RAID'ing Wireless Sensor Networks
Data Recovery for Node Failures

Ailin Zhou and Mario A. Nascimento
Department of Computing Science, University of Alberta, Edmonton, Canada

Keywords: Fault Recovery, Wireless Sensor Networks, Useful Network Lifetime.

Abstract: In wireless sensor networks (WSNs), sensor nodes may fail due to energy depletion or physical damage. To avoid data loss and incomplete query results when node failure takes place, we propose a node failure recovery scheme which can recover the data of failed nodes. Our scheme incorporates data redundancy and distributes, in an effective and storage efficient manner, redundant information among nodes in the WSN. When a node fails, the remaining functioning sensors can use the redundant information regarding the failed node to recover its data. An energy consumption model is also presented for calculating the communication cost of the proposed scheme. We use simulations to compare the network lifetime with and without recovery being involved, where the network lifetime is defined as the time that a node failure is observed and its data can not be recovered. Our experimental results show that the recovery scheme can yield a lifetime up to three times longer than that of no-recovery scheme.

1 INTRODUCTION

A wireless sensor network (WSN) is made up of large numbers of sensor nodes that use wireless communication protocol to communicate. The sensor nodes can collect, process and exchange sensed data in a large area to achieve various goals autonomously. The collected data is typically forwarded to sink node(s) for further data analysis.

Sensor nodes come with limited energy supply and when it is depleted the node simply stops working. The fact that sensors are often deployed in harsh and unattended areas renders them vulnerable to physical damage. 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.Therefore, it is desirable for a WSN to have some fault recovery ability so that the network can provide accurate information to the user in the presence of node failures.

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. RAID (Redundant Array of Independent Disks) is an original storage technique that distributes data and utilizes redundant disks for data recovery (Patterson et al., 1988). Similar to the idea of RAID, we propose to let each sensor node store some redundant information about all its direct neighbors. 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.

Our scheme is different from previous works in WSN research. We do not aim at minimizing the energy consumption of data dissemination or tolerating link failure. Instead, we focus on the data management level, in particular a RAID-like technique, to achieve fault tolerance at the cost of additional data communication. To the best of our knowledge, this is the first attempt to achieve in-network *node* failure recovery for wireless sensor networks.

The remainder of this paper is organized as follows: Section 2 reviews recent research on fault tolerance in WSNs. Section 3 introduces our proposed data recovery scheme, as well as the energy consumption model we adopt. The analysis of the experimental results is presented in Section 4. A brief conclusion and possible future work directions are discussed in the final section.

2 RELATED WORK

Many researchers have focused on designing fault tolerant routing protocols. According to a survey (Al-

Zhou A. and A. Nascimento M..
RAID'ing Wireless Sensor Networks - Data Recovery for Node Failures.
DOI: 10.5220/0004672702980308
In *Proceedings of the 3rd International Conference on Sensor Networks* (SENSORNETS-2014), pages 298-308
ISBN: 978-989-758-001-7

wan and Agarwal, 2009), fault tolerant routing techniques can be classified into two main categories: retransmission and replication. Retransmission is quite popular since the packet loss rate in WSNs is higher than in traditional networks. Two popular replication mechanisms are multipath routing (Karlof et al., 2003; Ganesan et al., 2001) and erasure coding (Wang et al., 2005). 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. 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, which 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.

Most of the existing 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 (Marti et al., 2000). In these papers, fault tolerance is achieved in the 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 with these approaches.

Similar to our goal, Chessa and Maestrini (Chessa and Maestrini, 2005) present a fault recovery mechanism to cope with node failures in single hop WSNs. 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 not realistic *single hop* WSNs, whereas our work can be applied to multi-hop WSNs and can cope with multiple node failures at the same time.

3 PROPOSED NODE FAILURE RECOVERY SCHEME

To investigate the performance gain and the energy consumption overhead under a generic network topology setting, the first assumption we made is that the topology of the network is flat, i.e., that there are no 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. The sink is considered as being constantly charged by a reliable energy source. We assume that sensor nodes are stationary after being deployed, each node operates within a fixed radio range and, while nodes themselves can fail, links between nodes are reliable. Finally, since our approach depends on location of sensors and neighborhood relationships between sensors, we assume that the sink stores the whole network topology and each sensor node has a list of all the neighbors that reside within its communication range; this can be achieved using inexpensive GPS modules at deployment time.

3.1 System Model

We consider a WSN composed of large numbers of sensor nodes with one single sink located at the center of the deployment field, however, this can be easily generalized to other cases. The sensor nodes do continuous and periodic sensing and data collection at their locations. A sensor node can be either in an alive or a failed state. The transition from an alive state to a failed state is one-way and irreversible. Our premise is that when one node fails, the remaining alive sensor nodes should be 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 no node failure.

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[1]. Upon receiving the query message, each node determines whether it is within the query area and will respond (or not) to this query.

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

[1]There are protocols more efficient than simple flooding, but their usage is orthogonal to our purposes in this this paper.

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.

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 (Patterson et al., 1988), each sensor node in our scheme now serves as the dedicated parity disk for the array composed of all its direct neighbors. (For a brief discussion on how RAID 4 works, please refer to the Appendix).

The parity in our context is the result of bitwise XOR operation (denoted as $\oplus$) of all the binary inputs. If there is more than one boolean input, 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 Figure 1, we can define the following relations for nodes with more than one neighbor: $P_1 = D_3 \oplus D_4$, $P_3 = D_1 \oplus D_2$ and $P_4 = D_1 \oplus D_5 \oplus D_6$.

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. For example, assume that node 1 fails and we need to reconstruct D_1. We can do so by using either of the two following relations: $D_1 = P_3 \oplus D_2$ or $D_1 = P_4 \oplus D_5 \oplus D_6$.

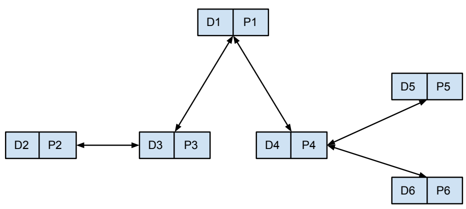


Figure 1: Storage unit partition in sensor nodes.

3.2 Recovery Initialization

Prior to proceeding with the recovery, the sink needs to figure out when a node failure takes place. Our experimental setup ensures that in every round, each alive node within the query area generates a response message and the response message is assumed to arrive at the sink in the next round. Thus, we treat the time when we observe a missing response as the time when the corresponding node failure happens.

When the failure of a node F has been observed, the sink initiates the recovery process immediately. The main goal of this process is to find the best recovery candidate, denoted as R_{parity}, whose *ParityData* will be utilized to fulfill the recovery.

First we establish R', the set of valid recovery candidate nodes. A node is a valid recovery candidate if it is a neighbor of F, all of its neighbors except F are alive and it is alive at the time the recovery is initialized. Then the list R' is traversed and the node with smallest degree is chosen and denoted as R_{parity}. 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 direct neighbors of R_{parity}, 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.

3.3 Centralized and Localized Recovery

After the sink initiates the recovery and chooses R_{parity}, the actual recovery process can take place either at the sink or in R_{parity}. If the process takes place at the sink, we name it centralized recovery. If it takes place in R_{parity}, we name it localized recovery. For centralized recovery, all the R_{parity} nodes and R_{data} nodes send their responses back to the sink while for localized recovery, the sink 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 received its query responses and stored its *SensedData*. Otherwise, the sink needs to send request messages to R_{data} nodes asking for the *SensedData*. After receiving all necessary responses, the sink completes the recovery calculation (as described in Section 3.1).
- For localized recovery, the R_{parity} node will be notified by the sink that it has been chosen as the best recovery candidate. Then R_{parity} sends request

messages to all R_{data} nodes and waits for their responses. The recovery calculation takes place in the R_{parity} node and then the R_{parity} node can respond to queries on behalf of the failed node.

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. When processing aggregate queries, sensors summarize (aggregate) their data in order to reduce the volume of data that needs to be transmitted to the sink. 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 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}. Then the whole recovery process is completed in the R_{parity} node so that no raw data transmission to the sink is required.

3.4 Modeling Energy Consumption

Energy constraint is one of the most fundamental challenges in WSN research. Researchers have come up with many energy models, e.g., (Du et al., 2010; Shnayder et al., 2004), that are used to evaluate network lifetime or compare different algorithms or protocols in network design and analysis. In general, energy cost has three components: sensing, processing and communication. The energy cost for sensing and processing are often considered as constant values, hence we focus on the cost for data communication only. In (Rappaport, 1996) the author presents a model for calculating the transmitting and receiving energy cost for a single bit as $E_{tx} = \alpha_{11} + \alpha_2 \times d^n$ and $E_{rx} = \alpha_{12}$, respectively, where α_{11} and α_{12} indicate the energy cost required 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, and n is the path loss exponent. Table 1 lists the notations used in the model.

Since we assume that the sink has unlimited power supply, our energy consumption model does not take the energy cost of the sink into consideration. In the proposed scheme, the energy cost of sensor nodes in every simulation round falls into the following components: (1) receiving query messages and transmitting query responses, denoted as E_{que}, (2) sending and receiving sensed value updates to/from neighbors, denoted as E_{upd} or (3) receiving recovery request messages and transmitting the corresponding responses, denoted as E_{rec}

Since we assume that the sensor nodes always transmit messages using a fixed transmission power, $E_{tx_{node}}$ can be calculated using the communication range $W = 60$ as the distance: $E_{tx_{node}} = \alpha_{11} + \alpha_2 \times W^2 = 86\ nJ$. The deployment field is in the size of $500\ m \times 500\ m$. In our simulations we use the following values as provided in (Heinzelman, 2000): $\alpha_{11} = \alpha_{12} = 50\ nJ/bit$, $n = 2$, $\alpha_2 = 10\ pJ/bit/m^2$.

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 as: $N_{nb} = \frac{N\pi W^2}{XY} - 1$.

To estimate the energy cost of a multi-hop routing path, one needs to know how many hops on average, denoted as N_{hop}, are needed to send the data from the source node to the sink. 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}. Thus, we have $N_{hop} = \frac{D_{sink}}{D_{proj}}$. An example of the orthogonal projection is shown in Figure 2. 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 projected onto the direction of D_{sink}.

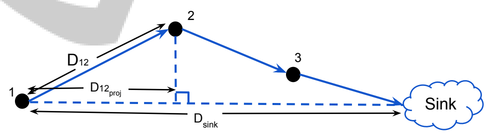


Figure 2: Projection of a direct communication path on the direction towards the sink.

For large size WSNs, the distance from R_{data} nodes and R_{parity} nodes within the query area 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 is defined as $D_{sink} = \sqrt{(S_x - Q_x)^2 + (S_y - Q_y)^2}$. In (Coman et al., 2005), D_{proj} is given as $D_{proj} = \frac{2W}{3} \cos \frac{\pi}{2N_{nb}}$. Thus, $E_{tx_{sink}}$ can be calculated as the total energy cost of N_{hop} hops of direct communication: $E_{tx_{sink}} = N_{hop} \times (E_{tx_{node}} + E_{rx}) - E_{rx}$. The $(-E_{rx})$ parcel is due to the fact that the source node does not need to receive any message.

Denoting the round number as T, the size of the response messages to the recovery request in the T^{th} round is: $L_{resp2} = L_{resp1} \times T$

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

Table 1: Notations used in the energy consumption model.

Nota.	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	(c.f., Table 2)
Q	The queried are (percentage) of the deployment field	(c.f., Table 2)
N_{in}	Number of nodes within the query area	N/Q
N_{nb}	Average number of neighbors per node	$\frac{N\pi W^2}{XY} - 1$
N_{hop}	Average number of hops from sensors to sink	$\frac{D_{sink}}{D_{proj}}$
N_{data}	Number of R_{data} nodes in a recovery process	related to R_{parity}
N_{out}	Number of R_{data} nodes outside the query area	related to R_{parity}
W	Communication range	60 m
$E_{tx_{node}}$	Energy used to transmit a bit to a sensor	86 nJ
$E_{tx_{sink}}$	Energy used to transmit a bit to the sink	$N_{hop} \times (E_{tx_{node}} + E_{rx}) - E_{rx}$
E_{rx}	Energy used to receive a bit	50 nJ
L_{que}	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	$L_{resp1} \times T$
L_{upd}	Size of the sensed value update message	64 bits
L_{req}	Size of the recovery request message	64 bits

pay the price of receiving the query message and only nodes within the query area (N_{in}) will respond to the sink. Therefore, we have: $E_{que} = N \times E_{rx} \times L_{que} + N_{in} \times E_{tx_{sink}} \times L_{resp1}$.

For E_{upd}, 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 $E_{tx_{node}} \times L_{upd}$. At the same time, one node receives N_{nb} update messages which have been sent from its neighbors in the previous round, thus: $E_{upd} = N \times E_{tx_{node}} \times L_{upd} + N \times N_{nb} \times E_{rx} \times L_{upd}$

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: $E_{rec} = E_{data} + E_{parity}$.

As discussed in Section 3.3, there are two different recovery approaches. For the centralized approach, the R_{parity} node and R_{data} nodes that reside outside of the query area receive the request messages from the sink and send the responses back to the sink. Thus, for this approach we have: $E_{data} = N_{out} \times (E_{rx} \times L_{req} + E_{tx_{sink}} \times L_{resp2})$ and $E_{parity} = E_{rx} \times L_{req} + E_{tx_{sink}} \times L_{resp2}$. For the localized approach, all R_{data} nodes receive from and respond to the chosen R_{parity}, yielding: $E_{data} = N_{data} \times (E_{rx} \times L_{req} + E_{tx_{node}} \times L_{resp2})$. 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, thus $E_{parity} = E_{rx} \times L_{req} + E_{tx_{node}} \times L_{req} + N_{data} \times E_{rx} \times L_{resp2}$.

3.5 Network Lifetime

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, e.g., (Liang and Liu, 2007; Zhang and Shen, 2009). As the design of WSN depends heavily on the specific application requirements, the definition and metrics for estimating network lifetime is also application specific.

Coverage and connectivity have both been used for evaluating WSNs (Dietrich and Dressler, 2009). 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.

By taking the specific goal of our scheme into account, we have the following definition: *The network lifetime is defined as the time interval from the point that a WSN starts operation up 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. Therefore failures (and the underlying recovery process) are transparent to the end users.

Table 2: Notations used in the experiments.

Notation	Description	Values
L	The number of rounds until the first node failure cannot be recovered	(result of simulation)
P_f	The probability that at least one node fails within the query area in each round	$1-(1-p)^{N_{in}}$
P_v	The probability that a recovery candidate is considered as valid	$(1-p)^{N_{nb}}$
p	The probability that one sensor fails in any 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%

4 EXPERIMENTAL RESULTS

In our experiments, we use the network simulator Sinalgo[2]. Sinalgo differs from other network simulators such as OMNeT++ and NS-3 in the 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. Since Sinalgo adopts the concept of "rounds" to achieve clock synchronization within the network, we use the number of rounds to measure network lifetime (as defined above).

We are interested in to what extent the proposed scheme can extend the network lifetime under different scenarios and network settings. 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 3.4 to investigate the energy cost for each round. Due to the lack of space we show only results regarding our centralized recovery approach, which is a worst case scenario in terms of energy overhead; recall that in the localized approach the exchange of messages is more constrained to the vicinity of the failed node.

We use a uniform distribution model and a grid distribution model for the initial placement of sensors 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 the following each point in the graphs is the average of 20 different simulation runs using distinct seeds. Since the network lifetime is a random variable, a 95% confidence interval is included to give an estimate of the true network lifetime. The notation used in this section is listed in Table 2.

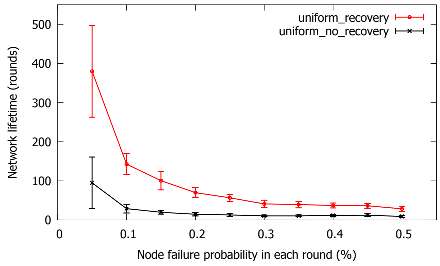
4.1 Node Failure Probability

Since the motivation of proposing a fault recovery 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 fails in one single round. 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.* 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.

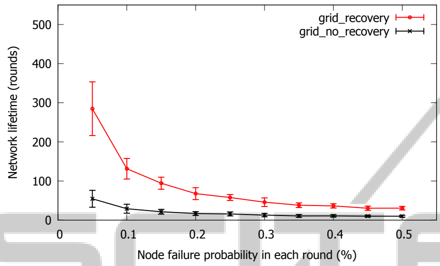
The network lifetime of uniform distribution and grid distribution is shown in Figures 3(a) and (b). As the node failure probability increases, the network lifetime of both schemes drops as one would expect.

For no-recovery scheme, the probability that at least one node failure takes place within the query area during one single round, P_f, is given by: $P_f = 1-(1-p)^{N_{in}}$. For this experiment, N_{in} is fixed since neither the query size nor the network density changes. When we increase p in the range of (0, 1), the resulting P_f increases monotonically. The number of rounds until the first node failure occurs, L, is a geometrically distributed random variable, with expected value given by: $E[L] = \frac{1}{P_f}$. Therefore, when P_f increases, $E[L]$ decreases, which indicates that the first node failure is expected to happen earlier. Thus,

[2]http://disco.ethz.ch/projects/sinalgo/index.html



(a) Uniform Sistribution



(b) Grid Sistribution.

Figure 3: Network lifetime vs. p.

we observe a shorter network lifetime for no-recovery scheme when the node failure probability increases.

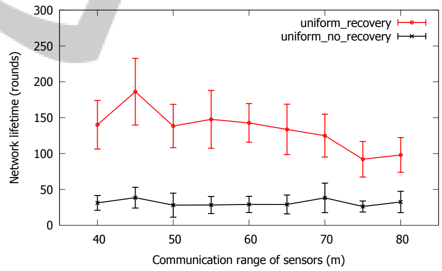
For the recovery scheme, the network lifetime depends on when a failed node can no longer be recovered. The recovery algorithm as defined in Section 3.2 builds a recovery candidate set R and a valid candidate set R'. p has no effect on the composition of R. For a specific recovery candidate C_i in R, 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 is: $P_v = (1-p)^{N_{nb}}$. When p increases, P_v decreases monotonically, which means that each recovery candidate has less chance of being considered a valid candidate. As a consequence, R' has a higher chance to be an empty set. Therefore, the network lifetime of the recovery scheme is also expected to decrease as the node failure probability increases.

4.2 Communication Range

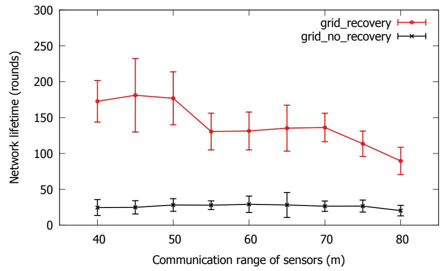
Controlling W directly affects the number of neighbors a node can communicate with. Since our proposed recovery scheme depends on the neighborhood relations, it is worthwhile investigating how the network lifetime gain will be influenced by W. The first problem is how to set a reasonable range of W to 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 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 and all other investigated parameters are set using their default values.

As shown in the formula for N_{nb}, for a randomly and uniformly distributed WSN, N_{nb} is proportional to W^2 given that N and the dimension of deployment field are fixed. For grid distribution though, the average number of neighbors does not grow linearly with W^2 because sometimes increasing W does not necessarily create more edges between nodes.

Now both the size of R and the size of 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 P_v. As N_{nb} increases, P_v 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. The total effect of a larger recovery candidate set R and a lower P_v for each recovery candidate depends on which factor plays a dominant role.



(a) Uniform Distribution



(b) Grid Distribution.

Figure 4: Network lifetime vs. W.

The network lifetime for uniform and grid distribution is shown in Figures 4(a) and (b), respectively. From Figure 4(a) we can see that when W varies from 40 to 45 meters, the network lifetime of the recovery scheme increases by approximately 33%. 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_v. After this the network lifetime remains quite smooth from 50 to 70 meters. 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 26% performance drop takes place, indicating that the negative effect of a decreasing P_v 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. Thus the average number of neighbors should be carefully chosen.

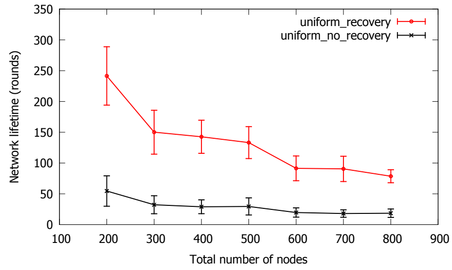
For grid distribution, the network lifetime of the recovery scheme has a similar trend as shown in uniform distribution. Note that $W = 40$ and $W = 45$ has the same connectivity since their N_{nb} is identical. The same case applies to $W = 55$, $W = 60$, and $W = 65$. Since the node distribution is fixed, the same connectivity implies the same network lifetime performance. When W is in the range between 40 and 50 meters, the network lifetime achieves the highest values. After that the network lifetime decreases by nearly 50 rounds when W increases from 50 to 55 meters. The corresponding N_{nb} increases but as previously discussed, the considerable increase in N_{nb} brings more recovery candidates that 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}.

The network lifetime of no-recovery scheme does not change significantly as W increases. As discussed in Section 4.1, $P_f = 1 - (1-p)^{N_{in}}$. In this experiment, both p and N_{in} are considered as fixed. Thus, increasing W will not affect the lifetime of no-recovery scheme.

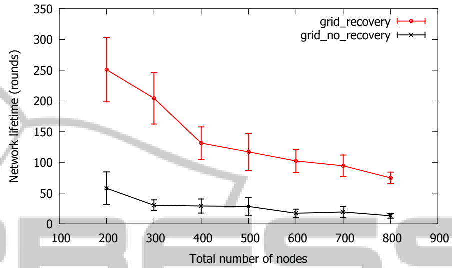
4.3 Node Density

In this experiment, p, W and Q are set to be their default values. We vary the total number of deployed nodes, N, from 200 to 800, with the increment being 100. The results for uniform distribution and grid distribution are shown in Figures 5(a) and (b), respectively.

As we can see that as N increases, the network lifetime of both no-recovery scheme and the recovery scheme decreases. For no-recovery scheme, this trend is anticipated. For uniform distribution in a given deployment field, the number of nodes that reside within



(a) Uniform Distribution



(b) Grid Distribution.

Figure 5: Network lifetime vs. N.

the query area, N_{in}, is proportional to the total number of nodes and the percentage of the deployment field covered by the query: $N_{in} = N \times Q$. For grid distribution, the previously indicated relations holds true approximately. As shown in the equation for P_f, when N_{in} increases, P_f increases monotonically, which results in a lower expected value of $E[L]$. Thus, the first node failure is expected to happen earlier when the network becomes denser.

For the recovery scheme, the average number of neighbors N_{nb} increases as the network gets denser. For uniform distribution, N_{nb} increases linearly with N and for grid distribution it increases monotonically with N.

Similar to the discussion in Section 4.2 , a larger N_{nb} can not guarantee a longer lifetime for the recovery scheme since each recovery candidate has a less likelihood to be valid. According to the equation for P_v, a larger N_{nb} yields a smaller P_v. However, the drop of P_v is not the only reason why the recovery candidates are less likely to be considered as valid. Note P_v 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 Figure 6.

Assume we are trying to recovery node 1. Node 1 has two recovery candidates: node 3 and node 4, which share the same neighboring node 7. Should

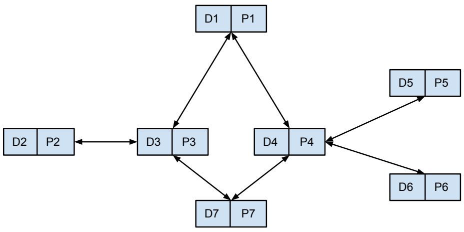
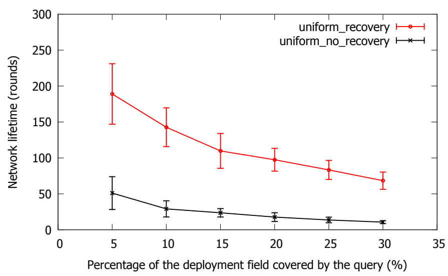


Figure 6: An example of common node shared by recovery candidates.

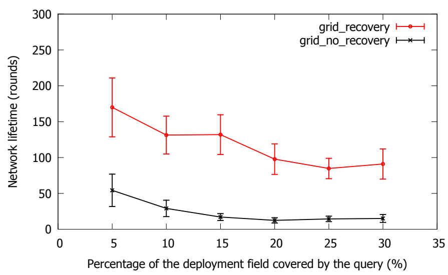
node 7 have failed, neither node 3 nor node 4 could be considered as a valid candidate for the recovery. The denser the network is, the more likely that this scenario can be observed. Thus, apart from the decrease of P_v, recovery candidates are less likely to meet the requirements of valid candidates as the network density increases.

4.4 Query Size

Since our definition of network lifetime is associated with observed node failures within a given query area, one may question whether the size of the query area matters to the outcome. In this experiment, we vary the percentage of the deployment field that is covered by the query from 5% to 30%. The other parameters are set to their default values.



(a) Uniform Distribution



(b) Grid Distribution.

Figure 7: Network lifetime vs. Q.

From Figure 7, we can see that the network lifetime of both recovery scheme and no-recovery scheme decreases as Q increases and the other parameters are kept constant. Clearly N_{in} increases as we enlarge Q. Therefore, the network lifetime of no-recovery scheme is expected to decrease since the first node failure is expected to come earlier, as per P_f and $E[L]$. For the recovery scheme, given this parameter setting where N_{nb} and p are fixed, the total number of recovery candidates and P_v 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 happen earlier and a shorter network lifetime is observed for the recovery scheme when Q increases.

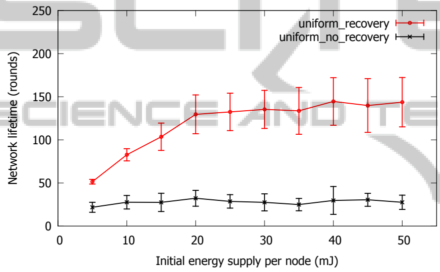
4.5 Communication Overhead

Our scheme clearly imposes an overhead in term of energy cost, due to the need to update a node's neighbor proactively, so that the node recovery can take place. In the experiments above, in order to isolate side-effects due to energy constraints and consequently better evaluate the pros and cons of our proposal in terms of data recoverability, we assumed that there was enough initial energy available that no node run out of energy during a simulation. We now drop this constraint.

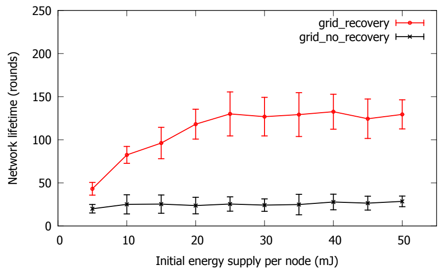
In order to investigate our proposal's energy overhead we compared the total energy cost of E_{upd} and E_{que}. E_{rec} was ignored since the other two components are typically one to two orders of magnitude greater than E_{rec} and p, W, N, Q were all set to be default values. We summed up the accumulative energy cost of E_{upd} and E_{que} till the network lifetime of the proposed scheme ends. We observed that the total E_{upd} was approximately 4 times as expensive as the total E_{que} on average. A legitimate question at this point is the following: given this energy cost overhead, does a WSN using our proposed scheme actually function for a longer time so that it is worthwhile, compared to the no-recovery scheme? The answer is affirmative.

Recall that we define end of WSN's lifetime as the point in time where it can no longer recover node from a failed node. That is, even though all non-failed nodes may still have energy the WSN is effectively not as useful as queries will report incorrect answers. In the following we will show that, given the same amount of initial energy supply to all nodes, in most cases a WSN *without* our recovery scheme will cease to be useful earlier than *any* node depletes its energy source when using our recovery scheme. In other words, the energy overhead is worth in the sense that it prolongs the useful lifetime of the WSN.

Let us now 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 were all set to be their default values. 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[3] making the energy overhead of our approach be even less noticeable when compared to the inherent probability of failure. The cause of a node failure is two-fold: either due to energy depletion or due to the node failure probability p. The network lifetime achieved for uniform distribution and grid distribution is presented in Figures 8(a) and (b), respectively.



(a) Uniform Distribution



(b) Grid Distribution.

Figure 8: Network lifetime vs initial energy budgets.

For both distributions, different levels of initial energy supply do not affect the network lifetime for our 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 yet not suffered from energy depletion in the vast majority of 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 no-recovery scheme. The trend of network lifetime is therefore the same as if there were no energy constraint.

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

For the recovery scheme, when the initial energy supply varies in the range of [5mJ, 20mJ], decreasing the initial energy supply can shorten the network lifetime. In this range, sensors typically run out of energy before the first node failure caused by p takes place, resulting in an early termination of the "useful" network lifetime. However, when sensors are initially charged with as little as 20mJ, 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, which is a quite reasonable assumption, will have very limited impact on the network lifetime.

5 CONCLUSIONS

We proposed a fault recovery scheme to recover the data after node failures take place in WSNs. Our scheme is suitable for WSN applications where the system designer is willing to pay for some communication overhead in exchange for higher data availability. Inspired by the RAID 4 technique for redundancy of hard disks, we designed the recovery scheme in a generic way so that it can be integrated with other WSN protocols/algorithms.

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. 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. In addition, we have shown that the communication overhead imposed by our approach pays off in the sense that a WSN using our proposed scheme will last longer (in terms of usefulness) than a "plain" WSN.

While out work focuses on recovering data due to node failures, it assumes that all the message transmissions are reliable. In fact, messages can be dropped due to the interference caused by either environmental noise or message collisions at sensor nodes. Thus, a mechanism to improve the transmission reliability that can be seamlessly integrated into our framework would be an interesting venue for further work. Erasure coding (Wang et al., 2005) may be an option to achieve such goal.

ACKNOWLEDGEMENTS

Research partially supported by NSERC Canada.

REFERENCES

Alwan, H. and Agarwal, A. (2009). A survey on fault tolerant routing techniques in wireless sensor networks. *Proc. of the 3rd Intl. Conf. on Sensor Technologies and Applications*, pages 366–371.

Chessa, S. and Maestrini, P. (2005). Fault recovery mechanism in single-hop sensor networks. *Computer Communications*, 28(17):1877–1886.

Coman, A., Sander, J., and Nascimento, M. A. (2005). An analysis of spatio-temporal query processing in sensor networks. In *Proc. of the 21st Intl. Conf. on Data Engineering (Workshops)*, pages 1190–1195.

Dietrich, I. and Dressler, F. (2009). On the lifetime of wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(1):1–39.

Du, W., Mieyeville, F., and Navarro, D. (2010). Modeling energy consumption of wireless sensor networks by SystemC. In *Proc. of the 5th Intl. Conf. on Systems and Networks Communications*, pages 94–98.

Ganesan, D. et al. (2001). Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25.

Heinzelman, W. R. (2000). Application-specific protocol architectures for wireless networks. *PhD thesis, Massachusetts Institute of Technology*.

Karlof, C., Li, Y., and Polastre, J. (2003). ARRIVE: Algorithm for Robust Routing in Volatile Environments. Technical report, University of California, Berkeley.

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

Marti, S. et al. (2000). Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of the 6th annual Intl. Conf. on Mobile computing and networking*, pages 255–265.

Patterson, D. A., Gibson, G., and Katz, R. H. (1988). A case for redundant arrays of inexpensive disks (RAID). In *Proc. of the 1988 ACM SIGMOD Intl. Conf. on Management of data*, pages 109–116.

Rappaport, T. S. (1996). *Wireless communications - principles and practice*. Prentice Hall.

Shnayder, V. et al. (2004). Simulating the power consumption of large-scale sensor network applications. In *Proc. of the 2nd Intl. Conf. on Embedded networked sensor systems*, pages 188–200.

Wang, Y. et al. (2005). Erasure-coding based routing for opportunistic networks. In *Proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 229–236.

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

APPENDIX

(Patterson et al., 1988) proposed the so-called RAID technique to solve the bottleneck issue of I/O performance in order to catch up with the increasing speed of CPUs and memories. 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. In our work, we choose the level RAID 4 to incorporate data redundancy and achieve fault recovery since it strikes a balance between space efficiency and data redundancy. This level can tolerate at most one physical disk failure. Figure 9 shows the diagram of a sample RAID 4 array.

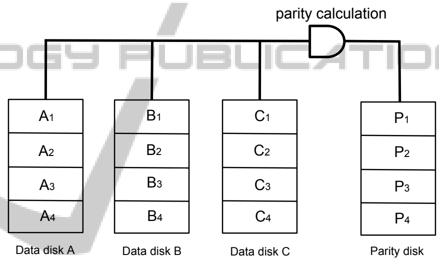


Figure 9: RAID 4 with three data disks and one parity disk.

RAID 4 is often referred to as a dedicated parity scheme since it utilizes a 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: $P_i = A_i \oplus B_i \oplus C_i$. 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 by computing $A_i = P_i \oplus B_i \oplus C_i$.