# Distributed coordinate-free algorithm for full sensing coverage

Xiaoyun Li

Complex and Adaptive Systems Laboratory,
University College Dublin,
Dublin 4, Ireland
Email: xiaoyun.li@ucd.ie

David K. Hunter*

School of Computer Science and Electronic Engineering,
University of Essex,
Colchester, CO4 3SQ, UK
Email: dkhunter@essex.ac.uk
*Corresponding author

**Abstract:** This paper shows how a distributed algorithm, derived and justified through computational geometry, can detect and recover holes in the coverage provided by wireless sensor networks. Computational geometry is used to define conditions for the existence of holes in sensor coverage and based on these conditions, it is used to generate new distributed algorithms to detect and recover coverage holes. The algorithm does not require coordinates or location information and requires only minimal connectivity information. Most holes can be detected with very low probability of error and simulation results suggest that redundant nodes are selected efficiently for activation when recovering the hole. Unlike existing proposals, the complexity of the algorithms described here does not depend on the overall size of the network and they do not require flooding, with its associated high level of traffic.

**Keywords:** wireless sensor networks; computational geometry; coordinate-free; hole detection; self-healing; network protocols; signalling; received signal strength indication; simulation.

**Reference** to this paper should be made as follows: Li, X. and Hunter, D.K. (2009) 'Distributed coordinate-free algorithm for full sensing coverage', *Int. J. Sensor Networks*, Vol. 5, No. 3, pp.153–163.

**Biographical notes:** Xiaoyun Li received a diploma in Computer Technology Applications from the Department of Electronic Engineering, Shenzhen University, PR China in 1993. He was awarded an MSc degree in Computer Information Networks from the Department of Electronic Systems Engineering at the University of Essex, UK in 2004, and graduated in 2008 with a PhD from the same department. His current research interests include wireless sensor networks, media access protocols and routing protocols.

David K. Hunter is a Reader in the Department of Electronic Systems Engineering at the University of Essex, UK. He was awarded a BEng with First Class Honours in Electronics and Microprocessor Engineering from the University of Strathclyde, UK in 1987 and a PhD in Electronics and Electrical Engineering from the same university in 1991. He has authored or co-authored over 125 conference and journal publications and was an Associate Editor of the *IEEE/OSA Journal of Lightwave Technology* from 2001 to 2006. His current research focuses on wireless sensor networks, networking protocols, optical networks and optical packet switching. He is a Senior Member of the IEEE, a Chartered Engineer, a Member of the IET and a Professional Member of the ACM.

## 1 Introduction

Wireless Sensor Networks (WSNs) monitor some specified physical quantity, collating and delivering the sensed data to at least one sink node, usually via multiple wireless hops. They are composed of very small, lightweight sensor nodes, each of which includes the sensor itself, a processor, a radio transceiver and a battery. Applications include environmental monitoring, conferencing, battlefield operations, disaster relief, rescue operations and police operations. They can be deployed in virtually any environment, even those that are inhospitable, or difficult for humans to reach.

Several types of hole can exist in such networks, each creating particular problems. Examples include coverage holes, routing holes, jamming holes, sink/black holes and worm holes (Ahmed et al., 2005). This paper focuses on the problem of coverage hole detection and recovery.

The sensor network must be able to sense the required physical quantity over the entire area to be monitored – this 'coverage issue' is fundamental to both power saving and data aggregation. As adjacent sensors may collect similar data, a subset of carefully selected sensors, covering the whole sensing area, can reduce data redundancy, preventing unnecessary use of battery power and hence prolonging the lifetime of the network itself.

It is important that no 'coverage holes' exist, these being parts of the area to be monitored which are not covered by any active nodes. Any such coverage holes that do exist must be 'recovered', where active nodes are elected within the hole in order to restore coverage. This paper focuses on coverage hole detection and recovery in wireless sensor networks, providing resilience of the wireless sensor network to failure scenarios such as nodes running out of battery power. This is especially important in hostile environments where node failure is more frequent than otherwise.

In this paper, arguments from another discipline, namely computational geometry, are employed to devise a distributed algorithm named 3MeSH (Triangular Mesh Self-organising Self-Healing protocol) which elects nodes for full coverage efficiently and detects holes introduced either by faults in these nodes, or by mobility of nodes. Computational geometry is used to define conditions for the existence of holes in sensor coverage, and based on these conditions, distributed algorithms are derived to detect and recover holes. Hole recovery is attempted by activating redundant nodes. Location awareness is not necessary, which is highly advantageous since micro-sensors generally cannot obtain this information. Through this computational geometry approach, a solution has been found to a challenging problem in wireless sensor networks, while overcoming the problems implied by previous solutions.

### *1.1 Related work*

There has been much related research on the coverage problem for both mobile and static sensor networks. Most such proposals use computational geometry with geometric tools such as Voronoi diagrams to detect holes. Solutions exist for single-level coverage, in static sensor networks (Meguerdichian et al., 2001; Tian et al., 2002; Huang et al., 2003; Zhang et al., 2003; Carbunar et al., 2004; Jiang et al., 2004; Wu et al., 2004), and in mobile sensor networks (Howard et al., 2002; Wang et al., 2004). Also, solutions exist for multiple coverage overlays in dense static sensor networks (Li et al., 2003; Wang et al., 2003; Yan et al., 2003). All these techniques require coordinate information about the sensor nodes in the target sensing area. As a node failure can disrupt full coverage, self-recovery must be implemented. *GS3* (Zhang et al., 2002) and *SoRCA* (Wang et al., 2005) employ self-healing algorithms to recover a hole with adjacent redundant nodes, but they require coordinate location information for all sensor nodes.

An algebraic topological method employing homology theory detects single overlay coverage holes without coordinates (Ghrist et al., 2005; de Silva et al., 2005). It employs a central control algorithm that requires connectivity information for all nodes in the sensing area. If there are $N$ nodes, the calculation time is $O(N^5)$. For 3MeSH, this is $O(HD^2)$, where $D$ is the maximum number of other active nodes which overlap a node's sensing area, and $H$ is the worst-case number of redundant nodes in a large hole, where $H \geq D$. In 3MeSH, the algorithmic complexity does not depend on the overall size of the network, whereas the homology algorithm encounters severe difficulties with networks of over say 1000 nodes, even with a powerful central computer and the message forwarding overhead can be impractically large, since the algorithm is centralised.

A boundary detection algorithm using a communication graph has also been proposed (Funke and Klein, 2006; Wang et al., 2006). It selects one or more nodes as seeds, to build a shortest path tree by flooding. With no coverage holes in a continuous target area, nodes which lie the same number of hops away from the seed node should form an unbroken circle. Therefore, each node can examine those neighbours which are an equal number of hops away from it, to determine whether or not such a circle can be formed. In this way, the boundary nodes of a coverage hole can be detected. This algorithm is suitable for a coordinate-free environment, but it requires the node density to be high enough to guarantee sufficient accuracy and it cannot detect multiple adjacent holes. Another problem is the high communication overhead of frequent flooding when the network topology changes due to node failure or mobility.

### *1.2 The 3MeSH algorithm*

The 3MeSH (Triangle Mesh Self-Healing) algorithm has the advantage over previous proposals of being distributed and requiring only local connectivity information. It has the following features:

1 It does not use or require co-ordinates and each node requires only local connectivity information. Each node requires no orientation or distance information about other nodes.

2 Node election, hole discovery and hole recovery are all accomplished by distributed algorithms. Each active node independently determines whether it is a boundary node, after receiving connectivity information from nearby nodes.

3 The algorithm is scalable to a large number of nodes, because the calculation time is not dependent on the overall size of the network.

4 Simulation results suggest that if a coverage hole exists resulting from node failure or node mobility, it elects nodes efficiently to recover the hole, without excessive redundancy.

5 The radio transmission range of each node must be at least twice the sensing range. (If the transmission range were less, then each node would not necessarily be able to transmit to its neighbours).

6 The communications overhead for active node election and hole detection is low, with flooding not being required. Only active nodes one or two hops apart need exchange adjacency messages for boundary node detection, while only nodes adjacent to a detected hole need communicate to recover it.

### *1.3 Assumptions*

Throughout most of the discussion in this paper, it is assumed that all sensor nodes have a circular sensing area of radius $R$, and that each node determines adjacency by measuring the distance to each nearby node, specifically finding whether it lies within distance $R$ units, or distance $2R$ units. Other ways of determining adjacency are discussed in Section 4.6.

If two nodes are less than $R$ units apart, they are *covered* by each other, and should not both be active, since only a subset of nodes need be active to obtain full sensing coverage. Two active nodes are called *neighbours* if they are between $R$ and $2R$ apart, where the implied connection between them is called a *link*. A circular sensing area, as discussed above, is assumed initially, although Section 4.6 considers irregular areas.

In its simplest form, 3MeSH can accurately detect large holes with up to ten edges, although larger holes can be detected by increasing the range over which each node gathers connectivity information.

## 2 Coverage holes

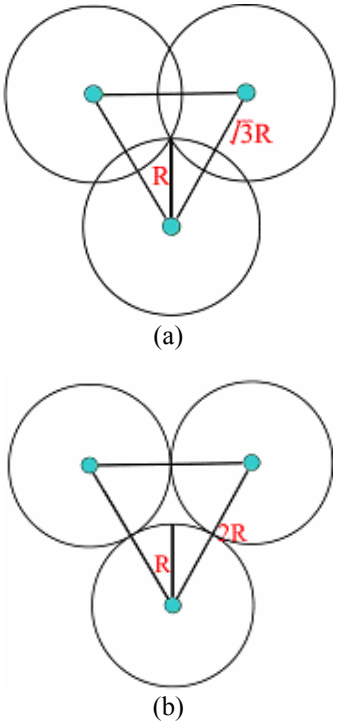
If the target area can be partitioned into triangles formed by links, and every part of each triangle is covered by one or more nodes, then no *coverage hole* (or simply *hole*) exists. Since the entire target area (not merely an individual node's sensing area) is partitioned into triangles, the procedure is not influenced by the shape of each node's sensing area.

There are two types of hole:

1 *Large hole*: This is found by checking whether the target area can be partitioned into triangles by links. If this is not possible, a large hole exists, having by definition at least four edges. The discussion in this paper concentrates on the detection and recovery of such large holes.

2 *Trivial hole*: If any uncovered position exists inside a triangle formed by links, a *trivial hole* exists [Figure 1(b)]. Trivial hole detection is straightforward if accurate distance information between neighbours is available.

**Figure 1** (a) Three nodes without a trivial hole and (b) trivial hole (see online version for colours)




## 3 The algorithm

Given a set of sensor nodes $V$ in a coordinate-free environment, a graph $G(V, E)$ may be defined, where the set $E$ contains all links between each node in the set $V$ and its neighbours. The graph hence indicates connectivity between neighbours by means of edges, but need not directly relate to the geographical or spatial organisation of the nodes.

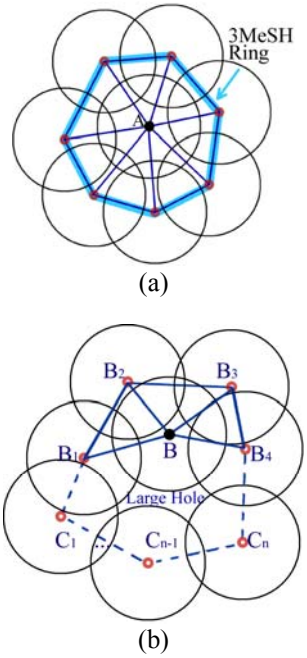
Nodes must not be active unnecessarily, because this wastes battery power, and hence shortens the life of the network. To prevent this, a subset of the nodes are elected and designated as active nodes, which reduces the size of the graph. Each node can potentially elect itself as an active node, where any node covered by it must become a redundant node. After all nodes have been designated as either active or redundant in this way, active node election is complete.

Figure 2 shows how a sub-graph of $G(V, E)$ is defined by a node and its neighbours, in order to facilitate discussion of the algorithm. In Figure 2(a), node $A$ is a *non-boundary node* (since it is not on the boundary of a large hole), but in Figure 2(b), a large hole exists, and node $B$ is a *boundary node*. Each active node $N$ defines a graph $G(V_N, E)$, where $V_N$ is the set of active nodes neighbouring $N$ and $E$ is the set of links between neighbouring pairs of nodes in $V_N$ and $N$ itself is not included in $V_N$.

In Figure 2(a), all the nodes in $V_A$ define a closed polygon, known as a *ring*. If the nodes in set $V_A$ cannot by themselves define links which partition the area within the ring into triangles, it is known as a *3MeSH ring*. By definition, all the links terminating on node $A$, in addition to the links in set $E$, partition the area within the 3MeSH ring into triangles, therefore node $A$ is not adjacent to a large hole. However,

in Figure 2(b), all the neighbours of node $B$, namely nodes $B_1$, $B_2$, $B_3$ and $B_4$, do not form a ring encircling $B$, and nodes $C_1$ through to $C_n$ must be added to do this. Therefore a large hole exists, enclosed by a ring with at least four edges, which cannot be partitioned into triangles by links.

**Figure 2** (a) 3MeSH ring and (b) a large hole (see online version for colours)



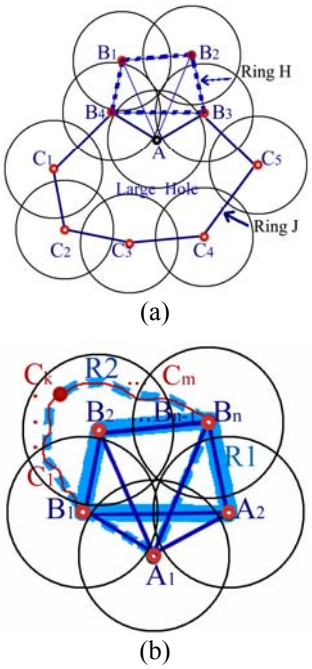

If all members of the set $V_A$ can form a 3MeSH ring, then node $A$ is a non-boundary node, but this is apparently inconsistent with Figure 3(a), where $A$ lies outside its 3MeSH ring. Lemma A states that in this configuration, where $A$ has one or more neighbours which are boundary nodes, it may be adjacent to a large hole, and if so, the hole can be found without $A$ being a boundary node. It is therefore reasonable to regard $A$ as a non-boundary node in such cases.

Lemma A: *If there is a large hole adjacent to node A, and all its neighbours form a 3MeSH ring, the large hole can always be detected without considering node A as a boundary node.*

*Proof*: In Figure 3(a), node $A$ has a 3MeSH ring $H$, which has four nodes in this example, namely $B_1$, $B_2$, $B_3$ and $B_4$. Another ring $J$ encloses the large hole, although it has nodes $B_3$ and $B_4$ in common with ring $H$. Initially, assume that ring $J$ includes node $A$, which must be a neighbour of at least two nodes in $J$, $B_3$ and $B_4$ in this case. This is because, since node $A$ is on ring $J$, it must have two neighbours on either side along the path around $J$, which must also be on ring $H$, because all neighbours of node $A$ form a 3MeSH ring $H$. Then a new ring $J$ may be defined, which also encloses the large hole, and which includes all the nodes in ring $J$ except node $A$. Depending on network configuration, other nodes from ring $H$ may exist on ring $J$ between nodes $B_3$ and $B_4$. Since node $A$ is no longer on ring $J$, it is not now considered to be a boundary node. The proof holds in either possible case, namely where node $A$ lies inside ring $H$, or when it lies outside, as already discussed and as shown in Figure 3(a). Therefore the large hole can be detected without considering node $A$, and Lemma A is proved.

Lemma B states that if two nodes are each located outside their respective 3MeSH rings, they cannot be neighbours. In particular, assume that they are both non-boundary nodes such as $A$ in Figure 3(a). If they were neighbours, the hole would not have been detected because they were both identified as non-boundary nodes.

**Figure 3** (a) Undetected boundary node and (b) adjacent non-boundary nodes (see online version for colours)




Lemma B: *If $A_1$ ($A_2$) has a 3MeSH ring $R_1$ ($R_2$), and node $A_1$ ($A_2$) lies outside ring $R_1$ ($R_2$), then nodes $A_1$ and $A_2$ cannot be neighbours.*

*Proof*: We proceed via proof by contradiction. Assume nodes $A_1$ and $A_2$ are in fact neighbours. Figure 3(b) depicts this, without loss of generality. Node $A_1$ has a 3MeSH ring $R_1 = \{B_1,\ldots, B_n, A_2\}$. A subset of neighbours of node $A_2$, namely $\{A_1, B_1, C_1,..., C_k,..., C_m, B_n\}$, form another ring $R_2$. Node $C_k$ cannot be inside ring $R_1$ because if it were, it would be a neighbour of node $A_1$ and $A_1$ could then partition ring $R_2$ into triangles, meaning that it cannot be a 3MeSH ring defined by node $A_2$. Hence node $C_k$ must lie outside ring $R_1$, with $\{B_2,\ldots,B_{n-1}\}$ being inside ring $R_2$. Because node $A_2$ is adjacent to all nodes on ring $R_2$, which has nodes $\{B_2,\ldots,B_{n-1}\}$ inside it, node $A_2$ is also adjacent to them. Therefore links between node $A_2$ and nodes $\{B_2,\ldots,B_{n-1}\}$ partition ring $R_1$ into triangles, meaning that it cannot be a 3MeSH ring defined by node $A_1$. Hence if nodes $A_1$ and $A_2$ are neighbours, both cannot lie outside their own 3MeSH ring, whether node $C_k$ is inside or outside ring $R_1$, so Lemma B is proved by contradiction.

Therefore a node can either have neighbours which form a 3MeSH ring, or it cannot. The former case is further subdivided into the case where all neighbours form a 3MeSH ring, or only a proper subset can do so. Clearly, this subdivision covers all cases. In our algorithm, a node $A$ is always regarded as a boundary node if a proper subset of set $V_A$ can form a 3MeSH ring, but all nodes in $V_A$ cannot.

The theoretical justification is omitted for brevity. While this assumption is not always valid, it simplifies implementation, without compromising the effectiveness of the shortcut detection algorithm described below.

Hence, assisted by suitable signalling protocols, a node can determine whether all its neighbours define a 3MeSH ring and hence whether it is a boundary node of a large hole. With this information, large hole detection is possible. If at least four boundary nodes form a ring, with part of the ring always forming the unique shortest path between any two boundary nodes, then they define a large hole. Again, a protocol must exist to distribute the local information necessary to implement this. Once a large hole is detected, the hole recovery algorithm tries to cover it using redundant nodes within the ring that defines it, although some holes are unrecoverable if sufficient redundant nodes do not exist.

Hence the entire process may be summarised as follows:

1 Elect a set $V$ of active nodes.

2 Each active node collects connectivity information from its neighbours and their neighbours, and decides whether it is a boundary node by detecting the existence of a 3MeSH ring defined by all its neighbours. If it is assumed that each node $A$ has $D$ neighbours forming the ring, and each of these has a further $D$ neighbours, the time complexity is $O(D^2)$. When detecting whether the ring can be partitioned into triangles without $A$, the complexity becomes $O(D^3)$.

3 Detect large holes defined by the boundary nodes, with time complexity $O(N^2D^N)$, that is $O(D^2)$ or $O(D^3)$ for $N = 2$ and $N = 3$ respectively, where $N$ is the maximum number of hops over which a boundary node gathers connectivity information.

4 If a large hole is detected, attempt hole recovery, trying to minimise the number of redundant nodes used. This has algorithmic complexity $O(HD^2)$, where $H$ is the worst-case number of redundant nodes in a large hole.

Section 4 describes these steps in more detail.

## 4 Large hole detection

### *4.1 Active node election*

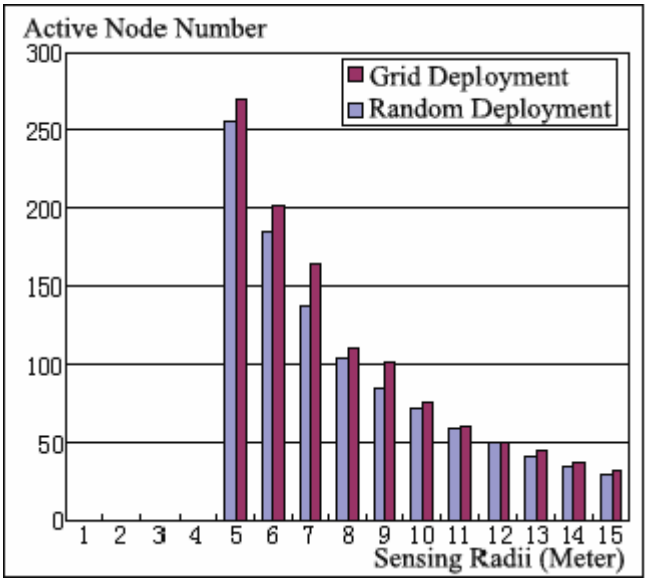
In high-density sensor networks, the mean number of nodes covered by each sensor node's sensing range is relatively high. A small subset of the nodes are elected as active nodes to achieve full coverage, while the remaining nodes are called *redundant nodes*, and are placed into standby mode in order to save energy.

Firstly the sink elects itself as an active node, and broadcasts an active node message to trigger active node election. Any node within a distance of between $R$ and $2R$ units of it may be elected as an active node, hence a node falling within its sensing area has a distance less than $R$ and is regarded as being redundant. Once a new active node is elected, it in turn broadcasts an active node message. The non-redundant nodes repeat this process until all the nodes become designated as either active or redundant.

Each active node message may be received by several candidate nodes. To prevent several nodes being elected simultaneously, the first candidate node to receive such a message registers with the existing active node by sending a registration message. The candidate node is declared as active only after receiving a confirmation message containing its node ID from the existing active node. If more than one candidate node sends a registration message to an active node simultaneously, these messages collide and cannot be received, so the active node does not send a confirmation message to the candidate nodes. After its timer expires, the candidate node re-transmits the registration message. Alternatively, it may become redundant if another node within distance $R$ units has since been declared as an active node.

In the optimum case, all adjacent active nodes form a triangular grid, where the distance between two adjacent nodes is $\sqrt{3}R$ [Figure 1(a)]. This achieves full coverage while minimising overlap between node coverage areas. Election of candidate nodes with distance close to $\sqrt{3}R$ from an existing active node is not in fact a strict condition for correct operation of the algorithm. It only requires the distance between any two active nodes to be no less than $R$, although this could result in the election of more active nodes than is strictly required.

**Figure 4** Mean number of active nodes for full coverage (see online version for colours)




Assuming that each node's sensing range is a circle of radius $R$, the nodes (deployed in an equilateral triangular grid with distance √3R between adjacent nodes) can achieve full coverage in the entire deployment area. The unique coverage area for each active node is a hexagonal cell of

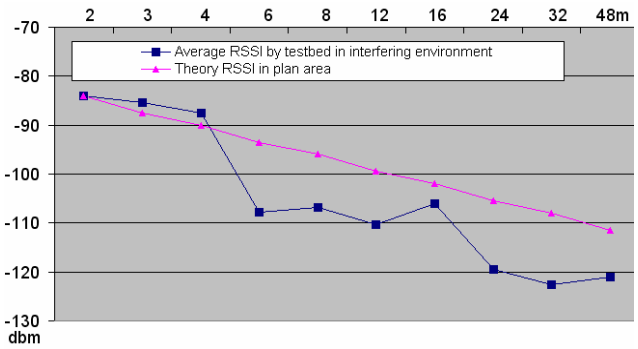
area $3\sqrt{3}R^2/2 \approx 2.6R^2$. A polygon with more sides (of equal length) has a greater coverage area, tending to $3.14R^2$, but the triangular grid is a simple topology, which partitions the area into hexagonal cells. The distance from one node to each of its neighbours is the same, and the minimum number of active nodes is: Total Sensing Area/$2.6R^2$.

When simulating a high density sensor network, 1024 nodes were deployed within an area of 100 m × 100 m in either a random or a grid configuration. Figure 4 shows the average number of active nodes for full coverage as the sensing radius varies from 5 metres to 15 metres. Both grid deployment and random deployment are considered, with the active node election algorithm. Both configurations involve the deployment of similar numbers of active nodes.

### 4.2 Received Signal Strength Indication (RSSI) distance estimation error

Received Signal Strength Indication (RSSI) can be used in an attempt to estimate the distance between a radio transmitter and a receiver. Within an experimental micro-sensor testbed, we found that the accuracy when estimating the distance using RSSI is poor – Figure 5 shows a typical example of the results that we obtained. In a plane area without obstacles, the RSSI attenuation is proportional to square of distance, but this may increase to powers of 3 or 4 for indoor radio transmission, or if there are a variety of obstacles. The RSSI value can change with time, due to variable background noise and other issues concerned with propagation. Therefore it is reasonable to ask whether it is feasible to estimate distance using RSSI.

**Figure 5** An example of the variation of RSSI with distance (see online version for colours)




As the 3MeSH algorithm is distributed, and does not involve adding up many smaller distances to make a larger distance, the estimation error and topology distortion do not accumulate. Also it requires only a very coarse measurement of distance, namely whether two nodes are within either the sensing radius or twice the sensing radius. Therefore the RSSI value can in fact be used with the 3MeSH algorithm, despite the shortcomings discussed above. Indeed, for example, one simulation demonstrated that the number of active nodes for full coverage increases from 30% of the total where there is no error in distance estimation, to 60% when the RSSI distance estimation error is 20%.

### 4.3 Assumptions

To detect a large hole, a set of at least four boundary nodes must be found which form a ring. The number of nodes in this set is denoted by $k$, where $k > 3$. Consider every pair of nodes on the ring, and assume in each case that $S$ is the number of hops separating them, with $2 \le S \le \lfloor k/2 \rfloor$, where $\lfloor x \rfloor$ is the largest integer which is not greater than $x$. If for each pair of boundary nodes, the shortest path between them is also $S$ hops, then a hole bounded by the ring must exist. If each node collects connectivity information from its neighbours up to $N$ hops away, shorter routes (known as *shortcuts*) with up to $S = 2N$ hops can be detected, so large holes with up to $4N + 2$ edges may be found. While the sensing radius is $R$, the radio transmission distance is assumed to be $4R$ or, optionally, $6R$, as discussed below. Of course, longer transmission distances could in principle be used in order to collect more connectivity information for each node, but they are not considered in this paper.

When detecting boundary nodes, it is generally assumed that each node only collects connectivity information from its neighbours and its neighbours' neighbours, hence $N = 2$, so rings with up to 10 edges may be detected. However, larger holes can be detected by permitting nodes to collect connectivity information from nodes further away, at the expense of increasing protocol complexity, computational complexity and network signalling traffic.

### 4.4 Signalling protocol

The procedure for detecting large holes may be summarised as follows:

1 When a node is elected as an active node, it broadcasts an active node declaration message.

2 After election, each active node broadcasts a message containing its own ID, indicating its unique address or serial number and the IDs of all its neighbours. This implies $N = 2$, as discussed above, and the message complexity is hence $O(D^2)$.

3 Each active node collects all these messages, and by attempting to detect a 3MeSH ring defined by all its neighbours, uses this information to deduce whether it is a boundary node. Then it broadcasts a boundary node detection message, declaring whether or not it is a boundary node, and containing the IDs of all its neighbours, and all their neighbours, implying in this case that $N = 2$.

4 Optionally, to detect larger holes, each boundary node collects boundary node detection messages from its neighbours, and broadcasts these to all active nodes up to $N = 3$ hops away. Since connectivity information is collected over three hops instead of two, shortcut paths of up to $S = 3 \times 2 = 6$ hops, instead of $2 \times 2 = 4$ hops may be detected. Furthermore, the maximum size of ring enclosing a detectable hole increases from $2 \times 4 + 2 = 10$ edges to $3 \times 4 + 2 = 14$ edges. However, the volume of data traffic increases, as does the amount of computation required, so this feature is optional. The message complexity is hence $O(D^N)$.

5 Such large holes can be detected, because each boundary node is aware of all node adjacencies within its range and it can hence construct a graph, generated from boundary node detection messages and with nodes as vertices, for large hole identification. If the shortcut filter algorithm running on every boundary node finds no shortcuts between each pair of nodes on the ring 2 to 5 (or 7) hops away, a boundary node broadcasts a *hole message* containing the IDs of all boundary nodes on the ring enclosing the hole, providing another node has not already done so. Shortcut filtering is also accomplished with the graph, with time complexity is $O(N^2D^N)$, as there are $O(N^2)$ pairs of boundary nodes on the ring, each requiring all paths over the ring between them of mean length $O(N)$ to be tested.

### 4.5 Shortcut filter algorithm

When the shortcut filter algorithm detects a large hole among all boundary nodes, a subset of boundary nodes forming a ring is identified, with no shortcut path between any pair of nodes on the ring.

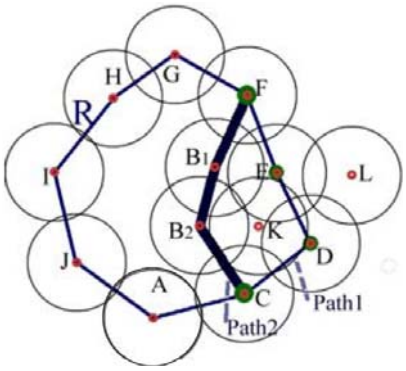
If two paths of the same length are detected between two boundary nodes, all neighbours of the nodes on each path are found. If the neighbours of path 2 are a proper subset of the neighbours of path 1, then path 2 is closer to the hole. Hence path 1 is discarded [Figure 6(a)].

The shortcut filter algorithm cannot select the closest path to the hole in the following cases:

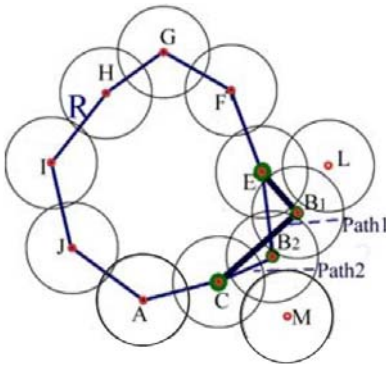
1 Assume two paths between a pair of boundary nodes have the same number of hops, and the neighbours to nodes on both paths are as shown in Figure 6(a), when disregarding node $L$. In this case, the algorithm would discard one path randomly and therefore cannot guarantee to select the closet path to the hole.

2 Assume that two paths between two boundary nodes have the same number of hops, the set of neighbours to nodes on each path is different, and one set is not a proper subset of the other [Figure 6(b)]. In this case, the algorithm would not discard either path, therefore the same hole would be detected twice with different subsets of boundary nodes.

**Figure 6** (a) Hole detection with two shortest paths – case 1 and (b) hole detection with two shortest paths–case 2 (see online version for colours)




(a)



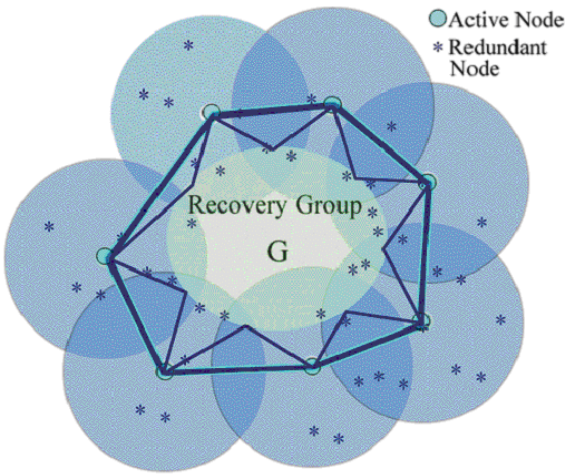

(b)

Once a large hole has been found, it may often be recovered by activating redundant nodes inside the ring.

### 4.6 Hole recovery

After a boundary node $K$ broadcasts a hole message, the hole recovery algorithm selects as few redundant nodes as are necessary in order to recover the hole (Figure 7).

**Figure 7** Hole recovery by a set G of redundant nodes (see online version for colours)



1 If a redundant node is adjacent to all the boundary nodes, it elects itself as an active node in order to recover the hole, and the algorithm therefore terminates.

2 Otherwise, the set $G$ initially contains all redundant nodes which are covered by the hole's boundary nodes. Node $K$ attempts to optimise $G$, the set of redundant nodes to be recovered, by removing redundant nodes. Each member $B$ of set $G$ is considered in turn. For each node $B$, all its neighbour's neighbours are examined, and a record is kept of how many ways each such node $A$ can be reached in this way from $B$. If this is equal to the number of neighbours' node $B$ has, node $A$ must have at least all the same neighbours as $B$. Hence node $B$ is redundant and can be removed from set $G$. Indeed, node $A$ may have neighbours and therefore links which node $B$ does not have, which can partition the hole into triangles. Several iterations of the above procedure are necessary because during each one, nodes are removed from set $G$, and their links are also discarded, hence the remaining nodes must re-calculate their adjacencies. The algorithm exits early if an iteration does not remove any redundant nodes from set $G$. If there are $H$ nodes in set $G$ initially, and since each node $B$ has $O(D^2)$ two-hop neighbours, the overall time complexity is $O(HD^2)$.

3 The nodes in set $G$ are all elected, making them active, and the algorithm terminates. If a large hole is then detected in the same place, it is unrecoverable. Otherwise, the hole has been recovered successfully.

In step 2 above, boundary node $K$ selects redundant nodes efficiently for hole-recovery set $G$. Simulations on over 500 random topologies with 1284 hole detections show that in 1280 cases (99.7%), the algorithm converges after the third iteration, and that five iterations are sufficient for all other cases. Inspection of the simulation results suggests that few redundant nodes are activated unnecessarily.

This process does not necessarily minimise the number of recovery nodes, because if the set of neighbours of redundant nodes $A_1, A_2,..., A_n$ is a subset of all neighbouring nodes of nodes $B_1, B_2,..., B_m$, then $A_1, A_2,..., A_n$ should be discarded from set $G$. This case is not considered by the present algorithm, since it is more complex to implement.

### *4.6 Detection of adjacency*

Boundary node detection relies on determining adjacency of nodes' sensing areas. Even if there is no information available about the distance between nodes, or the shape of their sensing areas, adjacency can still be determined, and boundary nodes can still be detected. It is merely necessary to determine without a third party whether a pair of nodes have overlapping sensing areas, regardless of the shape of them.

As an alternative to two nodes measuring the distance between them, they can detect one other through some hardware-based metric, depending upon application. If two nodes can detect each other with a certain radio signal strength, they could be considered as being adjacent. If the signal strength is above a threshold, they are considered to be covered by each other, and one of them can become redundant. If the sensing area is approximately circular, then when data and signalling information is transmitted by nodes in proximity, the RSSI can be used to ascertain adjacency.

Alternatively, in some applications, nodes could compare sensed information to determine whether their sensing ranges overlap, since with some applications, this must take place if they possess information in common. Nodes could also generate signals to be received by other nodes nearby, and nearby nodes could detect visual or other electromagnetic signals from one another. This is an important issue when implementing the algorithms described in this paper, and it merits further study. This all applies regardless of whether the signal in question permits estimation of the sensing range, with the important issue being whether it permits any two nodes to detect one other's presence.

Hence signal strength is only one practical definition of adjacency, and in fact, neither the distance between nodes nor the shape of the irregular sensing area need be explicitly considered.

## 5 Simulation and examples

Simulation of the boundary node detection and large hole identification algorithms were performed using script code with MATLAB version 7.0, in Windows XP. Over 99.8% of recoverable large holes having up to ten edges were detected and recovered. Each node collects connectivity information from its neighbours up to two hops away, implying that $N = 2$ (see Section 4.3).

Although the MATLAB simulations do not consider those channel impairments that can lead to packet loss, the protocols are in any case designed to be resilient to these through re-transmission of crucial information. The simulations hence do not need to model the physical layer and MAC layer, since this would have negligible impact on the results. However, packet loss may occur in the simulations due to collision, if two redundant nodes adjacent to a large hole send data during the same slot – this is taken into account in the simulation model.

For boundary node detection, each active node need only collect connectivity information from and through its immediate neighbours, therefore a communication range of one hop (twice the sensing range) is sufficient. However, since it is assumed a hole with up to ten edges can be detected, the communication range for hole detection must be at least five hops, if messages are to be passed in a single hop. (If forwarding through neighbours is possible, a smaller communication range could in principle be sufficient).

Mobility is not considered in this section, since it is assumed that the nodes are all stationary. The algorithms described here could nevertheless be employed in a mobile environment. Nodes can go offline in the sense that they can fail or run out of battery power, potentially creating a coverage hole. However, nodes cannot go offline to carry out offline calculations – this is to be avoided in a time-critical scenario such as this.

500 simulations were carried out, involving over 4000 recoverable or unrecoverable large holes being detected with only five holes recovery errors. Although all holes formed by active nodes were correctly identified, 0.125% of these apparently could not be filled by redundant nodes. This is either because of the 'two shortest paths' problem of Figure 6, or because the hole has in fact been recovered, but this is not detected by the algorithm. Furthermore, the results show that redundant nodes are activated efficiently, with few nodes being activated unnecessarily when recovering each large hole. The active node density in a recovered hole is therefore similar to that found in a failure-free environment.

### *5.1 Single overlay large hole detection*

256 nodes were simulated inside a 100 m × 100 m area, with nodal sensing radii of 10 m. With a grid topology, 64 nodes were elected as active nodes, with no hole detected in the target area (Figure 8). For random deployment, a mean of 63.4 active nodes were elected in over 100 random deployments. Figure 9 shows hole detection and recovery with random deployment. One unrecoverable hole and six recoverable holes are detected.

Simulations of 100 random deployments show that the mean number of active nodes elected is similar to that with grid deployment. There were a total of 256 active and redundant nodes. In 100 different random deployments, recoverable holes are always detected, but in 15 random cases, unrecoverable holes are also detected.

The simulation results demonstrate that the hole detection and recovery algorithm can elect a number of active nodes which is close to the minimum necessary in a high-density randomly-deployed network while maintaining full coverage. Large holes are always detected, but may not be recovered.

**Figure 8** 256 nodes deployed in a grid configuration (see online version for colours)

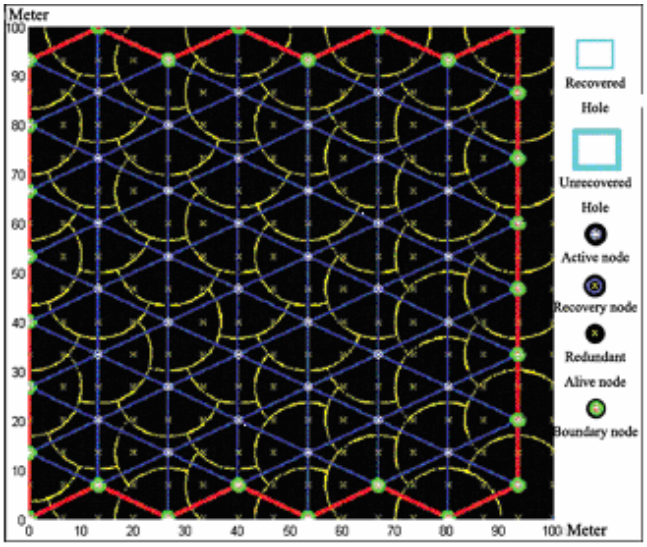
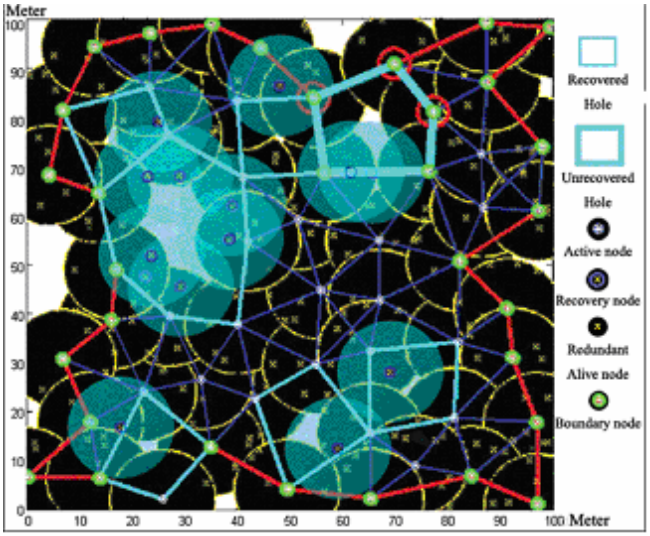



**Figure 9** 256 nodes deployed in a random configuration (see online version for colours)




### *5.2 Coverage hole detection and recovery due to accidental node failure*

This section considers a specific example, in order to illustrate operation of the algorithm. 1024 nodes with sensing radii of 8 metres were randomly deployed in area of 100 m × 100 m. In order to demonstrate operation of the algorithm, it was assumed that each elected active node had a failure rate of 2% per minute. Hole detection is performed every 10 minutes, so that after each 10 minute interval, approximately 20% of active nodes have failed accidentally. This results in one or more uncovered areas, each of which defines a large hole. Then the redundant nodes try to cover each area until the first unrecoverable hole is detected.

Figure 11 shows the sensing area after 300 minutes. In the final 10 minutes, 22 active nodes failed, creating 11 large holes which were recovered by the hole recovery algorithm. During the entire 300 minutes, there were two unrecoverable large holes, and the simulation terminated with 102 active nodes elected and a total of 476 nodes still alive.

Figure 10 shows that every 10 minutes, more active nodes are elected to retain full coverage by recovering, on average, approximately ten holes. The mean number of elected active nodes remains at approximately 100 throughout for full coverage, and is hence reasonable to assume that every 10 minutes, 20% of the target area is recovered since 20% of the active nodes have failed.

If an overlay is defined as the amount of sensor coverage required to cover the entire sensing area once, the total number of overlays which are recovered by new elected active nodes during the 300 minute simulation is 29 × 20% = 5.8. This is because in 300 minutes, recovery takes place 29 times.

Assume the ideal case where nodes can only fail through running out of battery power, and that failure for other reasons is not possible. Assume also that each active node has a uniform and constant battery lifetime of

*t* minutes. After *t* minutes, all the active nodes run out of battery power simultaneously, and using the 3MeSH active node election algorithm, new active nodes are elected which cover the entire target area. Simulations were carried out for this scenario, using the same number and positioning of nodes as before. The results show that, when repeatedly using the 3MeSH algorithm for active node election, only five mutually exclusive subsets can be formed one after another out of all available nodes, where each subset consists of elected active nodes providing full coverage. Some nodes are not members of any subset, and are always redundant because it is impossible to elect them. The result of 5.8 calculated above indicates that the hole recovery algorithm compares favourably with the hypothetical failure-free case.

**Figure 10** Hole recovery, showing the number of active nodes, failed nodes, and holes (see online version for colours)

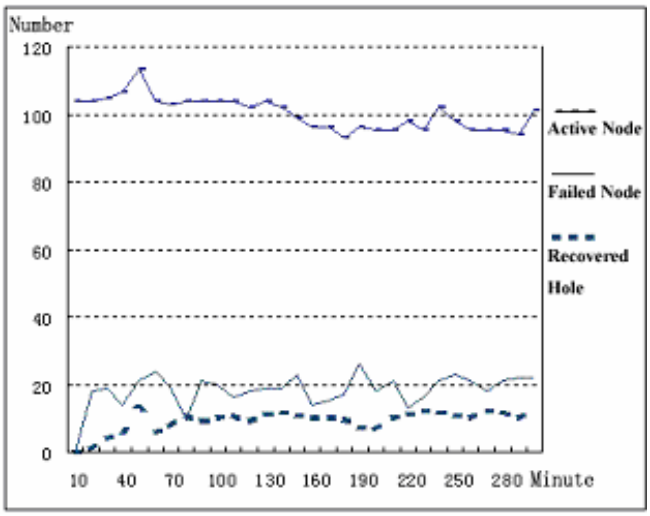
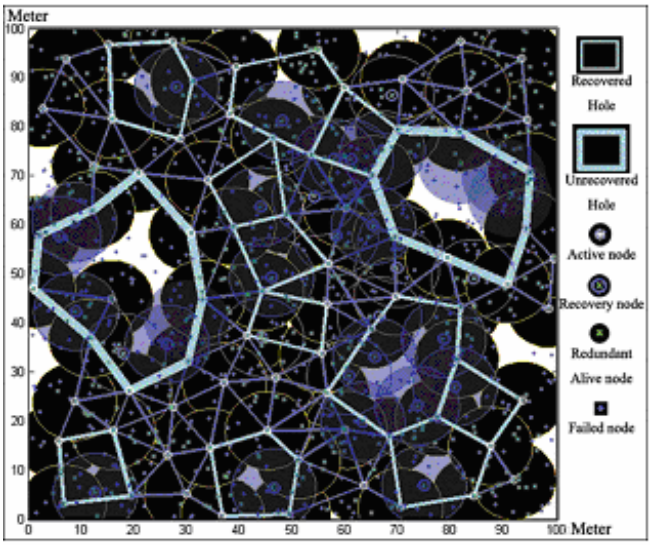



**Figure 11** Two unrecoverable holes (see online version for colours)




## 6 Conclusions

A simple distributed algorithm for wireless sensor networks, which implements coverage hole detection and recovery without requiring coordinates or location information, has been introduced. This provides resilience of the sensor network to the failure of sensor nodes, which is a particularly important issue in hostile environments where node failures occur more frequently than otherwise.

The algorithm utilises a simple connectivity model named the *3MeSH ring*, employing computational geometry and graph theory to define both appropriate conditions for the existence of holes and also algorithms to detect and recover them.

With minimal connectivity information (for example, information about whether any two nodes are between *R* and 2*R* metres apart, where *R* is the sensing range), large holes with between four and ten edges can be detected and recovered locally, with communication only being required between nodes bordering the hole. Larger holes may be detected at the expense of greater complexity and a higher level of signalling traffic.

The algorithm can recover large holes produced by accidental node failure, or topology changes in mobile ad-hoc networks. Recovery of trivial holes, having three edges, is also possible, but requires accurate information about the distances between nodes.

Because the algorithm is distributed, no central control is required for hole detection and recovery, which promises faster response than a centralised algorithm. Furthermore, communication is limited to those nodes neighbouring the hole boundary and flooding, with its associated high level of traffic, is not required. Unlike existing proposals, the algorithmic complexity does not depend on the overall size of the network, meaning that this concept can scale to arbitrarily large networks.

## References

Ahmed, N., Kanhere, S.S. and Jha, S. (2005, April) 'The holes problem in wireless sensor networks: a survey', *ACM SIGMOBILE Review*, Vol. 9, No. 2.

Carbunar, B., Grama, A., Vitek, J. and Carbunar, O. (2004, October) 'coverage preserving redundancy elimination in sensor networks', *IEEE SECON*.

de Silva, V., Ghrist, R. and Muhammad, A. (2005) 'Blind swarms for coverage in 2-D', *Proceedings of Robotics: Systems and Science*.

Funke, S. and Klein, C. (2006) 'Hole detection or: how much geometry hides in connectivity?', *Proceedings of the 22nd annual symposium on Computational geometry*, ACM Press, New York, NY, USA, pp.377–385.

Ghrist, R. and Muhammad, A. (2005, April 15) 'Coverage and hole-detection in sensor networks via homology information processing in sensor networks', *IPSN*, pp.254-260.

Howard, A., Mataric, M.J. and Sukhatme, G.S (2002, June) 'Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem', *6th International Symposium on DARS02*.

Huang, C-F. and Tseng, Y-C. (2003) 'The coverage problem in a wireless sensor network', *Proceeding of the 2nd ACM WSNA'03*.

Jiang, J. and Dou, W. (2004, July) 'A coverage preserving density control algorithm for wireless sensor networks', *ADHOC-NOW'04*, Springer-Verlag, pp.42–55.

Li, X-Y., Wan, P-J. and Frieder, O. (2003) 'Coverage in wireless ad-hoc sensor networks', *IEEE Transactions on Computers*, Vol. 52, No. 6, pp.753–763.

Meguerdichian, S., Koushanfar, F., Potkonjak, M. and Srivastava, M. (2001) 'Coverage problems in wireless ad-hoc sensor network', *IEEE INFOCOM*, pp.1380–1387.

Tian, D., Georganas, N.D. (2002) 'A coverage-preserving node scheduling scheme for large wireless sensor networks', *Proceedings of the 1st ACM International Workshop on WSN*, Georgia, USA.

Wang, G., Cao, G. and La Porta, T. (2004) 'Movement-assisted sensor deployment', *IEEE INFOCOM*.

Wang, X. and Berger, T. (2005) 'Self-organizing redundancy cellular architecture for wireless sensor networks', *WCNC*.

Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R. and Gill, C. (2003, November)'Integrated coverage and connectivity configuration in wireless sensor networks', *Proceedings of the ACM*, SenSys '03, pp.28–39.

Wang, Y., Gao, J. and Mitchell, J.S.B. (2006) *Boundary Recognition in Sensor Networks by Topological Methods*, MobiCom '06, ACM Press, New York, NY, USA.

Wu, J. and Yang, S. (2004) 'Coverage issue in sensor networks with adjustable ranges', *ICPP Workshops*, pp.61–68.

Yan, T., He, T. and Stankovic, J. (2003) 'Differentiated surveillance for sensor networks', *Proceedings of the ACM*, SenSys '03.

Zhang, H. and Hou, J.C. (2003) 'Maintaining sensing coverage and connectivity in large sensor networks', *UIUC*.

Zhang, H. and Arora, A. (2002) 'GS3: scalable self-configuration and self-healing in wireless networks', *ACM Symposium on Principles of Distributed Computing*.