Mobile Netw Appl (2009) 14:241–252
DOI 10.1007/s11036-008-0130-9

On Sensor Network Reconfiguration for Downtime-Free System Migration

Yangfan Zhou · Michael R. Lyu · Jiangchuan Liu

Published online: 17 January 2009

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 processing 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 be effectively 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. 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.

Keywords sensor system migration · sensor network reconfiguration · network partition

1 Introduction

Wireless sensor networks (WSNs) are usually employed to sense the physical-phenomenon data of interest and accordingly perform environmental event detection tasks [1]. There are many potential applications of WSNs. Examples include forest fire detection where a number of nodes equipped with thermoelectric and hygrometric sensors work collaboratively in raising a fire alarm, and borderline surveillance where many nodes equipped with infrared and acoustic sensors are deployed to conduct intruder detection.

It is usually expensive, if not impractical, for human-attended operations on a sensor node (especially for the WSNs applied in battle-field or habitat monitoring [12]). As a result, in most application scenarios, WSNs are expected to work in an unattended manner for a long period of time (usually several months) once the sensor nodes have been deployed.

Although sensing/processing environmental data is the major task of a WSN, during its month-long lifetime, it is inevitable for the WSN to perform certain system tasks in addition to the sensing/processing

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.

Y. Zhou (✉) · M. R. Lyu
Department of Computer Science & Engineering,
The Chinese University of Hong Kong,
Shatin, Hong Kong, China
e-mail: yfzhou@cse.cuhk.edu.hk

M. R. Lyu
e-mail: lyu@cse.cuhk.edu.hk

J. Liu
School of Computing Science, Simon Fraser University,
Burnaby, British Columbia, Canada
e-mail: jcliu@cs.sfu.ca

task, *e.g.*, system maintenance [2], diagnosis [5], and upgrading [16]. However, typical sensor platforms are simple and low-cost in nature. Multi-thread is usually not supported due to the capacity limitation of a sensor node [9]. As a result, these system tasks are exclusive to the data sensing/processing operations, *i.e.*, a sensor node has to cease data sensing/processing operations in performing these tasks. The most critical example of such system tasks is the WSN reprogramming task, which typically takes a network with one hundred sensor nodes a few hundred seconds to complete [16]. We use the term *system migration tasks* to include these long-term exclusive system processes 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.

Although there have been significant research efforts on implementing efficient online migration tasks for WSNs [16], they largely ignored the interruption of the major sensing/repoting task caused by such a system migration task. 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/intrusion event which happens during the process. A reprogramming interval of hundreds of seconds 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 a seamless scheme for performing the exclusive system tasks, which avoids 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 exclusive 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 tradeoff between system migration time and performance degradation.

Although various sensor grouping problems have been studied (e.g., work in [3, 15, 17, 18]), 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 formulate the new *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 and further extend one of them 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 the system migration time.

The rest of the paper is organized as follows. Section 2 briefly introduces the related work. In Section 3, we provide a formal description of this problem and justify its formulation. We then analyze this problem and shows its NP-hardness. Section 4 provides several algorithms in attacking this problem. The performance of these algorithms is studied in Section 5. We conclude this paper in Section 6.

2 Related work

Implementing system migrations such as reprogramming is crucial to the success of WSN applications, with which bugs can be eliminated and functionalities of a network can be updated. See a survey paper and the references therein [16]. However, existing work has not notified the problem that the major sensing/repoting task would be interrupted by a system migration task. Our proposal is to divide sensor into subsets and let each subset perform the migration task in turn. This scheme is then orthogonal to a reprogramming approach. It can be easily adopted in many existing reprogramming protocols, especially those that support the reprogramming of a selected group of nodes (e.g., [13]).

Much work has been done on how to divide the sensor nodes in a network into disjoint subsets where each subset can maintain the required sensing tasks. In [15], a sensing field is divided into regions. Sensor nodes are grouped with the most-constrained least-constraining algorithm, in which the priority of selecting a sensor to a subset is determined by how much uncovered area this sensor covers and the redundancy caused by this sensor. In [4], the problem is modeled as disjoint dominating sets, which is known as NP-complete. It then proposes a graph-coloring based approximation. A similar problem of covering target points are studied in [3], which is again NP-complete and mixed integer programming (MIP) approximation has been proposed.

Work in [17, 18] shows that maximizing the normalized minimum distance between sensors of a subset results in low redundancy of the subset, based on which fast algorithms are proposed to find the low-redundancy subsets of sensor nodes. These approaches, however, are unapplicable to our problem, as the objectives are generally different.

3 Models and problem formulations

3.1 Preliminaries

We consider a WSN composed of n stationary sensor nodes $\{s_i\}_{i=1}^n$ randomly deployed in a uniform manner in a network area ϕ. Let the status of s_i be denoted by a binary variable c_i. c_i is 1 if s_i is conducting event detection work, and 0 if s_i is performing a system migration task such as being reprogrammed. For the convenience of our discussion, we say that a sensor s_i is *on* if $c_i = 1$, and *off* if $c_i = 0$. 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 [11, 18]: 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. Let $p_i(x, y)$ denote this probability. Given the location of an event (x, y), the probability that the event can be detected by at least one sensor in D is:

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

Note that the network is employed to detect events and the events of interest can take place in any random location of the network area ϕ. As a result, the minimum value of $p_D(x, y)$ among all in-network locations (x, y) is a natural index to capture how badly the network division D might perform in event detection, which can then be considered as the network division D's capability on event detection. Formally, we define that the *event detection capability* P_D of the network division D is the minimum value of $p_D(x, y)$ among all locations (x, y) in the entire network area ϕ, i.e.,

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

This is a pessimistic measure, in which we pick the worst case as the representative case.

3.2 Problem formulation

We consider that the network should be divided into N subsets, denoted by S_k $(k = 1, 2, ..., N)$. In order to enforce downtime-free system migration, let each of the subsets perform the exclusive system migration task in turn, while the rest of the subsets still remain normal operations in sensing/processing environmental data.

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 (that a subset is off/on means all the sensors in the subset are off/on), sensor s_i is on if and only 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:

$$D_k = \{s_i\}_{i=1}^n - S_k. \quad (3)$$

In other words, D_k denotes the division of the sensor nodes which are conducting normal sensing and processing work when S_k is performing a migration task. d_{ik} then denotes whether s_i is in D_k.

N is naturally a parameter of such a downtime-free migration scheme. On one hand, a larger subset number means the longer time it requires for the whole network to finish the system migration task as the task is performed by each subset in turn. On the other hand, a larger subset number also means there are a smaller number of nodes in each subset. Since each time only a subset is off in performing event detection task, fewer nodes in each subset implies that the system performance (in terms of event detection capability) degrades less during the system migration task.

The number of subsets N can thus serve as a flexible parameter for a system maintainer to fine-tune the system migration process. 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. Given an N by the system maintainer, the critical problem then becomes how the sensor nodes should be partitioned into subsets so as to achieve the best tradeoff between system migration time and performance degradation.

Suppose during a system migration, the network has to be reconfigured into N disjoint subsets S_k $(k = 1, ..., N)$ and let each D_k work successively. The *event detection capability of the entire network* in this migration interval is defined as the minimum among the event detection capability values of all the N divisions

D_k ($\forall k$). This actually continues the pessimistic considerations as how P_D is defined in Eq. 2.

The sensor network reconfiguration problem is thus how to divide the sensors so that the event detection capability of the network in this migration interval is maximized. Given the event detection capability measure P_D in Eq. 2, the problem can be formulated as follows.

Problem 1 The sensor network reconfiguration problem.

Given:, $\{s_i\}_{i=1}^n$ and $p_i(x, y)$

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

$$P = \min_{\forall k} P_{D_k} = \min_{\forall k} \left\{ \min_{\forall (x,y) \in \phi} \left[1 - \prod_{i=1}^{n} (1 - d_{ik} p_i(x, y)) \right] \right\}$$

is maximized.

3.3 The difficulty of the sensor network reconfiguration problem

Given the situation that the number of the schemes to group n into N subsets is related to the permutation of n, while the event location is a continuous variable taken a value in the whole network area, the network location that results in $\min_{\forall k} P_{D_k}$ can be anywhere in the network. This makes it difficult to handle the sensor network reconfiguration problem.

Instead of dealing with all the network locations, let us confine our considerations such that events can just take place at a finite set of discrete points in the network area. Suppose we have m such discrete points in the network area, denoted by $\{t_j\}_{j=1}^m$. We call them the *sampling points* of the network area ϕ. The probability that an event taking place at t_j can be detected by s_i is then denoted by $c_i p_{ij}$.

The minimum probability value among the sampling points, denoted by P'_D, is then:

$$P'_D = \min_{\forall j} \left[1 - \prod_{i=1}^{n} (1 - c_i p_{ij}) \right]. \tag{4}$$

This P'_D thus serves as a simplified measure of the event detection capability P_D for division D. Then the sensor network reconfiguration problem turns to a *simplified* one given the practical measure P'_D in Eq. 4, which can be formulated as follows.

Problem 2 The simplified sensor network reconfiguration problem.

Given: $\{s_i\}_{i=1}^n$, $\{t_j\}_{j=1}^m$, and p_{ij} $(\forall i, j)$

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

$$P' = \min_{\forall k} P'_{D_k} = \min_{\forall k} \left\{ \min_{\forall j} \left[1 - \prod_{i=1}^{n} (1 - d_{ik} p_{ij}) \right] \right\}$$

is maximized.

Unfortunately, even this simplified 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 N disjoint subsets so that the event detection capability of the network in the migration interval is not smaller than a given value u. If the decision version of this problem is NP-Complete, then the sensor networks reconfiguration problem is NP-Hard [7]. In fact, we have the following lemma to prove the NP-Hardness of the simplified sensor network reconfiguration problem.

Lemma 1 *The decision version of the simplified sensor network reconfiguration problem is NP-Complete.*

Proof See Appendix. □

4 Heuristics for downtime-free system migration

Given the difficulty of the sensor network reconfiguration problem, 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?

Let us again consider the simplified sensor network reconfiguration problem (Problem 2) first. For each solution to this problem, there exists one sample point t_x where the event detection capability of some division results in the minimum value, i.e.,

$$x = \operatorname*{argmin}_{\forall j} \left\{ \min_{\forall k} \left[1 - \prod_{i=1}^{n} (1 - d_{ik} p_{ij}) \right] \right\}. \tag{5}$$

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

$$y = \operatorname*{argmin}_{\forall k} \left[1 - \prod_{i=1}^{n} (1 - d_{ik} p_{ix}) \right]. \tag{6}$$

Let r_k denote the event detection probability for each set S_k at t_x. We get:

$$r_k = 1 - \prod_{i=1}^{n}(1 - d_{ik}p_i(t_x)), \tag{7}$$

where $p_i(t_x)$ denotes the event detection probability of sensor i at location t_x.

The event detection probability for each division D_k at t_x, denoted by $p'_{D_k}(t_x)$, is then:

$$p'_{D_k}(t_x) = 1 - \prod_{l=1,l\neq k}^{N}(1 - r_l) = 1 - \frac{\prod_{l=1}^{N}(1 - r_l)}{1 - r_k}. \tag{8}$$

$\prod_{l=1}^{N}(1 - r_l)$ is a constant based on Eq. 7 because

$$\prod_{l=1}^{N}(1 - r_l) = \prod_{l=1}^{N}\prod_{i=1}^{n}(1 - d_{il}p_i(t_x)) = \prod_{i=1}^{n}(1 - p_i(t_x)), \tag{9}$$

which is irrelevant to how we group the sensor nodes.

Then based on Eq. 8, 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, r_y, *i.e.*, the event detection probability of subset S_y at t_x, must be 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 close as possible to the event detection probability of other subsets at t_x.

4.1 Greedy algorithm (GA)

The above consideration can be directly applied to an algorithm that solves the simplified sensor network reconfiguration problem (Problem 2): 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.

Algorithm 1 demonstrates the mechanism of this greedy algorithm (GA). It first randomly selects $\frac{n}{N}$ nodes for each subset S_k (line 2). Let p_{min} denote the minimum event detection probability among the event detection probabilities of any division D_k ($\forall k$) at any sampling point (line 5). GA locates the sampling point t_x at which the event detection probability of some division (denoted by D_y) is p_{min} (line 6). Based on the above discussions, 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 improves p_{min} by moving a node from S_y to S_z, which results in the largest improvement of p_{min} (lines 7–9). p_{min} is thus improved iteratively until it cannot be further improved (lines 4–10).

Algorithm 1 Greedy Algorithm (GA)

1: **Input:**
   $\{s_i\}_{i=1}^{n}$: the set of sensor nodes
   $\{t_j\}_{j=1}^{m}$: the set of sampling points
   $\mathcal{P}$: the n by m matrix, where each element p_{ij} denotes the event detection probability of s_i at t_j.
2: Randomly selects $\frac{n}{N}$ nodes for each subset S_k
3: $D_k \leftarrow \{s_i\}_{i=1}^{n} - S_k$
4: **repeat**
5: $p_{min} \leftarrow$ the minimum event detection probability among the event detection probabilities of any divisions at any sampling points
6: $t_x \leftarrow$ the sampling point at which the event detection probability of a division is p_{min}.
7: $D_y \leftarrow$ the division that results in p_{min} at t_x
8: $S_z \leftarrow$ the subset whose event detection probability is the minimum at t_x.
9: Move a node from S_y to S_z. A node is selected if it results in the largest improvement of p_{min} comparing with selecting any other node. Ties are broken arbitrarily.
10: **until** p_{min} cannot be further improved

Although Algorithm 1 is to solve Problem 2, with a set of well-designed sampling points $\{t_j\}_{j=1}^{m}$, its output can be deemed as a solution of Problem 1. The prerequisite is that the sampling points can well represent the event detection probability of the whole network. Quasi-random sequences, such as Hammersley sequence, which have asymptotically optimal discrepancy (a measure of uniformity for the distribution of the points) possess been widely employed in Quasi Monte Carlo methods [8]. In this regard, they are reasonably good sampling-point generators. We adopt Hammersley sequence [8] to generate the sampling points for Algorithm 1. We linearly map the 2-dimensional Hammersley sequence into the network area to generate the locations of the sampling points $\{t_j\}_{j=1}^{m}$ as the input of Algorithm 1.

4.2 Simple partitioning and picking algorithm and minimum spanning tree-based grouping algorithm

In the greedy algorithm, t_x is found in the set of sampling points. In fact, given the original settings of Problem 1, i.e., the event location is a continuous variable which can be anywhere in the network, actually

Algorithm 2 Simple Partitioning and Picking Algorithm (SPP)

1: **Input:**
   $\{s_i\}_{i=1}^{n}$: the set of sensor nodes
2: The whole network area is deemed as a region
3: **repeat**
4: **for all** regions **do**
5: draw a line parallel to the x-axis to partition the region into two so that the difference between the node numbers in both partitions is at most one.
6: **end for**
7: **if** there are more than $2N$ nodes in each region **then**
8: **for all** regions **do**
9: draw a line parallel to the y-axis to partition the region into two so that the difference between the node numbers in both partitions is at most one.
10: **end for**
11: **end if**
12: **until** there are less than $2N$ nodes in each region
13: randomly select nodes in each region to $S_k(k = 1, ..., N)$
14: **repeat**
15: randomly assign two neighboring regions as A and B
16: **for** each randomly-picked subset A_k in region A **do**
17: Couple A_k with a subset B_x in region B, such that the couple results in the largest ι comparing with the other couples formed by A_k and any other subsets in B. Ties are broken arbitrarily.
18: **end for**
19: each couple is deemed as a subset, and thus A and B are merged into a larger region.
20: **until** there is only one region

a possible t_x can also be anywhere in the network. To minimize r_y, it would therefore be better if all the subsets have a closer event detection probability at any point in the network as we do not know where t_x should be.

In order to make the event detection probability of all the subsets close 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 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 two algorithms, namely, the Simple Partitioning and Picking (SPP) algorithm and the Minimum Spanning Tree-Based Grouping (MSTBG) algorithm.

SPP tries to maximize the ι index, *i.e.*, the minimum distance over the average distance between each node pair [17, 18], of the resulting subsets. Because the ι index can serve as a good microscope in indicating the existing of a high redundancy area [17], by maximizing this fan-out index, SPP aims at avoiding high redundancy of some subsets comparing to the others at anywhere in the network.

Algorithm 2 shows the details of SPP. It performs two procedures in turn: the partitioning procedure (lines 2–13) and the merging procedure (lines 14–20). In the partitioning procedure, first consider all nodes are in one region. Supposing there is a Cartesian coordinate system in the network area, SPP iteratively cuts each region into two until there are less than $2N$ nodes in every region (lines 4–12). Then nodes in each region are randomly selected into N different subsets (line 13). In the merging procedure, neighboring regions are merged into one till there is only one region. During the merge procedure, one subset in a region is coupled to anther subset in another region if the couple can result in the largest ι comparing to the other possible couples. Ties are broken by picking randomly. This coupling process continues until all N couples are generated, since each region has N subsets (lines 16–18). Then each couple is deemed as a subset, and thus two neighboring regions are merged into a larger region (line 19).

Algorithm 3 Minimum Spanning Tree-Based Grouping Algorithm (MSTBG)

1: **Input:**
   $\{s_i\}_{i=1}^{n}$: the set of sensor nodes
2: $T \leftarrow$ the tree composed by two nearest nodes.
3: group the two nodes into two arbitrary subsets.
4: **repeat**
5: $s \leftarrow$ the nearest node to the tree among all the nodes that are not in the tree.
6: calculate the distance between s and each subset in the tree
7: $S \leftarrow$ the farthest subset (ties are arbitrarily broken)
8: group s to S, and add s to T
9: **until** all nodes are in T

Similarly, MSTBG constructs a minimum spanning tree (MST) of the network incrementally, and groups each newly-joining node to its farthest subset in the tree.[1] Thus MSTBG tries to group nodes that are near each other to different subsets so as to avoid the close-gathering of the nodes in the same subset. The algorithm is illustrated in Algorithm 3.

MSTBG 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 (lines 2–3). Select a node which is the nearest to the tree among all the nodes that are not in the tree (line 5). Group this node to the subset which is the farthest to this node. Then add this node to the tree (lines 6–8). 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 works in building a minimum spanning tree (MST) [14]. This is why we call this algorithm an MST-based grouping algorithm.

4.3 SNRP: a distributed and localized sensor network reconfiguration protocol

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., [6]) can be applied to decentralize MSTBG, global information is still inevitably required in constructing an MST [10]. 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 instead of constructing an MST of the whole network, in SNRP, each node builds its local MST of its neighborhood graph (i.e., the graph consisting of the node and its one-hop neighbors). The node then groups 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 (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

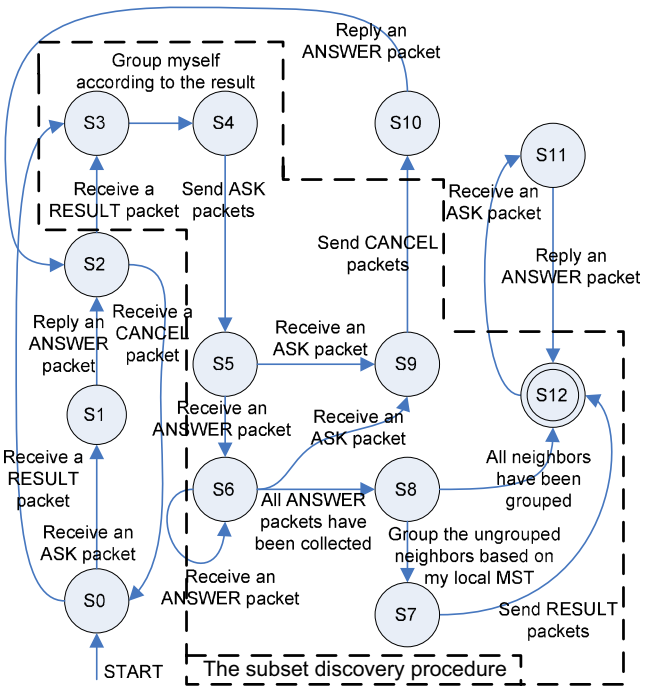

Fig. 1 The finite state machine of SNRP

[1]The distance between a node and a subset is defined the distance between the node and its nearest node in the subset.

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 send 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 the node to be 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).

5 Performance study

To study the effectiveness of our algorithms in solving the sensor network reconfiguration problem, we customize a sensor network simulator. Detailed settings of the simulation networks are shown in Table 1. β, δ and ϵ in the table are parameters of the probabilistic sensing model [18] in which if an event is L meters away from a sensor, the sensor can detect the event with probability p that satisfies:

$$p = \begin{cases} \frac{\delta}{(L/\epsilon+1)^{\beta}} & \text{if } L \le R_s, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

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 β in its way from the event location to the sensor location. This is a realistic consideration.

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.

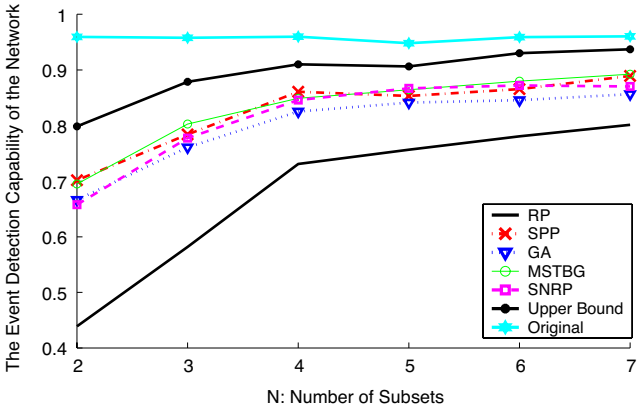

Fig. 2 EDC as a function of N (Node Number = 100)

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:

$$1 - (1 - P_{all})^{\frac{N-1}{N}} \tag{11}$$

where P_{all} is the EDC of the entire network when no subset is off. This is, however, a *non-achievable* upper-bound, 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), 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, 3, and 4 show the EDC of the

Table 1 Simulation Settings

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

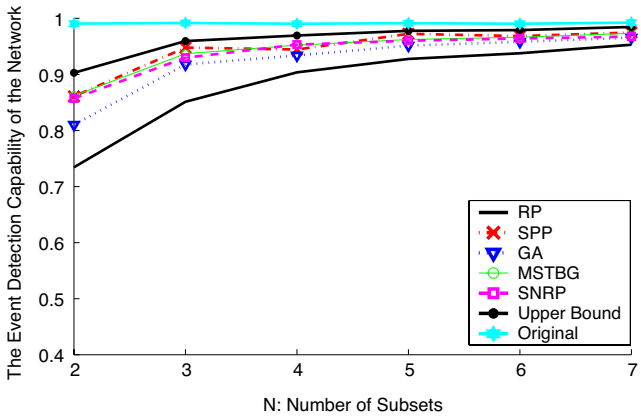

Fig. 3 EDC as a function of N (Node Number = 150)

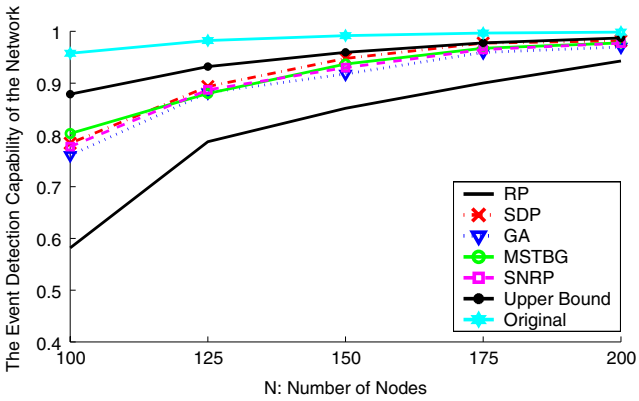

Fig. 5 EDC as a function of node number

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 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 40 m to 80 m (i.e., from one time to two times of the sensing range). Figure 6 demonstrates

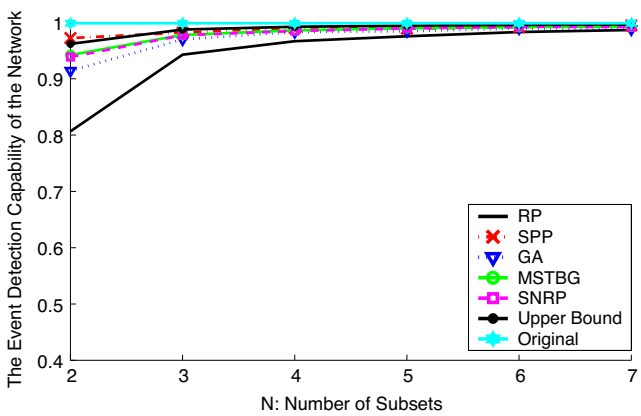

Fig. 4 EDC as a function of N (Node Number = 200)

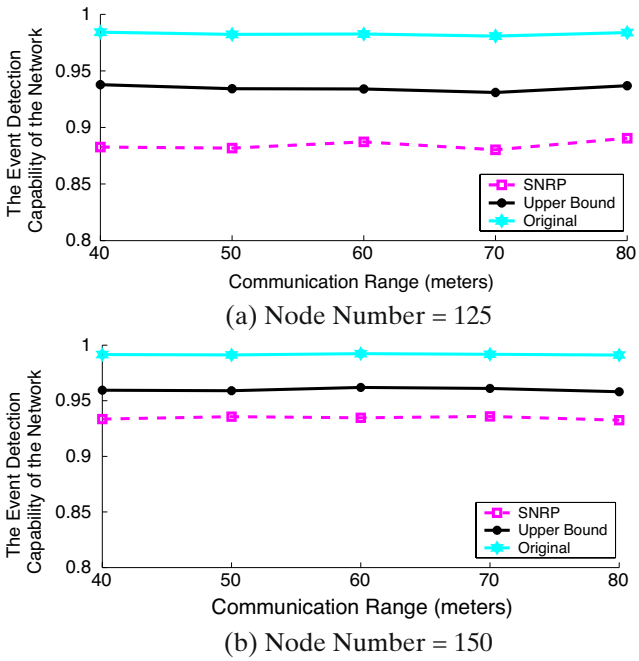

Fig. 6 EDC as a function of communication range

the experimental 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 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, works very well as such a specifically-tailored design does not degrade the resulting performance much.

6 Conclusions

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 in various performance evaluations.

Appendix: Proof of Lemma 1

First, this problem is in NP: Given a partition scheme, a nondeterministic algorithm only needs to calculate the event detection capability (EDC) of each division so as to get the EDC 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* [7] to the decision version of the simplified 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:

$$1 - d_{ik} p_{ij} = 2^{-d_{ik} q_{ij}}. \tag{12}$$

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:

$$\begin{aligned} P' - u &= \min_{\forall k} \left\{ \min_{\forall j} \left[1 - \prod_{i=1}^{n} (1 - d_{ik} p_{ij}) \right] \right\} - u \\ &= \min_{\forall k} \left[\min_{\forall j} (1 - 2^{-\sum_{i=1}^n d_{ik} q_{ij}}) \right] - u \\ &= \min_{\forall k} \left(1 - 2^{-\sum_{i=1}^n d_{ik} q_i} \right) - u \\ &= 2^{-\frac{\sum_{i=1}^n q_i}{2}} - 2^{-\left[\min_{\forall k} (\sum_{i=1}^n d_{ik} q_i) \right]}. \end{aligned} \tag{13}$$

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

$$\begin{aligned} &\min_{\forall k} \left(\sum_{i=1}^{n} d_{ik} q_i \right) \geq \frac{\sum_{i=1}^{n} q_i}{2} \\ &\Rightarrow \begin{cases} \sum_{i=1}^{n} d_{i1} q_i \geq \frac{\sum_{i=1}^n q_i}{2} \\ \sum_{i=1}^{n} d_{i2} q_i = \sum_{i=1}^{n} (1 - d_{i1}) q_i \geq \frac{\sum_{i=1}^n q_i}{2} \end{cases} \Rightarrow \\ &\frac{\sum_{i=1}^n q_i}{2} \geq \sum_{i=1}^{n} d_{i1} q_i \geq \frac{\sum_{i=1}^n q_i}{2} \Rightarrow \sum_{i=1}^{n} d_{i1} q_i = \frac{\sum_{i=1}^n q_i}{2}. \end{aligned} \tag{14}$$

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 simplified sensor network reconfiguration problem so that:

$$\begin{aligned} &\begin{cases} \sum_{i=1}^{n} d_{i1} q_i = \frac{\sum_{i=1}^n q_i}{2} \\ \sum_{i=1}^{n} d_{i2} q_i = \sum_{i=1}^{n} (1 - d_{i1}) q_i = \frac{\sum_{i=1}^n q_i}{2} \end{cases} \\ &\Rightarrow \min_{\forall k} \left(\sum_{i=1}^{n} d_{ik} q_i \right) = \frac{\sum_{i=1}^n q_i}{2}. \end{aligned} \tag{15}$$

According to Eq. 13, $P' = u$. Therefore, the answer to the decision version of the simplified 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 simplified sensor network reconfiguration problem is both NP-Hard and NP. Then it is NP-Complete. The lemma is proved.

References

1. Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E (2002) A survey on wireless sensor networks. IEEE Commun Mag 40(8):102–114
2. Cao Q, Stankovic J (2008) An in-field-maintenance framework for wireless sensor networks. In: Proc of the IEEE DCOSS, Santorini Island, 11–14 June 2008
3. Cardei M, Du D (2005) Improving wireless sensor network lifetime through power aware organization. ACM J Wirel Netw 11(3):333–340
4. Cardei M, MacCallum D, Cheng X, Min M, Jia X, Li D, Du DZ (2002) Wireless sensor networks with energy efficient organization. J Interconnect Netw 3(3–4):213–229
5. Chen J, Kher S, Somani AK (2006) Distributed fault detection of wireless sensor networks. In: Proc of the ACM workshop on dependability issues in wireless ad hoc networks and sensor networks, Los Angeles, September 2006
6. Gallager PG, Humblet PA, Spira PM (1983) A distributed algorithm for minimum-weight spanning tree. ACM Trans Program Lang Syst 5(5):66–77
7. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W H Freeman, New York
8. Hammersley JM (1960) Monte-Carlo methods for solving multivariable problems. Ann N Y Acad Sci 86:844–874
9. Levis P (2008) Tinyos programming manual. http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf
10. Li XY, Wang Y, Wan PJ, Song WZ, Frieder O (2004) Localized low-weight graph and its applications in wireless ad hoc networks. In: Proc of the 2004 IEEE INFOCOM, Hong Kong, 7–11 March 2004
11. Liu B, Towsley D (2004) A study of the coverage of large-scale sensor networks. In: Proc of the IEEE MASS, Fort Lauderdale, 24–27 October 2004
12. Mainwaring A, Polastre J, Szewczyk R, Culler D, Anderson J (2002) Wireless sensor networks for habitat monitoring. In: Proc. of the ACM international workshop on wireless sensor networks and applications, Atlanta, September 2002
13. Marron PJ, Lachenmann A, Minder D, Gauger M, Saukh O, Rothermel K (2005) Management and configuration issues for sensor networks. Int J Netw Manage 15(4):235–253
14. Prim RC (2001) Shortest connection networks and some generalisations. Bell Syst Tech J 36:1389–1401
15. Slijepcevic S, Potkonjak M (2001) Power efficient organization of wireless sensor networks. In: Proc. of the IEEE International Conference on Communications (ICC'01), vol. 2, Helsinki, June 2001
16. Wang Q, Zhu Y, Cheng L (2006) Reprogramming wireless sensor networks: challenges and approaches. IEEE Netw 20:48–55
17. Zhou Y, Lyu MR, Liu J (2008) A sensor-grouping mechanism for field-coverage wireless sensor networks. In: Proc of the IEEE ICC, Beijing, 19–23 May 2008
18. Zhou Y, Yang H, Lyu MR, Ngai ECH (2006) A point-distribution index and its application to sensor-grouping in wireless sensor networks. In: Proc of IWCMC, Vancouver, 3–6 July 2006

Yangfan Zhou is a PhD student in the Computer Science and Engineering Department at The Chinese University of Hong Kong. He received the B.Sc. (2000) degree in electronics from Peking University and the M.Phil (2005) degree in computer science and engineering from the Chinese University of Hong Kong. He also worked in industrial area as hardware engineer and later software engineer in China from 2000 to 2003. His research interests are in wireless ad hoc and sensor networks.

Michael R. Lyu received the B.S. (1981) in electrical engineering from National Taiwan University, the M.S. (1985) in computer engineering from University of California, Santa Barbara, and the Ph.D. (1988) in computer science from University of California, Los Angeles. He is a Professor in the Computer Science and Engineering Department of the Chinese University of Hong Kong. He worked at the Jet Propulsion Laboratory, Bellcore, and Bell Labs; and taught at the University of Iowa. His research interests include software reliability engineering, software fault tolerance, distributed systems, image and video processing, multimedia technologies, and mobile networks. He has published over 200 papers in these areas. He has participated in more than 30 industrial projects, and helped to develop many commercial systems and software tools. Professor Lyu was frequently invited as a keynote or tutorial speaker to conferences and workshops in U.S., Europe, and Asia. He initiated the International Symposium on Software Reliability Engineering (ISSRE), and was Program Chair for ISSRE'1996, Program Co-Chair for WWW10 and SRDS'2005, and General Chair for ISSRE'2001 and PRDC'2005. He also received Best Paper Awards in ISSRE'98 and in ISSRE'2003. He is the editor-in-chief for two book volumes: Software Fault Tolerance (Wiley, 1995), and the Handbook of Software Reliability Engineering (IEEE and McGraw-Hill, 1996). He has been an Associate Editor of IEEE Transactions on Reliability, IEEE Transactions on Knowledge and Data Engineering, and Journal of Information Science and Engineering. Professor Lyu was elected to IEEE Fellow (2004) and AAAS Fellow (2007) for his contributions to software reliability engineering and software fault tolerance. He was also named Croucher Senior Research Fellow in 2008.

Jiangchuan Liu received the BEng degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the PhD degree from The Hong Kong University of Science and Technology in 2003, both in computer science. He was a recipient of Microsoft Research Fellowship (2000), a recipient of Hong Kong Young Scientist Award (2003), and a co-inventor of one European patent and two US patents. He co-authored the Best Student Paper of IWQoS'08. He is currently an Assistant Professor in the School of Computing Science, Simon Fraser University, British Columbia, Canada, and was an Assistant Professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong from 2003 to 2004. His research interests include multimedia systems and networks, wireless ad hoc and sensor networks, and peer-to-peer and overlay networks. He is an Associate Editor of IEEE Transactions on Multimedia, and an editor of IEEE Communications Surveys and Tutorials. He is a Senior Member of IEEE and a member of Sigma Xi.