# Dynamic Coverage and Connectivity Maintenance Algorithms for Wireless Sensor Networks

Prasan Kumar Sahoo
Dept. of Information Management
Vanung University
Chungli, Taiwan, 32061, R.O.C.
Email: pksahoo@msa.vnu.edu.tw

Jang-Ping Sheu, Wei-Shin Lin
Dept. of Computer Science
and Information Engineering
National Central University
Chungli, Taiwan, 32001, R.O.C.
Email: {sheujp, puff}@axp1.csie.ncu.edu.tw

***Abstract*— In wireless sensor network, accidental death of nodes in predictable or unpredictable way may cause coverage and connectivity problems of the original network. In this paper, potential approaches to maintain the network in the post deployment scenario is proposed that lets the sensors work alternatively by identifying the redundant sensing regions in the dense networks under the assumption that the transmission range $(R_c)$ is equal to the sensing range $(R_s)$ or $< 2R_s$ of a node. The proposed coverage and connectivity maintenance algorithms decide which neighbors of a dead node to migrate, and to what distance, so that the loss of coverage and connectivity can be repaired with low mobility of the nodes. In this work, decision and movement of the nodes are completely autonomous and involve movement of only one-hop neighbors of a dead node to minimize energy consumption due to mobility. Performance analysis of the algorithms show that average mobility distance of the nodes is very small and energy consumption is very less by maintaining the coverage and connectivity.**

## I. Introduction

Wireless Sensor Network (WSN) is a special type of ad-hoc network, where wireless devices dynamically form the network without help of any infrastructure. It consists of hundreds to thousands of low-cost sensing nodes, which are capable of performing limited sensing and communication tasks and can be classified into static and mobile nodes. Recently, several researchers have investigated techniques of mobile sensors to obtain better solution for many issues. Several experiments such as MICAbot [7], Mobile Robot [8] and Robomote [9] have been done, which provide convenient platforms for investigating related algorithms and applications of mobile sensor networks. Due to mobility of sensor nodes, mobile sensors can change their position depending on the requirement of the missions. When sensors are deployed in a disaster environment, where human interference is not possible, we need mobile sensors to accomplish the tasks such as coverage and connectivity compensation, location assignment and node replacement. In a post deployment scenario, it is possible that some nodes over certain region are destroyed due to intrusion, explosion or due to environmental factors like heat, vibration, failure of electronic components or software bugs in the network. In another scenario, power sources of the nodes may lead death of the nodes, thereby affecting the coverage and connectivity of the original network. Hence, it is essential to reconfigure the network in order to maintain the connectivity and coverage and to avoid the network partitions.

In [1], the authors present a distributed algorithm for the coordinated coverage fidelity (Co-Fi) maintenance in sensor networks, where mobile nodes are used to repair the coverage loss of the area being monitored by it. In their proposal, the dying node notifies the network of its death, which is not practical for some factors causing death such as program failure and malicious damage. In their work, they consider that $R_c \leq 2R_s$. A Dynamic Coverage Maintenance (DCM) scheme is proposed in [2], which exploits the limited mobility of the nodes. Considering transmission range is twice of the coverage range, i.e. $(R_c = 2R_s)$, the paper proposes a set of DCM schemes, which can be executed on individual sensors and the proposed algorithms decide which neighbors to migrate, and to what distance. In [3], a potential field based approach is proposed for the self-deployment of mobile sensor network. The nodes only use their sensed information in making the decision to move, making it a cost effective solution to the coverage problem. In that work extensive simulation or experimental studies have not been conducted to test the sensitivity of the changes in transmission and sensing ranges and different network sizes. In [4], Wang et. al. have used Voronoi diagrams to discover the existence of coverage holes and a sensor node compares its sensing disk with the area of its Voronoi polygon to estimate any local coverage hole. They propose three distributed self-deployment algorithms: Vector based (VEC), Voronoi based (VOR) and Minimax algorithm.

Though some of the literature discuss about the coverage or connectivity maintenance issues of WSN, to the best of our knowledge, most of the work consider that transmission range is twice of the sensing range and none of them consider the maintenance of both coverage and connectivity for the mobile sensors. Since, communication is the main consumer [5] of energy resource, the most different assumption in our work is that transmission range is equal to the sensing range and our algorithms can also be extended to maintain the connectivity and coverage of the network for $R_c < 2R_s$. The main contributions of our work can be summarized as follows.

- We present completely distributed COverage and COnnectivity maintenance algorithms $(CoCo)$, considering $R_s \leq R_c < 2R_s$.

1-4244-0614-5/07/$20.00 ©2007 IEEE.

- We design algorithms to maintain both connectivity and coverage problems in WSN due to accidental death of the nodes either in predicted or unpredicted manner.
- We use low mobility of the nodes that only affects to the one-hop neighbors of a dead node, without disturbing the existing communication and coverage system of the network. Our algorithms can verify the disturbance in connectivity and coverage among nodes dynamically and can maintain the network integrity.

The rest part of the paper is organized as follows. Section II introduces the system model with some definitions related to our work. Section III describes our low mobility coverage and connectivity maintenance algorithms. Simulation work, performance analysis of our algorithms and comparison of our results with some related work are done in Section IV. Concluding remarks are made in Section V of the paper.

## II. System Model

We consider a wireless sensor network, where sensors are distributed randomly and densely with higher degree of neighbors. Each node is aware of its own location and coordinates of the boundaries of the deployed region through location services [6]. Each node is equipped with digital compass [10] and is capable of moving, as mentioned in [7, 8, 9]. It is assumed that each node has a unique ID and is equipped with homogeneous sensing and transmission ability, where $R_s \leq R_c < 2R_s$. For the whole network, either $R_c = R_s$ or $R_c < 2R_s$, the sensing and transmission range are uniform for all the nodes. At the time of deployment, it is assumed that the network is connected and the whole deployed region is fully covered. Each node knows its initial energy level and can keep track of its energy expenditure so that a node can predict its own death in advance.

### A. Definitions

**Sensing and Connecting Neighbors:** Two nodes $A$ and $B$ are said to be connecting neighbors, if their Euclidean distance $|AB| \leq R_c$ and said to be sensing neighbors if $|AB| \leq 2R_s$. From our definition and assumption, it is obvious that the connecting neighbors are always the one-hop neighbors; where as sensing neighbors may be within multihops of a node.

**Grid Length:** In our work, the whole rectangular deployed region is virtually divided into certain grids and grid length ( $L_g$) is defined as the length of each edge of the grids. To maintain the connectivity and coverage, since we need at least two nodes within each grid, $L_g$ must guarantee our need, which can be estimated as $(\sqrt{(R_c^2 - (R_c - R_s)^2)} + R_s + R_c)$, when $R_s \leq R_c < 2R_s$.

**Grid ID:** In our work, we assume that each node must belong to at most one grid, whose ID is an ordered pair and can be estimated as follows.

$GridID$ of $i$th node = $(\lfloor \frac{x_i - x_0}{L_g} \rfloor, \lfloor \frac{y_i - y_0}{L_g} \rfloor)$,

where $(x_i, y_i)$ is coordinate of the $i$th node, $L_g$ is length of each grid edge and $(x_0, y_0)$ is origin of the deployed region, as shown in Fig. 1. It is to be noted that in our protocol, each node keeps location information of the nodes in its own grid and the nodes within its surrounding eight grids. For example,

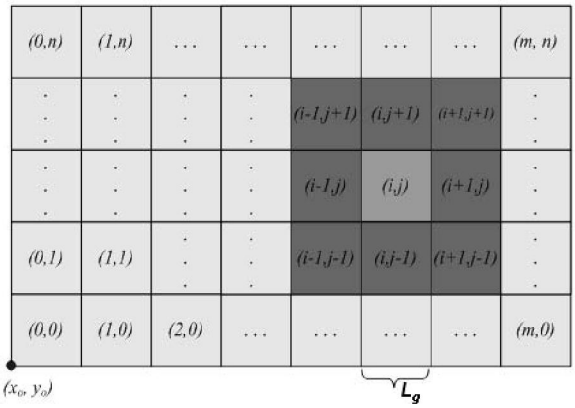


Fig. 1. Division of the network into grid that describes the grid identification

as shown in Fig. 1, nodes having grid id $(i, j)$ know location information of the nodes located within the grids having grid id $(i-1, j-1), (i-1, j), (i-1, j+1), (i, j-1), (i, j), (i, j+1), (i+1, j-1), (i+1, j)$, and $(i+1, j+1)$.

**Critical Sensing Point (CSP):** Critical sensing point of a node is the point of intersection of the sensing region of that node with sensing region of its sensing neighbors, as explained below.

Let, $K$ be the set of sensing neighbors of node $A$, whose sensing regions intersect with itself and $|K| = l$. Let, $a$ be the location of node $A$ and $k_i$ be the location of $K_i$th node, where $K_i \in K$, $\forall i = 1, 2, .., l$ and $\overline{ak_i} \leq 2R_s$. $S$ be the set of points of intersection formed by nodes $K_m$ and $K_n$, where $K_m, K_n \in K$, for $m \neq n$, then a point $p \in S$ is said to be a critical sensing point $(CSP)$ for node $A$, if $p$ satisfies both of the following conditions:

1. $\overline{pa} \leq R_s$
2. $\forall K_i \in K$, where $i = 1, 2, .., l$, $\overline{pk_i} \geq R_s$

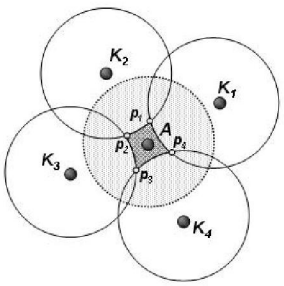


Fig. 2. The points $p_1$, $p_2$, $p_3$, and $p_4$, which enclose the one-cover shaded region are the critical sensing points of node $A$.

As shown in Fig. 2, $p_1$, $p_2$, $p_3$, and $p_4$ are the *CSPs* of node $A$. The shaded region formed by nodes $A$ and $K_1$, $K_2$, $K_3$, and $K_4$ and enclosed by these *CSPs* is always one-cover. It is to be noted that each node keeps information of the nodes within its surrounding eight grids and then finds its critical sensing points.

**Disjoint Transmission Set (DTS):** Each node in the network classifies its one-hop connecting neighbors into member of the disjoint transmission set *(DTS)*. A connecting neighbor $X$ of node $A$ is said to be a member of *DTS* $D_i$, if any of the conditions is satisfied:
(1) $X$ has a connecting neighbor, which is also in $D_i$
(2) $X$ is the only connecting neighbor of $A$.
(3) $X$ has an identical connecting neighbor with any other members of $D_i$.

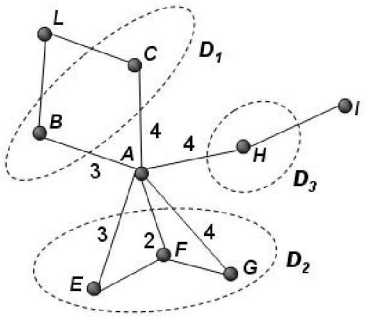


Fig. 3. An example of Disjoint Transmission Set, where $D_1$, $D_2$ and $D_3$ are $DTS$ of node $A$.

As shown in Fig. 3, let us find the *DTS* of node $A$. Since node $B$ and $C$ have an identical neighbor node $L$, which is two-hops away from $A$, only $B$ and $C$ belong to *DTS* $D_1$. In another example, node $E$ and $G$ are the connecting neighbors of $A$. Though, node $F$ is the identical neighbor of nodes $E$ and $G$, it is connecting neighbor of node $A$. So, node $E$, $F$ and $G$ belong to the same *DTS* $D_2$. The single node $H$ is the connecting neighbor of node $A$, without having any identical neighbor. So, it belongs to *DTS* $D_3$.

**Head Node:** Let, $D_i(A)$ be the disjoint transmission set of node $A$, for $i = 1, 2, .., n$. Any node, $N \in D_i(A)$ is said to be a *head node* in its *DTS*, if $N$ is closest to $A$. For example, as shown in Fig. 3, the number along each link represents the Euclidean distance between two nodes. Since, $B, C \in D_1(A)$, and node $B$ is closest to $A$, $B$ is the *head node* in $D_1$. Similarly, node $F$ and node $H$ are the *head nodes* in $D_2$ and $D_3$, respectively.

**Beacon Packet:** In our protocol, beacon packets are broadcast among the one-hop connecting neighbors, which contains the sender's ID, location information and critical sensing points, its connecting neighbor's list and list of the head nodes. The connecting neighbor's list includes the neighbor's ID, location and *DTS* ID, whereas list of head nodes contains the list of head node's location information and available mobility distance (AMD), which is discussed in the next section.

## III. Coverage and Connectivity Maintenance Algorithms

As soon as the nodes are deployed over certain region, each node starts estimating its grid id from its location information. Then, nodes flood their location information and grid id to estimate the critical sensing points, disjoint transmission set and selects the head nodes in each of its *DTS*, as defined in Section 2. If a node in the network is predictably dead or accidentally destroyed, connectivity among the nodes and existing coverage of the network may be disturbed, causing network partitions and deteriorating the coverage. Hence, the network should be maintained with some prior arrangements, taking the available mobility distance (AMD) of the nodes, so that nodes can move immediately as soon as such problem is happened. For this, we assume that beacon packets are transmitted periodically among the one-hop connecting neighbors of each node, as soon as each node knows location information of the nodes in its surrounding eight grids and grid's id. Then, each node starts estimating *AMD* in terms of maximum transmission mobility distance (MTMD) and maximum sensing mobility distance (MSMD), as described in the following sections.

### A. Available Mobility Distance (AMD)

*AMD* is the minimum value between the *MSMD* and *MTMD* and defined as the maximum distance that a node can move without affecting its own connectivity and coverage. Prior to estimating the *AMD*, each node forwards the beacon packet to its connecting neighbors and each receiver starts estimating *MTMD* and *MSMD*, without considering the sender node.

*1) Estimation of MTMD:* The maximum transmission mobility distance *MTMD* of a node is defined as the maximum distance that a node can move from its current position to the position of the sender such that connectivity is preserved between itself and with each of its *head nodes* in each of its *DTS*. Upon receiving the beacon packet, a node (receiver) reclassifies its *head node* list and *DTS*, without considering the sender. It is obvious that the distance between each head node and the receiver must be $\leq R_c$. Then, the receiver node finds points on the line joining to the sender and the receiver such that the point is $R_c$ units away from each of its *head nodes*. Minimum of the distance between the receiver and those points is termed as the *MTMD* of the receiver. The algorithm to estimate the MTMD is given below.

**Algorithm:** *MTMD Estimation*
$Notation$ :
$P$ : node who is going to estimate $MTMD$;
$D_i$: $i$th DTS of $P$; $X$: Set of disjoint transmission sets;
$N_H$: Head node;
$ShortestDistance$: The shortest distance between the sender node $C$ and $N_H$;
$Distance(A, B)$: Distance between any two nodes $A$ and $B$;

$Input$: Sender node $C$, who transmits the beacon packet;
$Output$ : MTMD;

**MTMD Estimation(Node C)**
$MTMD := \infty$;
$X$ :=Disjoint Transmission Set Classify( P.NbrList - $\{C\}$ );
for each set, $D_i \in X$
Shortest Distance from node $C := \infty$;
$N_H := \{\phi\}$;

for each node $M \in D_i$
if $(Distance(C, M) < ShortestDistance)$
$N_H := M$;
end if
end of for loop
find a point, $\omega$ on $\overline{CP}$ such that $Distance(N_H, \omega) = R_c$
if $(Distance(P, \omega) < MTMD)$
$MTMD := Distance(P, \omega)$;
end if
end of for loop
return $MTMD$;

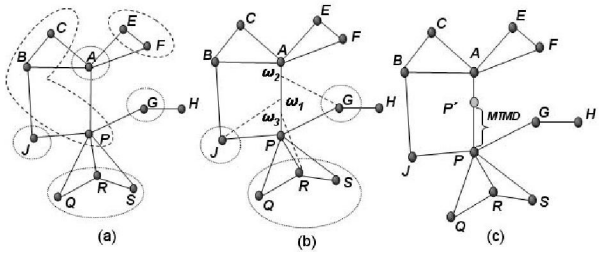


Fig. 4. (a) DTSs of node $A$ are represented by dotted curves and that of $P$ are represented by solid curves. (b) Shortest distance estimation from each of the head nodes of $P$, when $A$ is the sender. (c) Required *MTMD* of node $P$, so that each of its head nodes can be connected, if $A$ is dead and $P$ moves to $P'$.

As shown in Fig. 4(a), node $A$ has two *DTSs*, and node $P$ has four *DTSs*. When node $P$ receives the beacon packet from $A$, it reclassifies three *DTSs* without taking $A$, as shown in Fig. 4(b). To estimate the *MTMD*, $J\omega_1$=$G\omega_2$=$R\omega_3$=$R_c$ are taken such that $\omega_1$, $\omega_2$, and $\omega_3$ are situated on the line $\overline{PA}$. The point $\omega_3$ is considered as the new position for the node $P$ to move towards $A$, as it is $min(P\omega_1, P\omega_2, P\omega_3)$. So, *MTMD* of node $P$ towards node $A$ is estimated as $\overline{P\omega_3}$, which is the maximum distance that $P$ can move so that the link between nodes $J$, $R$ and $G$ are preserved, as shown in Fig. 4(c).

### B. Estimation of MSMD

The maximum sensing mobility distance (MSMD) of a node is defined as the maximum distance that a node can move without introducing any coverage problem in the existing network. The *MSMD* is estimated by a node, which has to move due to accidental death of its connecting neighbor and to ensure that the existing coverage is not affected by its mobility.

As shown in Fig. 5(a), the shaded region is one-covered region and death of node $A$ creates the coverage hole. In another scenario, the shaded region is two-covered region, common to both nodes $B$ and $C$. If nodes $B$ and $C$ move towards $A$ independently, coverage hole is created, as shown by the shaded region in Fig. 5(b). Hence, prior to their movement, each node should first estimate their *MSMD*, as given in the following algorithm.

**Algorithm:***Estimation of MSMD*
$Notation$ : $P$: Node, who is going to estimate $MSMD$;
$A.CSP$ : Set of critical sensing points of node $A$;

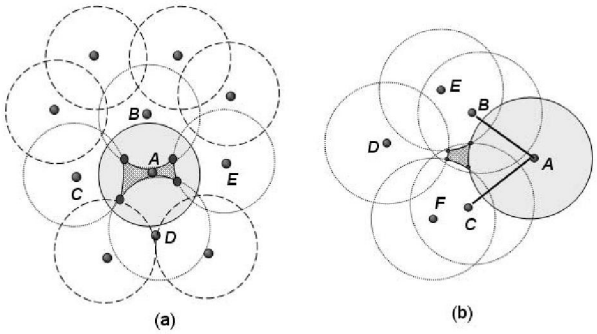


Fig. 5. (a) Death of node $A$ creates coverage problem, represented by the deep gray region. (b) The gray region is two-covered region covered by nodes $B$ and $C$ and creates coverage problem due to their independent mobility.

$Distance(B, C)$ : Distance between nodes $B$ and $C$;

$Input$ : Node $P$;
$Output$ : MSMD;

**MSMD Estimation**$(NodeP)$
$MSMD := \infty$;
for each critical sensing point $p \in A.CSP$
find a point, $\gamma$ on $\overline{PA}$, such that $Distance(p, \gamma) = R_s$;
if $(Distance(A, \gamma) < MSMD)$
$MSMD := Distance(A, \gamma)$;
end if
end of for loop
return $MSMD$;

### C. Maintenance Algorithms

As mentioned earlier, beacon packets are exchanged periodically among the one-hop neighbors of each node. If a node does not receive any beacon packet consecutively for three times (In our simulation, we have taken three times as the limitation, which can be user define) from any of its connecting neighbors, it assumes the accidental death of that neighbor. In case of death of a node due to energy scarcity, a node predicts about its death to its connecting neighbors in advance, when it exchanges the beacon packet. From the information given in the beacon packet, a node can infer if there will be any connectivity or coverage problem due to death of that connecting neighbor. So, we propose here the COnnectivity and COverage maintenance algorithms (CoCo), as follows.

*1) Connectivity Maintenance:* It is assumed that each node stores the information given in the beacon packet, until it receives the next one. In case of predictive or non-predictive death of a connecting neighbor, a node scans the information contained in the last beacon packet of that dead node or the node going to die. If that connecting neighbor did not declare any critical sensing point, instead it declares more than one disjoint transmission set, it implies that it would create communication problem with its connecting neighbors. Hence, the main goal of our connectivity maintenance algorithm is to reduce the distance among the head nodes of different *DTS*

so that the network can be reconnected. We use the minimum spanning tree concept to achieve the goal, as described below.

**Step 1:** Let us form a complete graph taking *head nodes* of all *DTS* of the dead nodes as vertices and Euclidean distance between two nodes as weight of each edge.

**Step 2:** Use Kruskal's algorithm to find the minimum spanning tree, taking dead node as the root. All nodes, representing vertices of the graph should move towards the root such that all edges of the minimum spanning tree are reduced to the communication radius $(R_c)$.

The required mobility distance(RMD) for any *head node* $A$ with respect to another *head node* $B$ and vise versa is defined as: $R_c^2 = (\overline{AD} - d)^2 + (\overline{BD} - d)^2 - 2(\overline{AD} - d)(\overline{BD} - d)cos\beta$,

where, $d$ is *RMD* of a node, $\beta$ is angle between node $A$ and $B$ with respect to the dead node $D$. It is to be noted that one head node can get location information of another one from the last beacon packet sent by the dead node, though the head nodes are not the connecting neighbors. A node may calculate several required *RMD*, but chooses the longest one as its final *RMD*.

As discussed earlier, since, *AMD* of each connecting neighbor towards the dead node is known and each *head node* estimates the *RMD* of another one, it can know if its neighbor sharing the same edge with it has enough *AMD* or not. If all connecting neighbors of the dead node have sufficient *AMD*, they move towards the dead node for the required distance and maintain the connectivity, otherwise a node goes for the *Cascading Mobility*, as discussed later. The algorithm to estimate RMD is given below.

**Algorithm:** *Estimation of RMD for the Connectivity*
$Notation : AMD(i)$ : AMD of node $i$;
$RMD(i)$ : RMD of node $i$;
$\beta(i, j, D)$ : Angle between node $i$ and $j$ with respect to $D$;
$Distance(i, j)$ : Euclidean distance between node $i$ and $j$;
$D$ : dead node;
$R_c$: Communication range of each node;
$d$ : RMD of node $i$ with respect to node $j$;
$d'$ : new RMD of node $i$ with respect to node $j$, in case of insufficient $AMD(j)$;

$Input$: Location information of nodes $i$ and $j$ and $\beta(i, j, D)$;
$Output : RMD$;

**RMD Estimation( )**
$boolean$ : Cascading Mobility = false;
for each head node $i$ of dead node $D$;
for each node $j$ who has edge connect to node $i$;
find $d$ such that
$R_c^2 = (Distance(i, D) - d)^2 + (Distance(j, D) - d)^2 - 2(Distance(i, D) - d)(Distance(j, D) - d)\cos\beta(i, j, D)$;
if $((d \leq AMD(i))$ and $(d \leq AMD(j)))$
$RMD(i) = d$;
$RMD(j) = d$;
else if $((d \geq AMD(j))$ and $(d \leq AMD(i)))$
find $d'$, such that
$R_c^2 = (Distance(i, D) - d')^2 + (Distance(j, D) - AMD(j))^2 - 2(Distance(i, D) - d')(Distance(j, D) - AMD(j))\cos\beta(i, j, D)$;
if $(d' \leq AMD(i))$
$RMD(i) := d'$;
$RMD(j) := AMD(j)$;
else
$CascadingMobility := true$;
else if $((d \geq AMD(j))$ and $(d \geq AMD(i)))$
$CascadingMobility := true$;
end of $for$ loop
end of another $for$ loop
if (Cascading Mobility = true)
execute $CascadingMobility$;

**A. Cascading Mobility:** A head node goes for the Cascading Mobility, if its $AMD$ is zero. While moving to compensate the connectivity of the dead node, a *head node* itself needs to be connected with its neighboring *head nodes*. However, if its $AMD = 0$, it gives up the minimum spanning tree procedure and turn to cascading mobility procedure. In this procedure, we assume that the head node going for the cascading mobility procedure is considered as a *leader node*, which resets its *RMD* as the distance from its location to the dead node. The *leader node* sends a mobility request $(MOB\_REQ)$ packet to its head nodes in each of its disjoint transmission sets and to its sensing neighbors, which contains its mobility direction and *RMD*. It is to be noted that each node in the network has its own *DTS*, *CSP* and *head node sets*. The sensing and connecting neighbors of the *leader node*, in each of its *DTS* receive the $MOB\_REQ$ packet and move, as described in form of examples shown in Fig. 6 and Fig. 7.

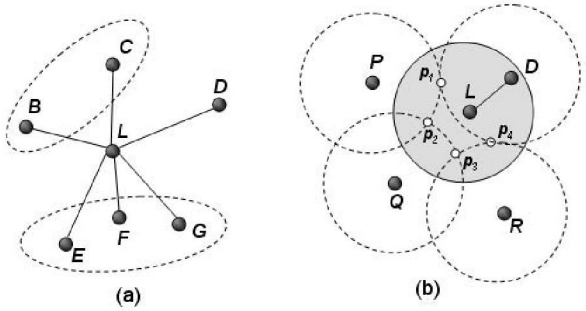


Fig. 6. (a) Disjoint Transmission Sets of *leader node* $L$, among its connecting neighbors. $L$ has to move to the position of the dead node $D$. (b) Critical Sensing Points formed by sensing neighbors of *leader node* $L$. The region enclosed by CSPs $p_1$, $p_2$, $p_3$, and $p_4$ should be covered, if $L$ moves to the position of dead node $D$.

As shown in Fig. 6(a) and 6(b), the *leader node* $L$ has to lead the cascading mobility. Nodes $B$, $C$, $E$, $F$ and $G$ are connecting neighbors of $L$, where $B$ and $C$ belong to one of its *DTS* and $E$, $F$, $G$ are members of another *DTS* of $L$, as shown in Fig. 6(a). As shown in Fig. 6(b), node $P$, $Q$, $R$ and $D$ are the sensing neighbors of $L$ and the region enclosed by

the points $p_1$, $p_2$, $p_3$, and $p_4$ is uncovered, if $L$ moves towards $D$. Hence, objective of the cascading mobility is to decide the magnitude and direction of the mobility of head nodes of the leader node $L$ and to follow it, when it moves towards the dead node so that connectivity and coverage can still be maintained.

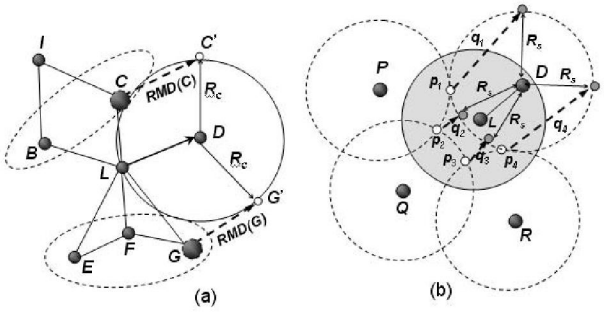


Fig. 7. (a) Node $G$ and $C$ are head nodes in different *DTS* of leader node $L$. So, both $G$ and $C$ move parallel to the mobility direction of $L$ with $RMD(G)$ and $RMD(C)$, respectively. (b) $P$, $Q$ and $R$ are the sensing neighbors of $L$, which move parallel to the mobility direction of $L$ such that the critical sensing points $p_1$, $p_2$, $p_3$, $p_4$ should be displaced with $q_1$, $q_2$, $q_3$, $q_4$ units to maintain the connectivity and to avoid the coverage hole.

All the sensing and connecting neighbors of *leader node* $L$ receive the $(MOB\_REQ)$ packet, when $L$ forwards it. In Fig. 7(a), the head nodes $G$ and $C$ move parallel to the mobility direction of $L$ and the *RMD* for node $G$ is $RMD(G)$ units such that the distance from the new position of node $G$ to the new position of leader node $L$ should be $R_c$ units. Similarly, nodes $C$ moves $RMD(C)$ units to $C^{'}$, so that distance between $C^{'}$ and $L$ should be $R_c$ units. Upon receiving the $MOB\_REQ$ packet, a node estimates its *AMD* based on the mobility direction and *RMD* inside the $MOB\_REQ$ packet and sends back a $MOB\_RPY$ packet to the leader node, indicating its $AMD$. The node has to move, if it has sufficient *AMD* i.e. more than the *RMD*, otherwise it sends the $MOB\_RPY$ packet with zero *AMD*. If $AMD = 0$, the sink is requested to redeploy new nodes in the network, as connectivity problem cannot be solved by any means.

As shown in Fig. 7(b), since node $P$, $Q$ and $R$ have to move parallel to the mobility direction of the leader node $L$, consider a line starting from $p_1$, whose slope is same as the mobility direction of $L$. Then, find a point on the line such that the distance from that point to the destination of node $L$ should be equal to $R_s$. So, the distance from $p_1$ to the new point is termed as the $RMD$ for $P$, which is $q_1$ units, as shown in Fig. 7(b). Similarly, $q_2$ is estimated, taking point $p_2$ and $max(q_1, q_2)$ is the *RMD* for node $P$. This procedure is repeated to estimate the RMD for $Q$ and $R$, estimating $q_2$, $q_3$ and $q_4$ and finding the maximum value among them for each related nodes.

**B. Neighbor grids updating** It is to be noted that we divide the whole network into certain grids in order to limit the amount of information stored in each node. Hence, each node stores information about the nodes of its own and neighboring eight grids and grid length can be estimated as per our definition for $R_c = R_s$ or $R_c < 2R_s$. The neighbor grid information will not change, if a node stays in its original grid after its mobility. However, if a node migrates to another neighboring grid, the neighbor grid information should be updated. For example, if node $A$ migrates to another grid, it informs to the nodes in its neighboring grids with a $Grid\_Update$ message, which contains its destination grid's ID. Nodes in the overlapping grids, which are still neighboring grids of node $A$, keep $S$'s information, while nodes in other neighboring grids of $A$ eliminates it from their neighboring grid node's list. Out of all nodes in the destination grid of node $A$, node having minimum ID informs about the neighboring girds to $A$. Upon receiving information of those nodes, node $A$ sends a $Grid\_Update$ message to inform about its arrival. Once the neighboring grid information is updated, node $A$ and all the receivers of $Grid\_Update$ message, update their critical sensing points based on the new neighbor grid's information.

### D. Coverage Maintenance

If the last beacon of a node declares some critical sensing points, but it has only one disjoint transmission set, it implies that coverage hole is created by that node after its death. In this case, the head nodes of the dead node try to maintain the coverage hole by estimating the required mobility distance, using the following algorithm.

**Algorithm:** *Coverage Maintenance*
$Notation$ :
$AMD(i)$ : AMD of node $i$;
$RMD(i)$ : RMD of node $i$;
$i.CSP$ : critical sensing point of node $i$;
$Distance(A, B)$ : Distance between node $A$ and node $B$;
$Min\_RMD$ : the minimum RMD value among head nodes;
$Coverage\_Solver$: node who is going to solve coverage problem;
$Farthest\_Distance$: The farthest distance from all the critical sensing point;

$Output$ : $RMD$;

**Coverage RMD Estimation( )**
$Min\_RMD := \infty$;
$Coverage\_Solver := \{\Phi\}$;
for each head node $i$ of dead node $D$
$Farthest\_Distance := 0$;
for each critical point $p$ in $i.CSP$, find a point $\gamma$, on $\overline{iQ}$ such that $Distance(p, \gamma) = R_s$;
if $(Distance(i, \gamma) > Farthest\_Distance)$
$Farthest\_Distance := Distance(i, \gamma)$;
end of $for$ loop
$RMD(i) := Farthesr\_Distance$;
if $(RMD(i) \leq AMD(i)$ and $RMD(i) < Min\_RMD)$
$Coverage\_Solver := i$;
$Min\_RMD := RMD(i)$;
end of $for$ loop

if $(Coverage\_Solver \neq \{\Phi\})$
if $(Coverage\_Solver = self\_id)$
$RMD := Min\_RMD;$
else
$RMD := 0;$
else
Go for $CascadingMobility$

It is to be noted that each *head nodes* of the dead node executes the above algorithm and estimates the RMD and compares with the AMD. Then the node having enough AMD will move, as described in the algorithm. The explanation of algorithm is presented in Fig. 8 as an example.

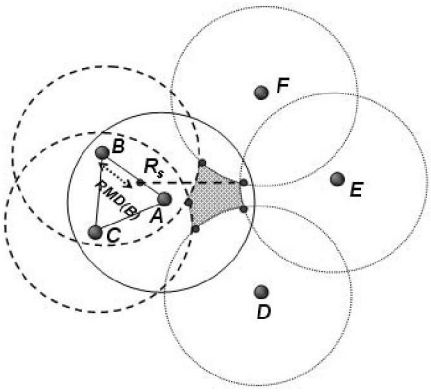


Fig. 8. The farthest critical point of node $A$ to node $B$ is $p$. Find a point $\gamma$ on $\overline{AB}$ such that $\overline{p\gamma} = R_s$ and the RMD of node $B$ is $\overline{\gamma B}$.

## IV. Performance Analysis

### A. Simulation Setup

We have simulated our work using ns-2.29 and taking 800 sensors deployed randomly over an area of $250m \times 250m$. The AODV routing protocol and the TwoRayGround propagation model along with IEEE 802.15.4 CSMA-CA channel access mechanism are considered in our simulation. Initially, each node has a fixed amount of 50J reserved energy and energy cost due to mobility is taken as 1J/m. As per our proposed coverage and connectivity maintenance algorithms, we have simulated them under different transmission and sensing range. The sensing range is fixed and set to be 10m. The traffic data rate is kept as 250Kb and beacon packets are sent in every two seconds. In order to justify our assumption, the nodes are considered to be dead in our simulation in two different scenarios. In the first scenario, a node is dead due to its energy exhaustion, which is assumed, if its energy level reaches to zero. In another scenario, certain nodes are considered as dead by switching off them randomly and abruptly. A node is assumed to be dead, if it does not receive continuously three beacons from its neighbors. Based on our algorithms, our simulations are run for 20 different topologies. We first simulated the residual energy and percentage of alive nodes for different transmission range. Once the packet transmission starts, each node starts sending 1000 packets randomly, each of whose size is 64KB.

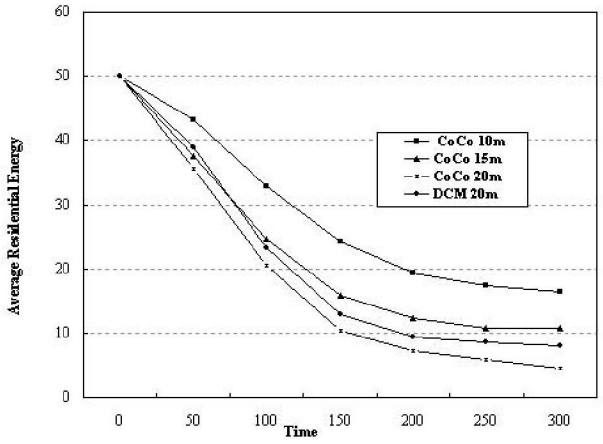


Fig. 9. Average residual energy for different transmission range with fixed value of sensing range i.e. when $R_s = 10m$

In order to simulate the coverage and connectivity maintenance, and average mobility distance, each node is given a unique ID seed and time seed, which are used to generate the ID and next generating ID time, respectively. First, nodes use time seed to generate the next generating time, whose values lies between 4 to 20 seconds. When the generating time expires, each node uses the ID seed to generate a random node ID. If the generated ID of a node is same as its own ID, it sets its residual energy to zero and assumed to be dead. So, nodes in our simulation die in a probabilistic manner. Since, Co-Fi [1], DCM [2] and our algorithms are extremely suitable for the only one node dies at the same time, this arrangement of node lifetime can promise at most one node dies for the average 12 seconds. In our simulation, the life time of flooding message for updating sensing neighbors in Co-Fi[1] is considered as 10 hops and we allow the nodes to send panic message [1] and to decide about a node that should move before the death of each node from a fair point of view. We do not deploy any additional node in our simulation, when cascading mobility is unable to solve the problem. In order to compare the performance of our algorithms with DCM [2], we only consider the condition that $R_c = 2R_s$, as it totally complies with their assumptions.

### B. Simulation Results

In our connectivity and coverage maintenance algorithms (*CoCo*), since each node initially requires a flooding message and we need to attach more information in it, we consume more energy than the *DCM* [2], as shown in Fig. 9, when the transmission range is 20m and sensing range is 10m, i.e. we consider $R_c = 2R_s$. However, in *CoCo*, since the transmission range is allowed to be less than 20m $(R_c < 2R_s)$ or equal to be 10m $(R_c = R_s)$, we find that the average residual energy is increased by reducing the transmission range. From Fig. 9, it is to be noted that *CoCo* outperforms *DCM* [2] in terms of energy consumption, and fulfil the same purpose of maintaining the coverage and connectivity through the low mobility of nodes. As shown in Fig. 10, we have also simulated our algorithms and compares our results with *DCM* [2], in terms of percentage

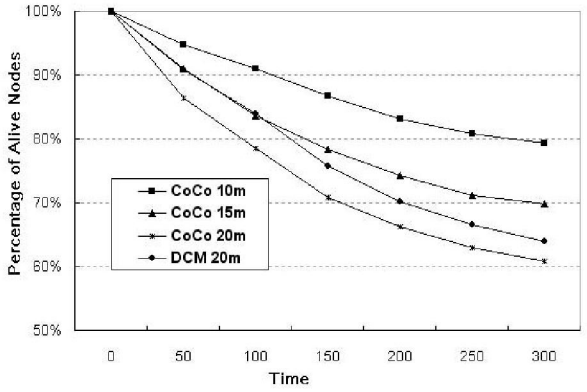


Fig. 10. Percentage of alive nodes with time, for different value of transmission range and fixed value of sensing range i.e. when $R_s = 10m$.

of alive nodes with time for various transmission range. It is to be noted that communication is the main consumer of energy. Since, we have considered the least transmission range, we got the most expected result, which is better than the *DCM*.

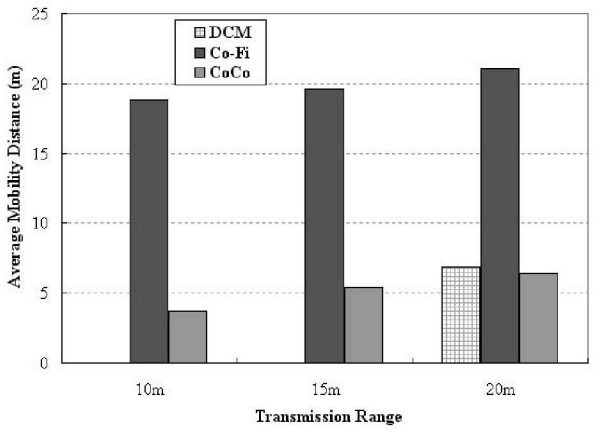


Fig. 11. Average mobility distance vs transmission range, for fixed $R_s$=10m.

In order to verify the performance in terms of average mobility distance of the nodes, we present the performance of *CoCo* in Fig. 11, for different transmission ranges. Since, limited mobility is one of the characteristic in *CoCo*, we compare the average mobility with similar protocols and find that the average mobility of *CoCo* is extremely less than *Co-Fi* [1], which is not a limited mobility model. In our algorithm, we can find that the average mobility distance is increased while the transmission range is increased. This is because of larger the transmission range, longer distance the sensor nodes have to move, when communication problem arises. On the other hand, with larger transmission range, each node could have more *MTMD*, which may increase the value of *AMD*, for which the average mobility distance in our simulation has also increased. *DCM* [2] allows only one node to move after coverage hole is appeared, whereas, in *CoCo* multiple nodes are involved to solve the connectivity problem. Hence, each node has a least average mobility distance in our algorithm

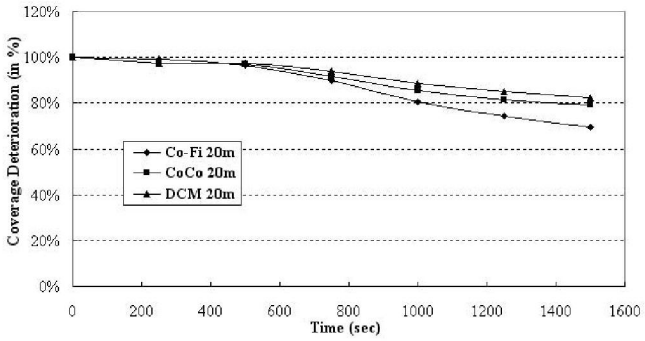


Fig. 12. Percentage of coverage deteriorations for $R_c$=20m and $R_s$=10m.

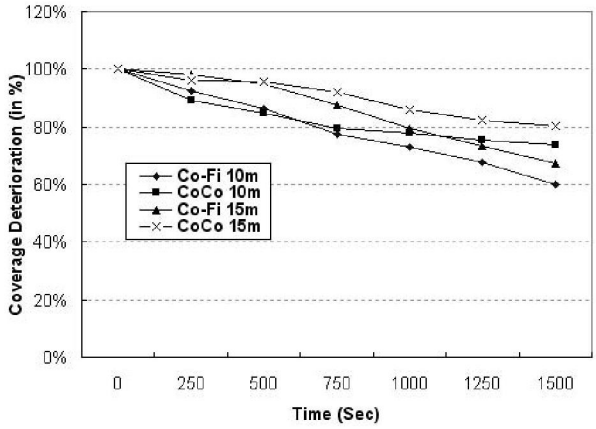


Fig. 13. Percentage of coverage deteriorations for $R_c$=10m, 15m and $R_s$=10m.

and average mobility distance in our work is slightly less than *DCM*, when we simulate our algorithms taking $R_c = 2R_s$.

In order to verify the network coverage deterioration, we have presented our simulation results in Fig. 12 and Fig. 13. In order to get a fair comparison, we have simulated *CoCo* taking $R_c$=20m, as both *Co-Fi* and *DCM* consider $R_c = 2R_s$. As shown in Fig. 12, we can see that our algorithm has better performance than *Co-Fi* for certain period of time i.e. within 500 sec and always outperforms *DCM*. As shown in Fig. 13, *Co-Fi* outperforms *CoCo* for $R_c < 2R_s$ or $R_c = R_s$. It is observed that due to less value of $R_c$, less percentage of region is covered in *CoCo*. Obviously, $R_c$ is correlated to *MTMD*, and due to reduction in $R_c$, the number of disjoint transmission sets of a node is increased, thereby reducing the value of *MTMD* and *AMD*, as well. With less value of *AMD*, there is higher probability that nodes may be unable to solve the coverage or connectivity problems. So, the performance is degraded when range of $R_c$ is reduced in *CoCo*. The better performance of *Co-Fi* over *CoCo* is due to the long average mobility distance of the nodes in *Co-Fi*, as they ask the redundant nodes to replace the dead node and thereby their performance is better than ours. However, the redundancy of nodes in the network may not happen always and no redundant node may be available after several replacements of the dead nodes. Besides, the nodes in the network may die by replacing them time to

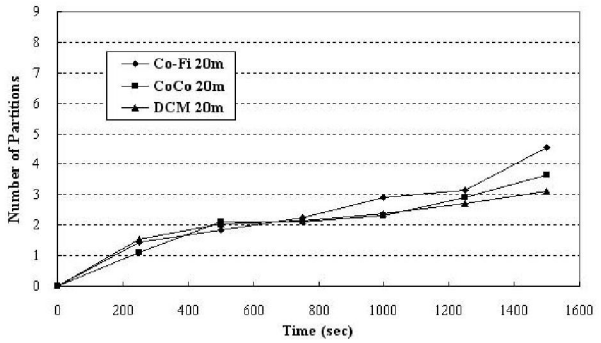


Fig. 14. Possible number of network partitions for $R_c$=20m and $R_s$=10m.

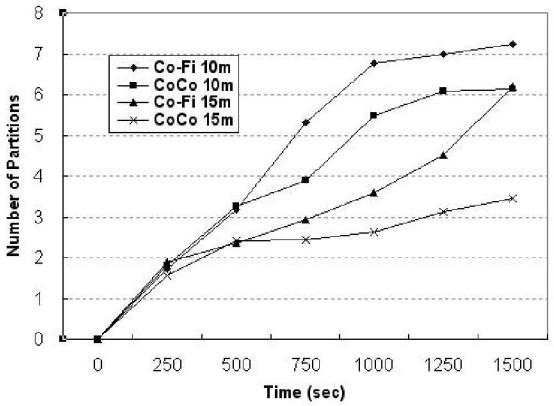


Fig. 15. Possible number of network partitions for $R_c$=10m, 15m and $R_s$=10m.

time and due to their long distance mobility. From this point of view, our algorithm is better than *Co-Fi*, as the average mobility distance in our protocols is minimized.

The performance evaluation of *CoCo* in terms of connectivity maintenance is presented in Fig. 14 and Fig. 15. In Fig. 14, we compare our results with both *Co-Fi* and *DCM*, as we consider $R_c = 2R_s$, whereas in Fig. 15 we compare *CoCo* with *Co-Fi*, as we consider $R_c = R_s$ or $R_c < 2R_s$, which is not a case in *DCM*. As shown in Fig. 14 and Fig. 15, *CoCo* outperforms *DCM* and worse performs *Co-Fi* in terms of connectivity as number of partitions are increased with time in *CoCo*. This situation arises in CoCo, as the number of neighbors is decreased when range of $R_c$ is reduced and thereby increasing the probability of partitions. However, as discussed above, since we do not consider the redundant nodes to compensate the coverage or connectivity, we use the least mobility of nodes to maintain the network, thereby saving more energy of the nodes. Since, after certain time increase in number of partitions or coverage holes is a natural phenomenon in all of the algorithms, we propose our algorithms to solve these problems with least energy consumption and least average mobility of the nodes.

## V. Conclusion

In this paper, we design the low mobility sensor network models to maintain the connectivity and coverage problems, if nodes in the network are dead either in a predictable or unpredictable way. We design dynamic maintenance algorithms without disturbing the existing communication and coverage systems of the network taking $R_s \leq R_c < 2R_s$, and our low mobility of nodes can guarantee the network integrity. In our algorithms, only one-hop connecting neighbors of a dead node has to move, thereby minimizing the energy consumption due to mobility. From our performance analysis, it is clear that our algorithms outperforms in terms of energy consumption and average mobility distance as compared to similar algorithms along this direction.

## Acknowledgment

This work is supported by the National Science Council of Republic of China under grant NSC 95-2221-E-238-003.

## References

[1] S. Ganeriwal, A. Kansal, and M. B. Srivastava, "Self aware actuation for fault repair in sensor networks", in *Proc. IEEE International Conference on Robotics and Automation*, vol 5, pp. 5244-5249, Spain, May, 2004.

[2] A. Sekhar, B. S. Manoj, and C. S. R. Murthy, "Dynamic Coverage Maintenance Algorithms for Sensor Networks with Limited Mobility", in *Proc. IEEE International Conference on Pervasive Computing and Communications*, pp. 51-60, USA, Mar, 2005.

[3] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem", in *Proc. International Symposium on Distributed Autonomous Robotic Systems*, pp. 299-308, Japan, Jun., 2002.

[4] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment", *IEEE Transactions on Mobile Computing*, Volume 5, Issue 6, pp. 640-652, Jun., 2006.

[5] R. Mi and A. Chandrakasan, "Energy-Efficient Communication for Ad-Hoc Wireless Sensor Networks", in *Proc. Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers*; 1, pp 139-143, Pacific Grove, CA, Nov, 2001.

[6] B. Karp, and H.T. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks", in *Proc. ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 243-254, USA, Aug., 2000.

[7] M. B. McMickell, B. Goodwine, and L. A. Montestruque, "MICAbot: A Robotic Platform for Large-Scale Distributed Robotics", in *Proc. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1600-1605, Taipei, Taiwan, Sep. 2003.

[8] J. P. Sheu, P. W. Cheng, and K. Y. Hsieh, "Design and Implementation of a Smart Mobile Robot", in *Proc. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, vol. 3, pp. 422-429, Canada, Aug., 2005.

[9] G. T. Sibley, M. H. Rahimi, G. S. Sukhatme, "Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks", in *Proc. IEEE International Conference on Robotics and Automation*, pp. 1143-1148, USA, Sep., 2002.

[10] http://www.kvh.com/DigiComp/