that all the message transmissions are reliable. In reality, messages can be dropped due to the interference caused by either environmental noise or message collisions at sensor nodes. Thus, we need a mechanism to ensure the transmission reliability. Erasure coding might be an option to achieve such goal.

We also assume that the topology of the network is flat. However, it is worth investigating how the network lifetime and the energy consumption will be when the network topology is hierarchical.

Bibliography

[1] Jamal N. Al-Karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.

[2] Hind Alwan and Anjali Agarwal. A survey on fault tolerant routing techniques in wireless sensor networks. *2009 Third International Conference on Sensor Technologies and Applications*, pages 366–371, June 2009.

[3] Muhammad Asim, Hala Mokhtar, and Madjid Merabti. A self-managing fault management mechanism for wireless sensor networks. *International Journal of Wireless Mobile Networks*, 2(4):184–197, 2010.

[4] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 322–331, 1990.

[5] Manish Bhardwaj. Power-aware systems. *Master of Science thesis, Massachusetts Institute of Technology*, 2001.

[6] Douglas M. Blough and Paolo Santi. Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 183–192, 2002.

[7] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Computer Communication Review*, 28(4):56–67, October 1998.

[8] Stefano Chessa and Piero Maestrini. Fault recovery mechanism in single-hop sensor networks. *Computer Communications*, 28(17):1877–1886, October 2005.

[9] Alexandru Coma, Jörg Sander, and Mario A. Nascimento. An analysis of spatio-temporal query processing in sensor networks. In *Proceedings of the 21st International Conference on Data Engineering*, pages 1190–1195, 2005.

[10] Jing Deng, Richard Han, and Shivakant Mishra. A robust and light-weight routing mechanism for wireless sensor networks. In *Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, 2004.

[11] Isabel Dietrich and Falko Dressler. On the lifetime of wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(1):1–39, February 2009.

[12] Wan Du, Fabien Mieyeville, and David Navarro. Modeling energy consumption of wireless sensor networks by SystemC. In *2010 Fifth International Conference on Systems and Networks Communications (ICSNC)*, pages 94–98, 2010.

[13] Stefan Dulman, Tim Nieberg, Jian Wu, and Paul Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In *Wireless Communications and Networking*, volume 3, pages 1918–1922, 2003.

[14] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25, October 2001.

[15] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, New York, NY, USA, 1984. ACM.

[16] Wendi Rabiner Heinzelman. Application-specific protocol architectures for wireless networks. *PhD thesis, Massachusetts Institute of Technology*, 2000.

[17] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, volume 8, 2000.

[18] Mohammad Ilyas, Imad Mahgoub, and Laurie Kelly. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[19] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67, New York, NY, USA, 2000. ACM.

[20] Chris Karlof, Chris Karlof, Yaping Li, Yaping Li, Joseph Polastre, and Joseph Polastre. ARRIVE: Algorithm for Robust Routing in Volatile Environments. Technical report, 2002.

[21] Santosh Kumar, Ten H. Lai, and József Balogh. On k-coverage in a mostly sleeping sensor network. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 144–158, New York, NY, USA, 2004. ACM.

[22] Weifa Liang and Yuzhen Liu. Online data gathering for maximizing network lifetime in sensor networks. *Mobile Computing, IEEE Transactions on*, 6(1):2–11, 2007.

[23] Hai Liu, Amiya Nayak, and Ivan Stojmenovic. Fault-tolerant algorithms /protocols in wireless sensor networks. In *Guide to Wireless Sensor Networks*, Computer Communications and Networks, pages 261–291. Springer London, 2009.

[24] Ritesh Madan, Shuguang Cui, Sanjay Lall, and Andrea Goldsmith. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1964–1975, 2005.

[25] Diana Marculescu, Nicholas H. Zamora, Phillip Stanley-Marbell, and Radu Marculescu. Fault-tolerant techniques for ambient intelligent distributed systems. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, pages 348–355, Washington, DC, USA, 2003. IEEE Computer Society.

[26] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265, New York, NY, USA, 2000. ACM.

[27] Lilia Paradis and Qi Han. A survey of fault management in wireless sensor networks. *Journal of Network and Systems Management*, 15(2):171–190, June 2007.

[28] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pages 109–116, New York, NY, USA, 1988. ACM.

[29] Theodore S. Rappaport. *Wireless communications - principles and practice*. Prentice Hall, 1996.

[30] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM Computer Communication Review*, 27(2):24–36, April 1997.

[31] Iman Saleh, Adnan Agbaria, and Mohamed Eltoweissy. In-network fault tolerance in networked sensor systems. In *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 47–54, New York, NY, USA, 2006. ACM.

[32] Iman Saleh, Hesham El-Sayed, and Mohamed Eltoweissy. A fault tolerance management framework for wireless sensor networks. In *Innovations in Information Technology*, pages 1–5, 2006.

[33] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, VLDB '87, pages 507–518, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[34] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 188–200, New York, NY, USA, 2004. ACM.

[35] Chao Song, Ming Liu, Jiannong Cao, Yuan Zheng, Haigang Gong, and Guihai Chen. Maximizing network lifetime based on transmission range adjustment in wireless sensor networks. *Computer Communications*, 32(11):1316 – 1325, 2009.

[36] Jessica Staddon, Dirk Balfanz, and Glenn Durfee. Efficient tracing of failed nodes in sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 122–130, New York, NY, USA, 2002. ACM.

[37] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[38] Gayathri Venkataraman, Sabu Emmanuel, and Srikanthan Thambipillai. Energy-efficient cluster-based scheme for failure management in sensor networks. *Communications, IET*, 2(4):528–537, 2008.

[39] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 28–39, New York, NY, USA, 2003. ACM.

[40] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. Erasure-coding based routing for opportunistic networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 229–236, New York, NY, USA, 2005. ACM.

[41] Hakim Weatherspoon and John Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 328–338, London, UK, UK, 2002. Springer-Verlag.

[42] Mengjie Yu, Hala Mokhtar, and Madjid Merabti. Fault management in wireless sensor networks. *Wireless Communications, IEEE*, 14(6):13–19, 2007.

[43] Haibo Zhang and Hong Shen. Balancing energy consumption to maximize network lifetime in data-gathering sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 20(10):1526–1539, 2009.

[44] Honghai Zhang and Jennifer C. Hou. Maximizing α-lifetime for wireless sensor networks. *International Journal of Sensor Networks*, 1(1/2):64–71, September 2006.

[45] Haiying Zhou, Danyan Luo, Yan Gao, and Decheng Zuo. Modeling of node energy consumption for wireless sensor networks. *Wireless Sensor Network*, 3(1):18–23, 2011.