On Sensor Network Reconfiguration for Downtime-Free System Migrations

Yangfan Zhou
Dept. of Comp. Sci. & Eng.
The Chinese U. of Hong Kong
Shatin, Hong Kong, China
yfzhou@cse.cuhk.edu.hk

Michael R. Lyu
Dept. of Comp. Sci. & Eng.
The Chinese U. of Hong Kong
Shatin, Hong Kong, China
lyu@cse.cuhk.edu.hk

Jiangchuan Liu
School of Computing Sci.
Simon Fraser University
Burnaby, BC, Canada
jcliu@cs.sfu.ca

ABSTRACT

Many state-of-the-art wireless sensor networks have been equipped with reprogramming modules, *e.g.*, those for software/firmware updates. System migration tasks such as software reprogramming however will interrupt normal sensing and data reporting operations of a sensor node. Although such tasks are occasionally invoked, the long time of such tasks may disable the network from detecting critical events, posing a severe threat to many sensitive applications. In this paper, we present the first formal study on the problem of downtime-free migration. We demonstrate that the downtime can effectively be eliminated, by partitioning the sensors into subsets, and let them perform migration tasks successively with the rest still performing normal services. We then present a series of effective algorithms, and further extend our solution to a practical distributed and localized implementation, namely, the *Sensor Network Reconfiguration Protocol (SNRP)*. The performance of these algorithms have been evaluated through extensive simulations, and the results demonstrate that our algorithms achieve good balance between the sensing quality and system migration time.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed networks; C.2.1 [**Network Operations**]: Network management

General Terms

Algorithms, Design, Management

Keywords

Sensor System Migration, Sensor Network Reconfiguration, Network Partition

1. INTRODUCTION

A wireless sensor network (WSN) is usually employed to sense some physical data of interest so as to conduct environmental event detection [1]. There are many potential applications [2], *e.g.*, forest fire detection where a large amount of nodes equipped with thermoelectric and hygrometric sensors work cooperatively in raising a fire alarm, and borderline monitoring where many nodes equipped with infrared and acoustic sensors are deployed to conduct intruder detection.

In most application scenarios, a WSN is expected to work in an unattended manner for a long period of time once the sensor nodes have been deployed, because it is usually expensive or even impractical for human-attended operations on a sensor node (*e.g.*, especially for WSNs applied in battle-field or habitat monitoring). Yet the networks may have to occasionally perform certain *system migration tasks*. A system migration task is a process for reconfiguring, upgrading, or re-initializing existing network components or software/firmware, *e.g.*, reprogramming a sensing software-unit or re-initializing a communication protocol. Such tasks, however, are usually exclusive to the sensing operations; *i.e.*, a sensor node may have to cease its normal sensing and data reporting operations in performing these tasks.

There have been significant research efforts on implementing efficient online system migration for WSNs [3]. However, they have largely ignored the interruption caused by the system migration tasks. This can be a critical threat in many application scenarios. For example, if a WSN for fire or intruder detection is being reprogrammed, it may fail to detect and alarm a fire or an intrusion which happens during the process. Note that it typically takes a network with one hundred sensor nodes several hundred seconds to complete a reprogramming task [3], which is long enough to cause severe problems. Regular system migration tasks that occur periodically would further open this back-door for intruders to explore. Hence, it is very important to develop seamless system migration schemes that avoid the downtime of normal network operations so as to maintain the uninterrupted event detection functionality of the network.

A natural way for downtime-free system migration is to divide the network into several subsets and let the subsets perform the system migration task in turn, while the rest of the subsets still remain normal operations in sensing and processing environmental data. Obviously, the more the number of subsets is, on one hand, the longer the time is to finish the system migration task for the whole networks; on the other hand, the less the performance degrades during the system migration. The number of the subsets thus can serve as a flexible parameter to fine-tune the system migration process. Given this number, the critical problem then becomes how the sensor nodes are partitioned into subsets so as to achieve the best trade-off between system migration time and performance degradation.

Although a lot of efforts have been made on various sensor group-

QShine'08 July 28-31, 2008, Hong Kong, China

ing problems (*e.g.*, work in [4][5][6] and our recent work in [7]), their objectives are generally to maximize the number of the subsets while maintaining the performance of each subset. This is quite different from the problem context here, and the conventional algorithms thus cannot be applied. In this paper, we for the first time formulate the *sensor network reconfiguration problem* for downtime-free system migration. We prove that the problem is NP-hard with a general probabilistic sensing model. We then present a series of heuristics, which are further extended to a distributed and localized implementation. The performance of these algorithms have been evaluated through extensive simulations, and the results demonstrate that our algorithms achieve satisfactory balance between the sensing quality and system migration time.

The rest of the paper is organized as follows. In Section 2, we provide a formal description of this problem and justify its formulation. We then analyze this problem and prove it NP-Hard. Section 3 provides several algorithms in attacking this problem. The performance of these algorithms is studied in Section 4. We further point out some future work and conclude this paper in Section 5.

2. MODELS AND FORMULATIONS

2.1 Preliminaries

Suppose there are n sensor nodes in a WSN, denoted by $\{s_i\}_{i=1}^n$. The nodes can be *on* (*i.e.*, conducting event detection work) or be *off* (*i.e.*, during a system migration task such as being reprogrammed). c_i is used to denote the status of s_i. It is 1 if s_i is on, and 0 otherwise.

We call a collection of sensors a *division*, which can be represented by a sequence of n binary variables. For example, c_i ($i = 1, 2, ..., n$) can represent a division D, which contains s_i if and only if $c_i = 1$, *i.e.*, all the on-sensors.

We consider a general probabilistic sensing model [4][8]: The probability that an event e can be detected by a sensor s_i is related to the distance between e and s_i if the sensor is on; otherwise, it is zero since off-sensors can never detect an event. As the location of each s_i is fixed, the probability is determined by the event location (x, y) when the sensor is on, which then can be denoted by $p_i(x, y)$.

Given the location of an event (x, y), the sensors detect it in an independent manner, *i.e.*, the probability that the event can be detected by at least one sensor in D is:

$$p_D(x, y) = 1 - \prod_{i=1}^{n}(1 - c_i p_i(x, y)). \quad (1)$$

Among all locations (x, y) in the entire network area ϕ, the minimum value of $p_D(x, y)$ is defined as the *event detection capability* P_D of the network division.

$$P_D = \min_{\forall (x,y) \in \phi} p_D(x, y). \quad (2)$$

This is in fact a pessimistic measure, in which we pick the worst case as the representative case. P_D thus captures how badly the network division might perform in event detection.

Assume events can randomly take place in any place of the network area ϕ in a uniform manner. Suppose we have m discrete quasi-random sampling points in the network area, denoted by $\{t_j\}_{j=1}^m$. The probability that an event taking place at t_j can be detected by s_i is denoted by $c_i p_{ij}$. The minimum probability value among the sampling points, denoted by P'_D, is then:

$$P'_D = \min_{\forall j}[1 - \prod_{i=1}^{n}(1 - c_i p_{ij})]. \quad (3)$$

This P'_D serves as a practical measure of the event detection capability P_D for division D.

2.2 Problem Formulation

Considering the tradeoff between the system migration time and how much a system can tolerate the degrading of event detection capability during a system migration task, a system maintainer can determine how many subsets the network should be divided. Suppose the number is N, with the subsets being denoted by S_k ($k = 1, 2, ..., N$).

Let d_{ik} denote whether s_i is in subset S_k. d_{ik} is 0 if s_i belongs to S_k, and 1 otherwise. In other words, when subset S_k is off[1], sensor s_i is on if $d_{ik} = 1$. So actually each sequence of d_{ik} ($i = 1, 2, ..., n$) denotes the working division D_k during the system migration of sensors in S_k, *i.e.*, let:

$$D_k = \{s_i\}_{i=1}^n - S_k. \quad (4)$$

Suppose in a T-second time interval the network has to be reconfigured into N disjoint subsets S_k ($k = 1, 2, ..., N$) and let each D_k work successively. The *event detection capability of the entire network* in this T-second time interval is then deemed as the minimum among the event detection capability values of all the N divisions D_k ($k = 1, 2, ..., N$). This actually continues the pessimistic considerations as how P_D is defined in Equation (2).

The sensor network reconfiguration problem is thus how to divide the sensors so that the event detection capability of the network in this T-second time interval is maximized. Given the practical measure P'_D in Equation (3), the problem can be formulated as follows.

Problem 1: *The sensor network reconfiguration problem.*
Given:

- A set of sensor nodes $\{s_i\}_{i=1}^n$.
- A set of network locations $\{t_j\}_{j=1}^m$.
- An n by m matrix $\mathcal{P}$ of which each element p_{ij} denotes the probability that sensor s_i (when it is on) successfully detects an event when the event takes place at t_j.

Partition the set $\{s_i\}_{i=1}^n$ into N disjoint subsets S_k ($k = 1, 2, ..., N$) so that:

$P' = \min_{\forall k}\{\min_{\forall j}[1 - \prod_{i=1}^{n}(1 - d_{ik} p_{ij})]\}$ is maximized. ■

This problem is generally NP-Hard. To prove this, let us consider the decision version of this problem in which given the same problem settings, it asks whether the set $\{s_i\}_{i=1}^n$ can be partitioned into

[1]That a subset is off/on means the sensors in the subset are off/on.

N disjoint subsets so that the event detection capability of the network in the T-second time interval is not smaller than a given value u. If the decision version of this problem is NP-Complete, the sensor networks reconfiguration problem is then NP-Hard [9]. In fact, we have the following lemma.

Lemma: The decision version of the sensor network reconfiguration problem is NP-Complete.

Proof: First, this problem is in NP: Given a partition scheme, a non-deterministic algorithm only needs to calculate the event detection capability of each division so as to get the event detection capability of the network during the time interval T. And then it can verify whether this value is smaller than u or not. So now we need to prove that this problem is harder than a known NP-Complete problem.

We transform the provably NP-Complete *set partition problem* [10] to the decision version of the sensor network reconfiguration problem. Given a set of non-negative numbers $\{q_i\}_{i=1}^n$, the set partition problem asks whether it is feasible to partition the set so that the sum of numbers in each partition is equal.

As $p_{ij} \in [0, 1)$, we can construct an n by m matrix $\mathcal{Q}$ of which each element is defined as $q_{ij} = -log_2(1 - p_{ij})$. We can know $q_{ij} > 0$. Based on the property of d_{ik}, we get:

$$1 - d_{ik}p_{ij} = 2^{-d_{ik}q_{ij}}. \tag{5}$$

Let us construct an instance of the sensor network reconfiguration problem in which $N = 2$, q_{ij} is equal to each other given the same i and equal to the q_i in the set partition problem, and $u = 1-2^{-(\sum_{i=1}^n q_i)/2}$. Now we can always have $d_{i1} = 1-d_{i2}$ because a sensor should be in either division D_1 or division D_2, but not in both. Also we can write q_{ij} as q_i without the subscript j. Therefore, we get:

$$\begin{aligned} P' - u &= \min_{\forall k}\{\min_{\forall j}[1 - \prod_{i=1}^{n}(1 - d_{ik}p_{ij})]\} - u \\ &= \min_{\forall k}[\min_{\forall j}(1 - 2^{-\sum_{i=1}^n d_{ik}q_{ij}})] - u \\ &= \min_{\forall k}(1 - 2^{-\sum_{i=1}^n d_{ik}q_i}) - u \\ &= 2^{-\frac{\sum_{i=1}^n q_i}{2}} - 2^{-[\min_{\forall k}(\sum_{i=1}^n d_{ik}q_i)]}. \end{aligned} \tag{6}$$

If the answer to whether $P' \geq u$ is *yes*, we get:

$$\begin{aligned} &\min_{\forall k}(\sum_{i=1}^{n} d_{ik}q_i) \geq \frac{\sum_{i=1}^{n} q_i}{2} \\ &\Rightarrow \begin{cases} \sum_{i=1}^{n} d_{i1}q_i \geq \frac{\sum_{i=1}^n q_i}{2} \\ \sum_{i=1}^{n} d_{i2}q_i = \sum_{i=1}^{n}(1 - d_{i1})q_i \geq \frac{\sum_{i=1}^n q_i}{2} \end{cases} \\ &\Rightarrow \frac{\sum_{i=1}^n q_i}{2} \geq \sum_{i=1}^{n} d_{i1}q_i \geq \frac{\sum_{i=1}^n q_i}{2} \\ &\Rightarrow \sum_{i=1}^{n} d_{i1}q_i = \frac{\sum_{i=1}^n q_i}{2}. \end{aligned} \tag{7}$$

Therefore, the answer to the set partition problem is also *yes*.

On the other hand, if the answer to the set partition problem is *yes*, in the same way we can partition the sensors in the sensor network reconfiguration problem so that:

$$\begin{aligned} &\begin{cases} \sum_{i=1}^{n} d_{i1}q_i = \frac{\sum_{i=1}^n q_i}{2} \\ \sum_{i=1}^{n} d_{i2}q_i = \sum_{i=1}^{n}(1 - d_{i1})q_i = \frac{\sum_{i=1}^n q_i}{2} \end{cases} \\ &\Rightarrow \min_{\forall k}(\sum_{i=1}^{n} d_{ik}q_i) = \frac{\sum_{i=1}^n q_i}{2}. \end{aligned} \tag{8}$$

According to Equation (6), $P' = u$. Therefore, the answer to the decision version of the sensor network reconfiguration problem is also *yes*.

The above reduction requires only $O(n)$ steps to be completed (for calculating p_i and u with q_i). Therefore, the decision version of the sensor network reconfiguration problem is both NP-Hard and NP. Then it is NP-Complete. The lemma is proved.■

3. HEURISTICS FOR DOWNTIME-FREE SYSTEM MIGRATION

Given that the sensor network reconfiguration problem is NP-Hard, we resort to heuristics that can find the approximation solutions efficiently. We start from investigating this question: What should we make the resulting subsets look like, if we want to design a good approximation algorithm?

For each solution of this problem, there exists one sample point t_x where the event detection capability of some division results in the minimum value, *i.e.*,

$$x = \operatorname*{argmin}_j\{\min_{\forall k}[1 - \prod_{i=1}^{n}(1 - d_{ik}p_{ij})]\}. \tag{9}$$

Suppose division D_y results in the minimum event detection capability at t_x, *i.e.*,

$$y = \operatorname*{argmin}_k[1 - \prod_{i=1}^{n}(1 - d_{ik}p_{ix})]. \tag{10}$$

Also suppose the event detection probability for each set S_k at t_x is r_k. The event detection probability for each division D_k at t_x is:

$$1 - \prod_{i=1, i\neq k}^{N}(1 - r_i) = 1 - \frac{\prod_{i=1}^{N}(1 - r_i)}{1 - r_k}. \tag{11}$$

Since $\prod_{i=1}^{N}(1 - r_i)$ is the same for all D_k, if the event detection probability of D_y is the minimum among all D_k, $1 - r_y$ should be the smallest among all $1 - r_k$ ($\forall k$). In other words, the event detection probability of S_y at t_x is the largest among all the subsets S_k.

The larger the event detection probability of S_y at t_x, the smaller the event detection probability of D_y at t_x. To maximize the event detection probability of D_y at t_x, the event detection probability of S_y at t_x should be minimized. Therefore, a good heuristic algorithm should let r_y be as approximate as possible to the event detection probability of other subsets at t_x.

This consideration can be directly applied to an algorithm that solves the sensor network reconfiguration problem: After initially grouping nodes into each S_k, we can greedily move the nodes in subset S_y to other subsets so as to reduce r_y. This is the mechanism of a greedy algorithm that would be discussed in Section 3.1.

Given the situation that the number of the schemes to group n into N subsets is exponentially related to n, while the event location is a continuous variable which can be anywhere in the network, a possible t_x can be almost anywhere of the network. To minimize r_y, it would therefore be better if all the subsets can have an closer event detection probability at any point in the network. In order to make the event detection probability of all the subsets approximate to each other at any points, the resulting subsets should look similar in a dispersive manner.

It is therefore necessary that nodes in the same subset should be dispersedly distributed. Nodes that are near to each other should not be grouped into the same subset so as to avoid high event detection probability of the subset at locations around these nodes. In other words, if we examine an arbitrary area in the network, there should not be outstanding dominant-population of any one of the subsets.

Based on this consideration, we design the other three algorithms, namely, the Simple Partitioning and Picking (SPP) algorithm, the Minimum Spanning Tree-Based Grouping (MSTBG) algorithm, and the Sensor Network Reconsturing Protocol (SNRP), to attack the problem.

SPP tries to maximize the ι index, *i.e.*, the minimum distance over the average distance between each node pair [4][7], of the resulting subset. Because the ι index can serve as a good microscope in indicating the existing of a high redundancy area, by maximizing this index, SPP aims at avoiding high redundancy of some subsets comparing to the others at anywhere in the network. Similarly, MSTBG constructs a minimum spanning tree (MST) of the network incrementally, and groups each newly-joining node to a farthest subset where the distance here means the distance between the node and the nearest member in the subset. Thus MSTBG tries to group nodes that are near to each other to different subsets so as to avoid the close-gathering of the nodes in the same subset. SNRP implements a similar mechanism as that in MSTBG. However, unlike other algorithms that are centralized, SNRP is specifically tailored as a distributed and localized algorithm for WSNs. Details on SPP, MSTBG, and SNRP would be discussed in Sections 3.2-3.4.

3.1 Greedy Algorithm (GA)

The greedy algorithm (GA) first randomly selects $\frac{n}{N}$ nodes for each subset S_k.

Let p_{min} denotes the minimum event detection probability among the event detection probabilities of any division (*i.e.*, $\forall D_k = \{s_i\}_{i=1}^{n} - S_k$) at any sampling point. GA locates the sampling point t_x at which the event detection probability of some division (denoted by D_y) is p_{min}.

According to the discussions in the beginning of this section, the event detection probability of S_y is the maximum among all S_k at t_x. Now suppose the event detection probability of S_z is the minimum among all S_k at t_x.

GA tries to improve p_{min} by moving a node from S_y to S_z. A node is selected if it would result in the largest improvement of p_{min} comparing to selecting any other node. Ties are broken arbitrarily.

The above procedure is iteratively conducted until p_{min} cannot be further improved.

3.2 Simple Partitioning and Picking Algorithm (SPP)

The SPP algorithm performs two procedures in turn: the partitioning procedure and the merging procedure. In the partitioning procedure, suppose there is a Cartesian coordinate system in the network area. First consider all nodes are in one region. Perform the following two steps iteratively until there are less than $2N$ nodes in every region and then nodes in each region are randomly selected into N different subsets.

- *Step 1:* For each region, draw a line parallel to the x-axis to partition the region into two so that the number of nodes in each partition is the same[2].
- *Step 2:* Then for each region, draw a line parallel to the y-axis to partition the region into two so that the number of nodes in each partition is the same.

In the merging procedure, neighboring regions are merged into one till there is only one region. The merging method is as follows. First randomly assign two neighboring regions as region A and region B. Couple each randomly-picked subset (say A1) in region A and a selected subset in region B. The selection criterion is that the couple can result in the largest ι comparing to the other couples formed by A1 and any other subsets in B (Ties are broken by picking randomly). This coupling process continues until all N couples are generated (since each region has N subsets). Then each couple is deemed as a subset, and thus A and B are merged into a larger region.

3.3 Minimum Spanning Tree-Based Grouping Algorithm (MSTBG)

The MSTBG algorithm first builds a tree that is composed only by the two nearest nodes among all the in-network nodes. These two nodes are grouped into two different subsets.

Select a node which is nearest to the tree among all the nodes that are not in the tree. The distance between this node and a subset is defined as the distance between this node and the nearest node in the tree which is in the subset. Group this node to the subset which is the farthest to this node. Then add this node to the tree. This procedure is thus iteratively conducted until all nodes are in the tree.

Such a process of building a tree is exactly how Prim's algorithm does in building a minimum spanning tree (MST) [11]. This is why we call this algorithm an MST-based grouping algorithm.

3.4 SNRP: Distributed and Localized Sensor Network Reconfiguration Protocol

[2]If the number of nodes is odd, then one arbitrarily selected region can have one more node than the other. Similarly for Step 2.

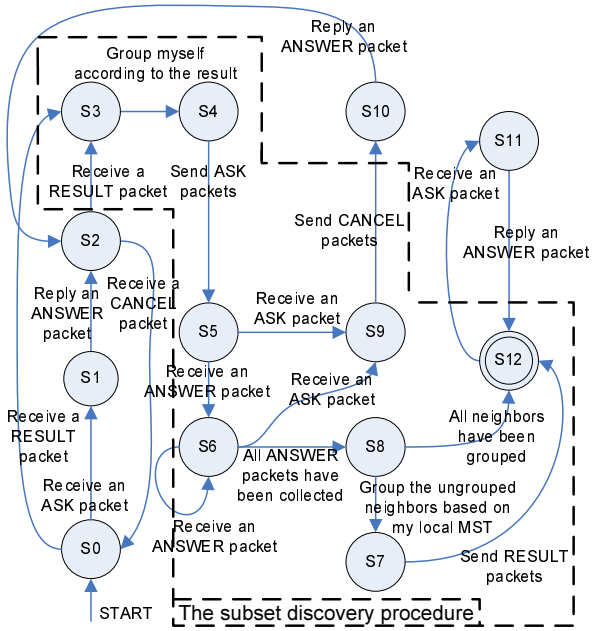


Figure 1: The finite state machine of SNRP

The algorithms discussed above (*i.e.*, GA, SPP, and MSTBG) are all centralized approaches. A global picture of the network is required to run these algorithms. However, WSNs are usually large-scale networks which contain hundreds of nodes. Global information is not easy, if not impossible, to be obtained. According to the features of WSNs, a distributed and localized solution for the sensor reconfiguration problem is surely of practical interests.

Although some well-investigated mechanism can help implement the above algorithms in a distributed way, for example, a distributed MST algorithm (*e.g.*, [12]) can be applied to decentralize MSTBG, global information is still inevitably required in constructing an MST [13].

We therefore design a new distributed and localized algorithm called the Sensor Network Reconfiguration Protocol (SNRP). It is based on a mechanism similar to that in MSTBG. But to tailor a distributed and localized algorithm for WSNs, instead of constructing an MST of the whole network, each in-network node builds local MST of the node's neighborhood graph (*i.e.*, the graph consisted of the node and its one-hop neighbors) and group neighboring nodes to the subsets based on the local MST.

SNRP is an event (packet) driven algorithm. There are four types of packets involved in this algorithm, *i.e.*, ASK, CANCEL, ANSWER, and RESULT packets. Figure 1 demonstrates SNRP with a finite state machine. Details on the protocol are as follows.

Initially, a node does not belong to any subset (S0). The base station (*i.e.*, the network control center) will firstly send a RESULT packet to a randomly selected node, telling it that it belongs to subset 1 and let it begin to perform the subset discovery procedure.

When a node (suppose it is node s) receives a RESULT packet (S0→S3), it starts its subset discovery procedure by firstly sending ASK packets to enquire its neighbors which subset they belong to (S4→S5). It waits until every neighbor has replied with an ANSWER packet (S6→S8). Then node s constructs a local MST of its neighborhood graph. Based on the same mechanism as that in MSTBG, it groups each of the nodes which do not belong to any subset into a subset (S8→S7). Then node s notifies these nodes the grouping results by sending them RESULT packets (S7→S12). Thus the subset discovery procedure is handed over to the neighboring nodes of node s and node s comes to the final state ($S12$).

If a node receives an ASK packet from a neighbor (again, suppose the neighbor is node s), the node will behave differently according to whether or not it is currently conducting the subset discovery procedure. If this is true, *i.e.*, when the node is waiting for collecting all ANSWER packets in the subset discovery procedure (S5 or S6), in order to avoid deadlocks it will send CANCEL packets to all the neighbors to which it has sent ASK packets (S9→S10) before it sends an ANSWER packet to node s in reporting which subset it belongs to (S10→S2). Otherwise, ($S0$ or $S12$), it will directly sends an ANSWER packet to node s (S1→S2 or S11→S12).

Then after sending the ANSWER packet to node s, the node will return to the final state if the node has successfully performed the subset discovery procedure before (S12). Otherwise, it waits for a RESULT packet or a CANCEL packet from node s (S2). Note that the node will also queue the ASK packets from other neighbors without reply during this waiting time. This can avoid that the node is grouped into different subsets by different neighbors. Now if a CANCEL packet is received, the waiting is canceled (S2→S0) and the node returns to the initial state (S0).

4. PERFORMANCE STUDY

To study the effectiveness of our algorithms in solving the sensor network reconfiguration problem, we build a customized sensor network simulator. The quasi-random sampling scheme we adopt is the 2-dimensional Hammersley sequence [14], which is linearly mapped to the network area to sample the event detection probability of the network area. Detailed settings of the simulation networks are shown in Table 1. α, β, γ and ϵ in the table are parameters of the probabilistic sensing model [4] in which if an event is L meters away from a sensor, the sensor can detect the event with probability p that satisfies:

$$p = \begin{cases} \frac{\delta}{(L/\epsilon+1)^{\beta}} & \text{if } L \leq R_s, \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

This model implies that the event detection probability is determined by the event-signal strength received by a sensor, while the signal fades exponentially with a factor equal to β in its way from the event location to the sensor location. This is a realistic consideration.

Table 1: Simulation Settings

Area of sensor field	200m × 200m
Rode deployment scheme	Randomly deployed in a uniform manner
Sensing range R_s	40m
Communication range R_c	40m
α, β, γ and ϵ	1.0, 2.0, 1.0 and 40.0
Number of sampling points	100
Sampling method	2-dimensional Hammersley sequence

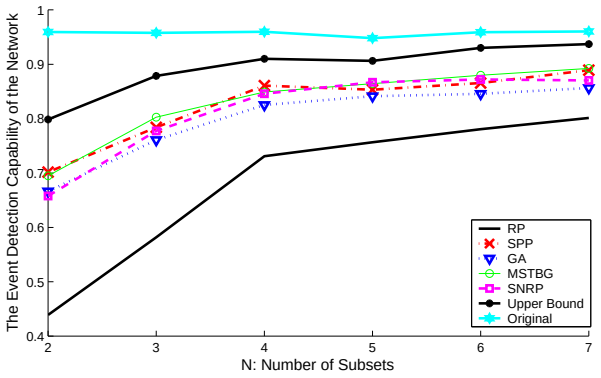


Figure 2: EDC as a function of N (Node Number = 100)

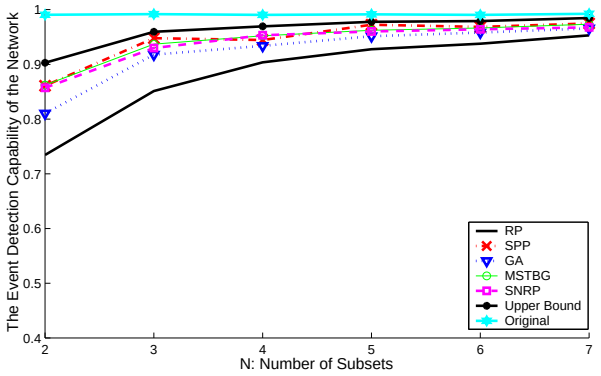


Figure 3: EDC as a function of N (Node Number = 150)

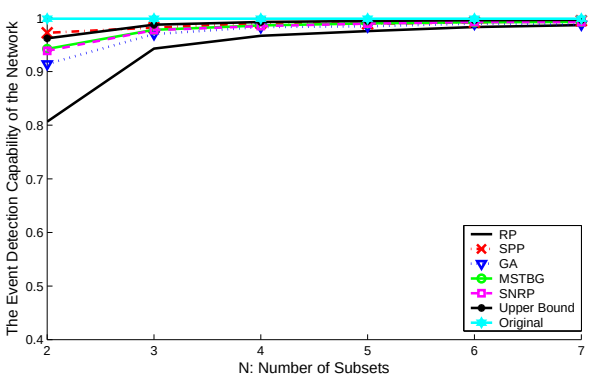


Figure 4: EDC as a function of N (Node Number = 200)

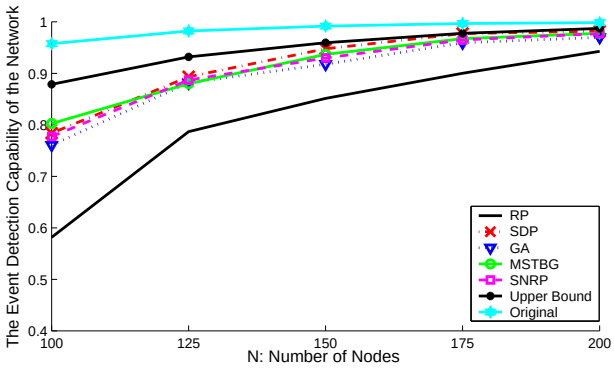


Figure 5: EDC as a function of node number

We employ SPP, GA, MSTBG, and SNRP to reconfigure the in-network sensor nodes into N subsets. We study the event detection capability (EDC) of the network during a system migration task where each subset has to cease to work for a given period of time successively. For each network setting, simulations are performed for 100 times with different random seeds and the results are averaged.

For comparison purpose, we also draw another three curves. The first curve (named "Original" in the figures) shows the EDC of the entire network when no subset is off. The second curve (named "Upper Bound" in the figures) shows the EDC upper-bound of the network when one subset cease to work, which is computed by:

$$1 - (1 - P_{all})^{\frac{N-1}{N}} \tag{13}$$

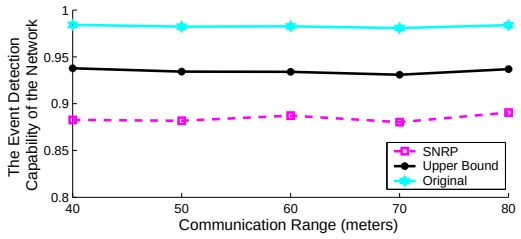
where P_{all} is the EDC of the entire network when no subset is off. This is a *non-achievable* upper-bound, however, as it considers the non-achievable but optimum case where each subset has equal event detection probability at any point of the network. Lastly, the third curve (named "RP" in the figures) is the EDC of the network when reconfigured by the Random Pick algorithm (RP) we design, in which we randomly select n/N nodes for each subset without any performance considerations. This serves as a baseline in our simulation study.

We first study how the value of N influences our algorithms. Figures 2-4 show the EDC of the networks composed of different numbers of nodes. We can see that the naive RP algorithm performs by far the worst, which is what we have expected. SPP, MSTBG, and SNRP always perform better then GA, which verifies that it is necessary to disperse the nodes in the same subset.

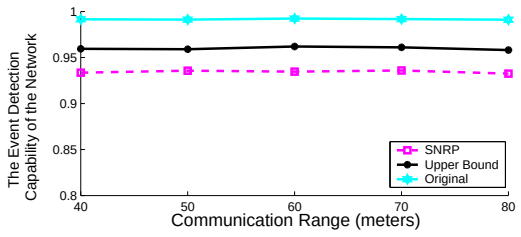
Also it can be found out that when N is large enough ($N > 4$ in our simulations), improving N cannot effectively improve the EDC of the network. This is not strange as the difference between the ratios $\frac{N-1}{N}$ and $\frac{N}{N+1}$ gets smaller as N increases. As a larger N incurs longer time for the entire network to complete a migration task, the price of improving the EDC of the network during system migration becomes higher and higher as N increases. This should be an important consideration for a system maintainer to select a proper value of N.

When the node-density becomes higher, the differences among the performances of these algorithms become smaller. This is not surprising, either. The more the in-network nodes, the higher the redundancy of the network with respect to event detection. As a result, when the node density is high, if one subset of the nodes is off, it does not have a great impact on the performance of the network. Figure 5 further demonstrates this idea. We let $N = 3$ and change the number of nodes from 100 to 200. We can clearly see that the EDC of the network gradually approaches the original curve.

To see how the neighborhood graph size influences the results of SNRP, we change the communication range R_c from $40m$ to $80m$ (*i.e.*, from one time to two times of the sensing range). Figure 6 demonstrates the example results of SNRP where $N = 3$ and the node numbers are 125 and 150, respectively. We can find out that SNRP performs almost the same when R_c is larger than the sensing



(a) Node Number = 125



(b) Node Number = 150

Figure 6: EDC as a function of communication range

range. As usually the communication range of a node is larger than its sensing range, these results verify that grouping based on the local MST is good enough comparing to grouping based on the global MST. It shows that SNRP, as a distributed and localized algorithm, is very successful as such a specifically-tailored design does not degrade the resulting performance much.

5. CONCLUSIONS AND FURTHER DISCUSSIONS

Seamless system migration without downtime is necessary for wireless sensor networks that perform critical event detection tasks. Unfortunately, to our knowledge, this important problem has not been addressed in the literature. In this paper, we presented the first formal study on this problem. We demonstrated that the downtime can be eliminated by partitioning the sensors into a collection of subsets, and let each subset conduct the system migration tasks successively with the rest still performing normal event detection services. We proved the optimal partitioning of sensors in this context is NP-hard and then proposed a series of heuristics. We further extended our solution to a distributed implementation called the Sensor Network Reconfiguration Protocol (SNRP). Simulation results showed that these algorithms work well; Yet, we believe there is still room to extend this research. In particular, if the node locations are not available, we need to find a way to divide the sensors according to the in-network nodes' neighborhood information.

6. ACKNOWLEDGEMENT

The work was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4150/07E). J. Liu's work was supported in part by a Canadian NSERC Discovery Grant and an NSERC Strategic Project Grant.

7. REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on wireless sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] A. Kansal, S. Nath, J. Liu, and F. Zhao, "SenseWeb: an infrastructure for shared sensing," *IEEE Multimedia*, vol. 14, no. 4, pp. 8–13, October-December 2007.

[3] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming wireless sensor networks: Challenges and approaches," *IEEE Network*, pp. 48–55, May-June 2006.

[4] Y. Zhou, H. Yang, M. R. Lyu, and E. C.-H. Ngai, "A point-distribution index and its application to sensor-grouping in wireless sensor networks," in *Proc. of the International Wireless Communications and Mobile Computing Conference (IWCMC)*, Vancouver, Canada, July 2006.

[5] M. Cardei and D. Du, "Improving wireless sensor network lifetime through power aware organization," *ACM Journal of Wireless Networks*, vol. 11, no. 3, pp. 333–340, May 2005.

[6] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Proc. of the IEEE International Conference on Communications (ICC'01)*, vol. 2, Helsinki, Finland, June 2001.

[7] Y. Zhou, M. R. Lyu, and J. Liu, "A sensor-grouping mechanism for field-coverage wireless sensor networks," in *Proc. of the IEEE International Conference on Communications (ICC'08)*, Beijing, China, May 2008.

[8] B. Liu and D. Towsley, "A study of the coverage of large-scale sensor networks," in *Proc. of the 1st IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS'04)*, Fort Lauderdale, FL, October 2004.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York City, NY: W. H. Freeman and Co., 1979.

[10] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. (R. Miller and J. Thatcher eds.) New York: Plenum, 1972.

[11] R. C. Prim, "Shortest connection networks and some generalisations," *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[12] P. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning tree," *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 5, pp. 66–77, January 1983.

[13] X.-Y. Li, Y. Wang, P.-J. Wan, W.-Z. Song, and O. Frieder, "Localized low-weight graph and its applications in wireless ad hoc networks," in *Proc. of the 23rd IEEE INFOCOM*, Hong Kong, March 2004.

[14] J. M. Hammersley, "Monte-Carlo methods for solving multivariable problems," *Annals of the New York Academy of Science*, vol. 86, pp. 844–874, 1960.