ZONER: A ZONE-based Sensor Relocation Protocol for Mobile Sensor Networks

Xu Li
School of Computer Science
Carleton University, Canada
Email: xlii@connect.carleton.ca

Nicola Santoro
School of Computer Science
Carleton University, Canada
Email: santoro@scs.carleton.ca

***Abstract*— In mobile sensor networks, self-deployment and relocation are two different research issues, both of which involve autonomous sensor movement. They share in most cases a common goal, that is, to improve overall network sensing coverage. Under this circumstance, some self-deployment algorithms may be applied to solving relocation problem without modification. However, considering efficiency, they will not be a good option in the scenario with high sensor failure rate. Existing sensor relocation protocols are not quite practical because they rely on strong assumptions and/or have weakness in maintaining network topology. In this paper, we propose a distributed zone-based sensor relocation protocol, ZONER, for mobile sensor networks on the basis of a restricted flooding technique, i.e., ZFlooding. Requiring zero-knowledge about sensor field, the ZONER is able to effectively discover previously-deployed redundant sensors without being concerned with obstacles or network ununiformity, and it relocates them in a shifting way to replace failed non-redundant ones without changing network topology. At the end of the paper, we prove the correctness of the ZONER and point out our future work.**

I. Introduction

Mobile sensor networks (MSNs) is a new paradigm of wireless sensor networks (WSNs). They inherit all the features such as narrow bandwidth, limited lifetime, restricted computational capability, multi-hop communication, etc. from conventional WSNs, and they are meanwhile known for their own particularity – *node mobility*. As MSNs emerge, some new research issues motivated by the powerful locomotion property of mobile sensors come into the radar screen of sensitive researchers. Two interesting ones are *sensor self-deployment* and *sensor relocation*. The former focuses on the way of converting a randomized sensor distribution to a uniform one without human assistance, while the latter concentrates on how to strategically move sensors to maintain certain network topology or to respond to some interesting events. In fact, their combination can provide a complete solution to the sensing coverage problem. For example, a self-deployment algorithm is carried out at the beginning of the operating period of a MSN to achieve a uniform sensor distribution, and a relocation protocol is conducted throughout the network lifetime to maintain previously-achieved network uniformity by healing the sensing holes due to node failure. Such an integrated scheme has been proposed by us in [1]. In this paper, we will focus only on sensor relocation problem.

Some existing sensor self-deployment algorithms[2], [3], [4], [5], [6] are adaptive to node failure and may actually be employed to solve the sensor relocation problem regarding sensing hole healing. These algorithms share the same philosophy, that is, uniform sensor distribution is achieved when certain network equilibrium state is reached. Unfortunately, the equilibrium that they go for is vulnerable to node failure. After a node fails, the network or a portion of the network may have to be re-organized in order to recover the equilibrium state and consequently heal the sensing hole caused by the failed node. Considering that sensors are usually dropped in hostile and/or unknown environment where node failure is a common phenomenon, such a sensor self-deployment algorithm is costly because frequent topology change not only complexes networking protocols but leads to energy loss.

To our best knowledge, only two sensor relocation algorithms, a proxy-based protocol[7], which is an enhanced version of the early work presented in [8], and a Grid-Quorum based protocol[9] are proposed in literature. The proxy-based protocol fills large sensing holes by relocating some mobile nodes that contribute relatively less to overall coverage. Its disadvantages are the resulting frequent network topology change and uniformity degradation. The Grid-Quorum based protocol patches sensing holes with redundant nodes that are discovered according to certain policy and relocated in a cascading manner. This protocol solves the sensing hole problem without sacrificing existing network uniformity if the number of redundant nodes are sufficiently large. However, its strong assumption, i.e., pre-knowledge about sensor field, makes it less practical in real-world scenarios.

In this paper, under the assumption of global coordinates, we introduce a restricted flooding technique, Zone Flooding (or, ZFlooding for short), featured with void-area penetration capability, and then based on it, we propose a distributed zone-based sensor relocation protocol, ZONER, using sufficient redundant nodes. By the ZONER, each redundant node registers itself with all the non-redundant nodes inside a vertical *registration zone* across the entire network; when a node fails, its specified neighbors inquiry all the non-redundant nodes inside a bounded horizontal *request zone* for redundant nodes; because the request zone intersects with a number of registration zones, the non-redundant nodes in the intersection areas can provide the requester with redundant node information; once a satisfactory and available redundant node is identified,

it will be relocated in a shifting way to replace the failed node with no change in network topology. Because no network-wide flooding is used, and because only selected nodes are required to move for replacing purpose, the ZONER is both bandwidth and energy efficient. Although the ZONER and the Grid-Quorum based protocol[9] have similarity in their node discovery (in fact, both of them are a variant of the quorum based location service[10], [11] in the context of sensor relocation) and relocation methods, they differ a lot from each other in that the ZONER does not require any pre-knowledge about the sensor field and has immunity to void-areas (caused by obstacles or unbalanced node distribution) during node registration and node discovery processes.

The rest of this paper is organized as follows: Section II briefly reviews the previous work on sensor relocation in literature; Section III introduces the ZFlooding technique; Section IV presents the details of the ZONER; Section V proves the correctness of the ZONER; Section VI concludes the paper and points out our future work.

II. Related Work

Wang, Cao and Porta[7] presented a proxy-based sensor relocation protocol for the sensor networks composed of both static nodes and mobiles. It assumes global coordinates and location-awareness. For an arbitrary mobile node, the protocol estimates the size of the coverage hole generated by the node in the case that the node leaves its current location, and assigns the node a base price accordingly. Each static node independently identifies coverage holes based on Voronoi diagram[12] and bids the closest mobile node with smallest base price. In the case that a mobile node receives multiple bidding messages from different static nodes, it is bid by the bidding message with largest hole size and then moves to fill the corresponding hole. Hence, mobile nodes always intend to move to large holes from small ones, and they stay still only when no larger holes can be detected. To save energy, a mobile node logically moves to its target location by choosing proxy node, which then executes the protocol on behalf of it as if it already moved to that location; actual movement is performed only when its target location is the final location.

Wang, Cao, Porta and Zhang[9] proposed a Grid-Quorum based sensor relocation protocol under the assumptions of global coordinates, location-awareness, and known sensor field. In this protocol, the network field is geographically partitioned into grids. In each grid, one node is elected as grid head and takes the responsibility to collect the location of all the grid members. Based on grid members' location, a grid head determines redundant grid members and detects sensing holds. A grid row is called *demand quorum*, while a grid column is called *supply quorum*. Each grid head publishes the information about the redundant nodes inside its grid to all the grid heads in the supply quorum that it is residing in. When a grid head detects a sensing hole, it broadcasts a request within its demand quorum to discovery the closest redundant node. Because every demand quorum intersects with all the supply quorums, a redundant can always be found if any exists. To reduce delay and balance power consumption, a redundant node is located in a cascaded way. Namely, the nodes along a relocation path from the redundant node to the failed node move to the location of their successors simultaneously. Relocation paths are carefully selected to minimize the difference between the total power consumption and the minimum remaining power of cascading nodes.

III. A Restricted Flooding Technique

Flooding as a basic networking technique is widely applied to a variety of network operations such as routing and service discovery. Because network-wide flooding involves the entire set of network nodes,its application is quite controversial in resource-restricted networks like MSNs. Under this circumstance, restricted flooding technique is being studied. Assuming a global coordinate system and nodes' awareness of their own coordinates, a restricted flooding algorithm could be as simple as follows: a node starts a flooding process by broadcasting a packet carrying the boundary information of the area to be flooded; a receiver node retransmits (by broadcast) the packet if and only if it is inside the specified flooding area; the flooding process terminates after all the nodes in the flooding area obtain the packet. On the basis of this simple algorithm, we devise a restricted flooding technique, Zone Flooding (ZFlooding). The novelty of the ZFlooding is its void-area penetration ability.

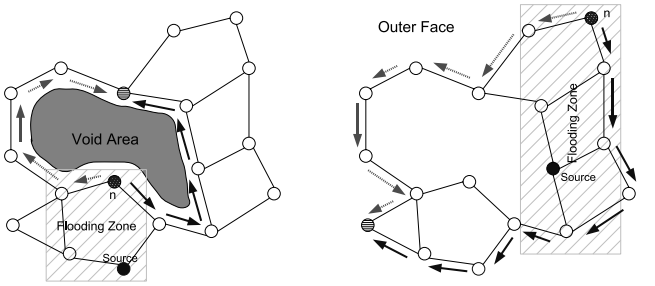
A. Planarizing Network Graph

A graph is a planar graph iff any two edges either do not intersect or intersect only at their common end vertex. A typical example of planar graphs is *Gabriel Graph (GG)*, where the closed diametral disc of each edge contains no other vertices than the two edge ends[13]. A GG-construction algorithm, which takes a connected graph G as input and outputs a GG, G', spanning G, is as follows: remove non-GG edges from G by testing every edge using the GG definition; an edge e remains in G iff it passes the GG test; finally, G becomes G'. Hence, a GG can be easily built over a connected network in a localized fashion as long as each network node knows about the coordinate of its every neighboring node.

Other localized planar graphs than GG include *Relative Neighborhood Graph (RNG)* and *Localized Delaunay Triangulation (LDT)*. Due to their planarity, connectivity and easy construction, these graphs are good options for supporting face routing, a proven effective building block for conquering the well known dead-end problem in geographic routing[14], [15]. Similarly, the ZFlooding employs face routing to penetrate void areas.

B. Penetrating Void Areas

A node is said to be a local minimum in some direction (west, east, north, or south) if it is the foremost node in that direction. For a local minimum n in direction d, its two incidental edges, which respectively have the smallest angle and the largest angle with d, identify a particular face in the GG constructed over the underlying network. This face



(a) The scenario of traversing an inner face.

(b) The scenario of traversing the outer face.

Fig. 1. An illustration of closed traversal circles

contains a void area that may stop the message transmission going through n in direction d, and to simplify expression, we call it d-face. Before retransmitting a received ZFlooding packet Pkt, n replicates it and stores the replica, denoted by Pkt', locally. After retransmission, n attaches its own coordinate to Pkt' and then sends two copies of Pkt', one in the clockwise direction and the other in the counterclockwise direction, to traverse its d-face along the face perimeter. Let FZ denote the flooding zone indicated by the Pkt/Pkt'. For a perimeter node x receiving Pkt' for the first time, it checks if it itself is inside FZ and meanwhile is more foremost than n in direction d, or if any of its neighbors satisfies the two conditions. If the answer is "yes", it terminates the face traversal process as a *terminator*, or otherwise forwards the Pkt to the next perimeter node in the traversal direction as a *forwarder*. During the face traversal process, if a forwarder did not ever receive Pkt before, it also takes the following extra actions: recover Pkt by removing the coordinate of node n from Pkt' and then broadcast Pkt locally.

If node n is the foremost in direction d in the flooding zone FZ, its attempt to pass the ZFlooding packet Pkt around its d-face will end up a closed traversal circle as shown in Figure 1. Observe that, if n is by any chance the foremost in direction d in the entire network, Pkt actually traversed the outer face as displayed in Figure 1(b). In this case, provided each boundary node is aware of the fact that it itself is a boundary node, the face traversal process can be terminated earlier, and thus saving both bandwidth and energy. To enable this early termination, a separate (or, integrated) boundary detection process should be performed after the GG construction. However, this optimization is not included in our current work but left for future study.

C. Packet Format

A ZFlooding packet consists of a header and a payload part. The payload part contains communication data, the thus its format is application dependent. As for the packet header, it is defined as a ten-field tuple:

$$\langle S, WB, EB, NB, SB, LM, DIR, TD, NH, TTL \rangle .$$

Field S contains the ID of the source node. The four fields WB, EB, NB, and SB are together called flooding boundary fields. The zone to be flood is defined as a square area surrounded by four lines, $x = a$, $x = b$, $y = c$, and $y = d$ ($a, b, c, d \in \Re \cup \{\infty\}$), and the flooding boundary fields respectively record these four parameters. Field LM stores the coordinate of a locally foremost node in the direction indicated by Field DIR. Field TD implies traversal direction, and its value could be either CLOCKWISE or COUNTERCLOCKWISE. By default, Field NH is set to BROADCAST which indicates that the packet is in broadcast mode. Its value is alerted by the local optimum presented by the two fields LM and DIR. If the packet is in face traversal mode, NH stores the coordinate of the next-hop node. Note that the fields LM, DIR, and TD are ignored when NH is set to BROADCAST. Field TTL specifies how far a packet can go in hop counts. Before a ZFlooding packet is forwarded, the value of its TTL field is decremented. When its TTL decreases to 0, the packet will be dropped rather than forwarded.

IV. Zone-based Sensor Relocation

In this section, we will present the zone-based sensor relocation protocol, ZONER. Some terminologies used in the rest of the paper can be found in the following table.

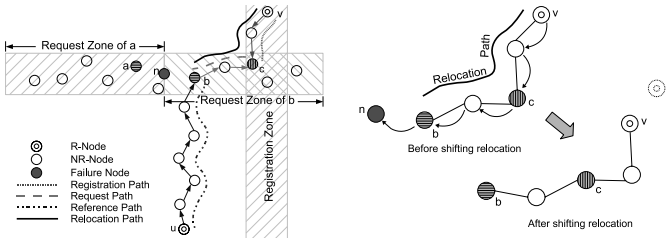
TERMINOLOGY	DESCRIPTION
Recommender	A NR-node that replies the R-node request of another NR-node.
Registration Zone	A horizontally bounded and vertically unbounded area where a R-node registers with all the inside NR-nodes.
Request Zone	A bounded square area where a NR-node inquires all the inside NR-nodes for R-node information.
Registration Path	A path along which a R-node registers with a NR-node.
Request Path	A path along which a NR-node is asked for R-node information by another NR-node.
Relocation Path	The accumulation of a registration path and a request path linked by a recommender node.
Path Length	A pair of values (Len, Cnt), where Len is the sum of the Euclidean distance between every two neighboring node in the path, and Cnt is the hop count of the path.

A. Network Model

Non-redundant nodes (referred to as NR-nodes) are randomly distributed in a plane and form a connected network through bidirectional communication links using a geographic routing protocol such as GFG[14]. Sufficient redundant nodes (referred to as R-nodes) are scattered in the network, but they do not participate in any network operation except the ZONER. All the nodes use omni-directional antenna for communication, and their communication radii are at least twice their sensing radii. Every node is assigned a unique ID and aware of its global coordinate, and it has the ability to move upon request. Any node may fail, but the rest of the network remains connected.

B. Overview

A NR-node maintains a one-hop neighborhood map by listening to a periodical HELLO message from its every



(a) Discovery (b) Relocation

Fig. 2. An illustration of how the ZONER works

neighboring NR-node, and transmits this map to all its immediate NR-node neighbors on a regular basis. By merging the received one-hop neighborhood maps with its own, every NR-node actually keeps a two-hop neighborhood map throughout network lifetime. At the beginning of network operation (or, when necessary), a R-node floods its *registration zone* with a registration message to register with all the NR-nodes inside the zone. After a NR-node failed, its westmost neighbor and eastmost neighbor respectively start a discovery process by flooding their *request zones* with a request message to find a replacement for it. The two default process initiators are called *discovery partner* of each other, and their request zones are adjacent by an imaginary line vertically across the failed node. During a discovery process, the initiator first locally searches for the registered R-node with shortest relocation path and then takes this R-node as reference to inquires all the NR-nodes inside its request zone for the R-nodes with yet shorter relocation path. For message-saving purpose, the length of the request zone is made subject to the reference node's relocation path length. Because the request zone intersects with a number of registration zones, the NR-nodes in the intersection areas may be able to reply the initiator's request as *recommender*. Finally, the initiator chooses the one with shortest relocation path among all the discovered available R-nodes as the failure node's replacement. Figure 2(a) is a big picture about a discovery process. Sequentially, the replacement discoverer triggers a relocation process by a relocation message. In this process, the nodes along the replacement node's relocation path relocate in a shifting manner to replace the failed node. That is, every node in the path simultaneously moves to the location of its path neighbor towards the replacement node discoverer, and the replacement discoverer moves to the location of the failed node as illustrated in Figure 2(b). After such a relocation process, the failed node is in fact replaced by the discoverer of the replacement node rather than by the replacement node itself. Note that, for any R-node, once it is actually involved in a relocation process, it transforms to a NR-node.

C. Data Definitions

A number of control messages and data structures are defined by our protocol. They play an essential role in coordinating nodes and helping accomplish protocol goal. Below, we shall introduce these data definitions.

1) Control Messages: There are four main types of control messages, i.e., Node Registration (NREG) message, Node Request (NREQ) message, Node Reply (NREP) message and Node Relocation (NREL) message.

NREG Message is used by a R-node to register itself with all the NR-node in a pre-defined registration zone. It is transmitted in a ZFlooding manner. The ZFlooding header of a NREG message always has the following settings: $NB = SB = \infty$, $WB = x^* - \alpha$, $EB = x^* + \alpha$, and $TTL = \infty$, where x^* is the X element of the source node's coordinate, and $\alpha \in \Re^+$ is a system parameter. These settings imply that a registration zone is a long zone vertically across entire network. Other than a ZFlooding header, a NREG message has three payload fields: $SeqNo$, $PriHop$, and $PathLen$. $SeqNo$ stores the sequence number of the registration process initiated by the source node; $PriHop$ records the ID of the node that most recently transmitted the message; ***PathLen*** contains the registration path length of the source node.

NREQ Message is used by a NR-node to inquire all the NR-nodes in a specified request zone for satisfactory R-nodes. It is transmitted in a ZFlooding manner. A NREQ message has four payload fields: $SeqNo$, $RefPathLen$, $PriHop$, and $PathLen$. $SeqNo$ stores the sequence number of the discovery process initiated by the source; $RefPathLen$ stores the length of a reference path, which is used to estimate whether a R-node is satisfactory (a R-node is said to be satisfactory iff its relocation path length is smaller than $RefPathLen$); $PriHop$ and $PathLen$ respectively record the ID and the request path length of the node that most recently transmitted the message. In the ZFlooding header, define $WB = x^* - RefPathLen.Len/2$, $EB = x^* + RefPathLen.Len/2$, $NB = y^* + \beta$, $SB = y^* - \beta$, and $TTL = RefPathLen.Cnt$, where (x^*, y^*) is the coordinate of the source, and β is a system parameter.

NREP Message is used by a NR-node to reply the NREQ message originated from another NR-node. It is transmitted along the request path of the source node to the destined NR-node. A NREP message carries the information about a R-node whose relocation path length is smaller than the reference path length contained in the NREQ message. It consists of five fields: $Source$, $Destination$, $SeqNo$, RID, $RelPathLen$, and $PriorHop$. $Source$ and $Destination$ respectively contain the IDs of the source and the destination; $SeqNo$ contains the sequence number of the discovery process that the message belongs to; RID and $RelPathLen$ respectively record the ID and the relocation path length of a satisfactory R-node; $PriorHop$ stores the ID of the node that most recently transmitted the message.

NREL Message is used by the discoverer of a R-node to notify all the nodes along the relocation path of the R-node to relocate. It also serves as a notification for other NR-nodes in the request zone of the R-node discoverer to release their resources. A NREL message is transmitted in a ZFlooding manner, and its ZFlooding header is configured as that of a NREQ message. Other than a ZFlooding header, a NREL message has one payload field, i.e., $SeqNo$, which contains

the sequence number of the discovery process during which the R-node node is discovered.

2) Data Structures: There are tree main types of data structures, i.e., Sequence Number (SeqNo), Registration Table (RegTab), and Request Buffer (RegBuf).

SeqNo is a monotonically increasing number locally maintained by each node. A node associates its current SeqNo with its registration/discovery process to be initiated, and it increments its SeqNo after the process is actually started. As a result, any two different registration/discovery processes started by the same node have distinct SeqNos, and for a particular process, the higher its SeqNo, the more recently it is initiated. In this sense, SeqNo implies the freshness of a registration/discovery process, and its combination with initiator ID uniquely identify such a process.

RegTab is a local structure maintained by each NR-node. It holds the information of every registered R-node. An entry, containing four fields ID, $SeqNo$, $PriorHop$, and $PathLen$, of a RegTab represents a particular R-node. ID and $SeqNo$ respectively contain the ID and registration process SeqNo of the R-node; $PriorHop$ records the hosting node's the prior hop in the R-node's registration path; $PathLen$ keeps the registration path length of the R-node to the hosting node.

RegBuf is a local structure maintained by each NR-node. For any NR-node, its RegBuf stores the information about the discovery process which it is currently participating in. If it is not involved in any discovery process, its RegBuf is empty. A RegBuf consists of six fields, i.e., ID, $SeqNo$, NID, $NPOS$, $PriorHop$, and $PathLen$. ID stores the ID of the NR-node that initiates the discovery process; $SeqNo$ records the SeqNo of the discovery process; NID and $NPOS$ respectively records the ID and the coordinate of the failed node that the discovery process is serving; $PriorHop$ records the hosting node's prior hop in its request path; $PathLen$ keeps its request path length.

D. Protocol Core

The three processes, i.e., *registration*, *discovery*, and *relocation*, constitute the core of the ZONER. In the sequel, we are going to elaborate on the three processes.

1) Registration Process: Consider an arbitrary R-node r. When r wants to register within its registration zone, it generates a NREG message Msg_{reg}, broadcasts the message to all its neighbors, and then increments its SeqNo. After a NR-node n in the registration zone receives Msg_{reg}, it computes the registration path length of r, updates the $PathLen$ field of Msg_{reg} with the computation result, and searches its RegTab for r's entry. There are four possible cases to be examined: (1) r does not have an entry in the RegTab; (2) r has an entry in the RegTab, but the content is outdated; (3) r has an entry in the RegTab, and the content is up-to-date, but the recorded registration path is longer than the new one; (4) otherwise. In Case (1), n creates an entry for r in its RegTab with the information carried by Msg_{reg}; in Case (2) and (3), n updates r's entry; in Case (4), n simply drops Msg_{reg}. After processing any of above four possible cases, if Msg_{reg} has not been dropped, n updates the $PriHop$ field of Msg_{reg} with its own ID and then continue to process Msg_{reg} following ZFlooding rules.

2) Discovery Process: A discovery process is composed of four successive stages: *local search*, *remote search*, *hold*, and *selection*. Taking a NR-node n_i, which is discovering a replacement for a failed neighboring NR-node n_f, as an example, we describe these four stages in detail.

Local Search Stage: In this stage, node n_i first looks up its local RegTab for the closest, available, and registered R-node, denoted by r_{ref}. Let $PLen_{ref}$ denote the length of r_{ref}'s registration path. In the case that r_{ref} is null, $PLen_{ref}$ is defined as ∞. Recall that a path length is composed of two elements, i.e., the Euclidean length (referred to as Len) and the hop count (referred to as Cnt). Then, n_i generates a NREQ message Msg_{req} with $RefPathLen = PLen_{ref}$ and $TTL = PLen_{ref}.Cnt$, increments its SeqNo, and sends this message to its neighbors. Afterwards, the discovery process enters the *remote search* stage.

Remote Search Stage: Each NR-node inside the request zone takes part in the discovery process (specifically, the remote search stage) by processing its received Msg_{req}. However, participation[1] is not mandatory. Suppose that a NR-node n_x in n_i's request zone receives Msg_{req} and is willing to participate in the discovery process. n_x computes its request path length and checks if its ReqBuf if empty. In the case that the ReqBuf is empty, n_x bookkeeps the information about the discovery process in its RegBuf and finds the most satisfactory R-node r_{min} from its RegTab. Afterwards, it sends a NREP message carrying r_{min}'s information back to n along its reverse request path as recommender if r_{min} is not null. We would like to indicate that each intermediate node in this reverse request path need to remember the node from which it receives the NREP message so that the forward request path can be established. In the case that the ReqBuf is not empty, if n_x is currently involved in the same discovery process and its recorded request path length is smaller than the new one, n_x updates the buffer with the data carried by Msg_{req}, or simply drops Msg_{req} otherwise. After dealing with either of above two cases, if Msg_{reg} has not been dropped, n_x updates the $PriHop$ and $PathLen$ fields of Msg_{req} respectively with its own ID and its request path length, and then continues to process Msg_{req} following ZFlooding rules. Considering discovery failure, a NR-node empties its ReqBuf if no more process messages are received in a predefined waiting period.

Hold Stage: In this stage, node n_i locally records the first-discovered m R-nodes in a node list in an increasing order of their relocation path length. Denote by C_i the i-th node in the list and by $R(C_i)$ the recommender of C_i . Node n_i sends a HOLD message to C_1 along C_1's relocation path and expect a reply from it. When the HOLD message reaches $R(C_1)$, if C_1 has already canceled its registration, $R(C_1)$ blocks the hold request and replies n_i with a NO message on behalf of C_1,

[1]The decision can be make based on certain policy that takes into consideration nodal remaining energy level and/or other application specific requirements.

or re-transmits the HOLD message to C_1 along C_1' reverse registration path otherwise. Mentionably, during the process that the HOLD message is transmitted to C_1 from $R(C_1)$, each intermediate node along the path need to remember the node that it just receives the message from so that C_1's forward registration path can be established. Recall that the forward request path of $R(C_1)$ is constructed in previous remote search stage. After C_1 receives the HOLD message, C_1's relocation path becomes complete. If C_1 is not in "held" status, it will mark itself as "held" and answers n with a YES message, or replies with a NO message otherwise. Once C_1 becomes held, it will no longer grant hold request from any other NR-node. If n does not receive any reply from C_1 after a predefined number of hold request trials, or if it receives a NO message, it will try to hold C_2 in the same way. n_i keeps doing so until a R-node C_i replies its hold request with a YES message. Then, C_i is taken by n_i as the replacement candidate of n_f. In the case that none of those discovered R-nodes make a positive reply, n_i has to re-start the entire discovery process once again within a size-increased request zone. Since there are sufficient R-nodes in the network, a replacement candidate will be eventually identified.

Selection Stage: The objective of this stage is to choose the official replacement from discovered replacement candidates. If n_i does not have a *discovery partner*, or if its *discovery partner* has failed, the discovered replacement candidate automatically becomes the official replacement of n_f. Otherwise, n_i and its discovery partner exchange their discovery results by making use of the underlying routing protocol and then independently determine n_f's official replacement, which is the replacement candidate with shorter relocation path.

3) Relocation Process: Let us continue with previous example. Denote by n_r the official replacement of n_f. Assume that node n_i is the discoverer of n_r. To start a relocation process, n_i generates a NREL message Msg_{rel} carrying the SeqNo of previous discovery process and broadcasts the message within its request zone. For a NR-node n_x in n_i's request zone, when it receives Msg_{rel}, it must be in one of the following four situations: (1) it is not participating in the discovery process; (2) it is involved in the discovery process but not in the relocation path of n_r; (3) it appears in the relocation path, but it is not the recommender of n_r; (4) it is the recommender of n_r. To simplify expression, we define the prior hop of a node in a replacement node's relocation path as the path neighbor towards that replacement node, and the next hop as the path neighbor in the opposite direction. In Case (1), n_x simply discards Msg_{rel}; in Case (2), n_x processes Msg_{rel} following ZFlooding rules and then clears its ReqBuf; in Case (3), n_x processes Msg_{rel} following ZFlooding rules, clears its ReqBuf, and then moves to the location of its next hop in n_r's relocation path; in Case (4), n_x forwards Msg_{rel} along n_r's relocation path to n_r, processes Msg_{rel} following ZFlooding rules, clears its ReqBuf, and then moves to the location of its next hop along n_r's relocation path. Before the replacement node n_r moves, it informs all the NR-nodes inside its registration zone via ZFlooding technique to remove its registration information. After shifting relocation, all the nodes in the relocation path fill their next hops's shoes. Under this circumstance, they must respectively pass their local data to their prior hops in order to restore the normal execution of the networking protocols and applications running on each nodes. As for the replacement node discoverer, after arriving at its target location, i.e., the location of the failed node, it has to ask its neighbors for necessary data since the failed node will not be able to pass it anything.

Remark: *The ZONER is both bandwidth and energy efficient.* The main communication cost of the ZONER is due to the flooding operation in its registration, discovery, and relocation processes. Recall that the communication in these three processes is confined within pre-defined bounded flooding zones. Although registration zones are vertically unbounded, a registration process is executed by each R-Node only once (or, merely on an occasional base). Consider the energy consumption due to nodal movement. No particular node will over consume its battery power since moving distance are distributed to multiple nodes according to the shifting relocation strategy, and thus prolonging network lifetime.

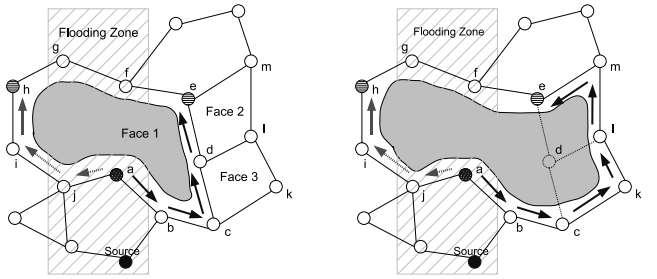
E. Fault Tolerance

The execution of the ZONER is vulnerable to node failures. In order to improve its availability and robustness, we equip the ZONER with a fault tolerance mechanism discussed below.

1) Tolerating node failure during face traversal: In a Gabriel Graph (GG), after a node fails, the faces adjacent by that node merge and form a larger face. This type of face merging naturally tolerates perimeter node failure during a face traversal process. Let us take the scenario in Figure 3 as an example. During a ZFlooding process, the ZFlooding packet Pkt is switched by node a to face traversal mode. Ideally, the face traversal path of Pkt is $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ around Face 1 as shown in Figure 3(a). After reaching node e, Pkt is switched back to broadcast mode because e's neighbor f is inside the flooding zone. However, if node d has failed, Pkt will be transmitted along a path different from the expected one. Specifically, having detected node d's failure[2], node c considers the merging face $LFace$ of Face 1, 2, and 3 as the designated face and forwards Pkt to node k instead of d, and Pkt's actual face traversal path becomes $a \rightarrow b \rightarrow c \rightarrow k \rightarrow l \rightarrow m \rightarrow e$, around $LFace$ as displayed in Figure 3(b). Nevertheless, Pkt still successfully penetrates the void-area (in this particular example, Pkt ends up with node e).

The parallel execution of node replacing can have side-effect on face traversal. In the previous example, when the failed node d is replaced by a node d', the large face $LFace$ is split back into the three small faces, Face 1, 2, and 3. If this node replacing takes place before node l transmits the packet Pkt, the recovered Face 3 becomes current face, and thus l will forward Pkt to d', which then sends the packet to c, therefore generating a unexpected traversal loop as shown in

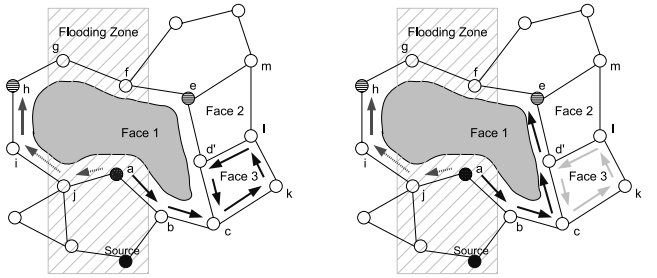
[2]Node failure can be detected either by listening to the periodical HELLO message or by monitoring a node's communication activities.



(a) Face traversal with no perimeter node failure

(b) Face traversal with perimeter node failure at d

Fig. 3. An example of tolerating node failure during face traversal



(a) Traversal loop due to failure-node replacing

(b) Face traversal resumption at node c

Fig. 4. An illustration of the side-effect from node replacing

Figure 4(a). After node c receives Pkt from d', it will be aware of the loop and consequently terminates face traversal. This unexpected early termination could lead to the failure of void-area penetration. To deal with this problem, each node needs to records not only its actual next hop but also its expected next hop when transmitting packets in face traversal mode. E.g., the expected next hop of c is d, while its actual next hop is k. To be adaptive to node replacing, the two type of next hops should be stored in the form of coordinates rather than IDs. In addition, each node needs to back up a face-traversing packet before transmitting it. For a node participating in a face traversal, in the case that its actual next hop is different from the expected one, if it receives the face-traversing packet back from the expected next hop, it considers there is a traversal loop due to node replacing and resumes the original face traversal using the backup packet as shown in Figure 4(b).

2) Tolerating node failure during node replacing: Consider an arbitrary failed NR-node n_f. If the two default discovery process initiators, namely the westmost neighbor $WN(n_f)$ and eastmost neighbor $EN(n_f)$ of n_f, have both failed, n will never be replaced according to previous protocol description. To conquer this *failed initiators* problem, while $WN(n_f)$ and $EN(n_f)$ are looking for a replacement for n_f, other neighbors of n_f keep monitoring their existence by periodically sending them a beacon message through the underlying routing protocol and expecting their reply. If $WN(n_f)$ and $EN(n_f)$ have both failed, and if the failed node n_f has not yet been replaced, the westmost, $WN'(n_f)$, and the eastmost, $EN'(n_f)$, among n_f's functioning neighbors will automatically take over $WN(n_f)$ and $EN(n_f)$'s responsibility and respectively start a discovery process. Meanwhile, the rest of n_f's live neighbors will turn to monitor $WN'(n_f)$ and $EN'(n_f)$. This type of monitoring and taking-over keeps going until n_f is successfully replaced or all its neighbors fail.

An extreme case of the *failed initiators* problem is that all the neighbors of the failure node n_f have failed. Assume that all these failed neighbors are successfully replaced. Because the neighbor replacements have no knowledge about n_f, they are not able to start a discovery process for it, resulting in that n_f will never be replaced. To handle this *failed neighborhood* problem, the ZONER requires that the NR-nodes neighboring n_f but having no knowledge about n_f and its neighborhood independently discover a replacement for n_f all alone (without any discovery partner). By this requirement, the failure node n_f may have more than two discovery process initiators, and therefore multiple replacements at the end.

During a relocation process, the blank spot due to a NR-node's leaving is not treated as a sensing hole since some other node is supposed to cover that spot soon. However, if a relocating NR-node/R-node fails before it arrives at its target location, the blank spot at its target location turns into a sensing hole as a result. This *failed relocating nodes* situation can be identified in the following way: a NR-node informs all its neighbors before it moves of the estimated time period in which its location could be occupied by another node; its neighbors checks if its original location is still uncovered after that time period and then makes its decision accordingly.

F. Collision Resolution

When more than one node, due to the utilization of the fault-tolerance mechanism, initiates discovery processes for a single failure node, multiple replacement nodes may be discovered and relocated, resulting in node collision and consequent energy loss. A collision resolution method, *Preliminary Replacement Attempt (PRA)*, is integrated within a relocation process by the ZONER.

Specifically, the discoverer $Disc(n_r)$ of the replacement node n_r of a failed NR-node n_f delegates its only neighbor n_x in the relocation path of n_r to initiate a deferred relocation process, while it itself performs a PRA. In the PRA, $Disc(n_r)$ attempts to replace n_f by moving towards it; while moving, $Disc(n_r)$ constantly broadcasts a message carrying its ID and n_f's coordinate; if it hears such a message carrying a smaller ID and the same coordinate, or if it finds that its target location has already been occupied, $Disc(n_r)$ will return to its original location. While $Disc(n_r)$ is performing this PRA, all the other nodes in n_r's relocation path stay put. In particular, n_x keeps monitoring the return action of $Disc(n_r)$ during this waiting period. If it finds that $Disc(n_r)$ actually returns, it informs all the nodes involved in the discovery process, in which n_r is discovered, to release their sources by a cancellation message. The cancellation message is processed by each receiver in a way similar to that a relocation message is handled. If n_x finds that $Disc(n_r)$ does not return after the waiting period, it

initiates a relocation process on behalf of $Disc(n_r)$. By this means, although collision can still occur, excrescent moves and misspent moving distance are greatly reduced.

V. Proof of Correctness

In an ideal environment where no failure happens to the nodes involved in current protocol execution:

Lemma 1: *A replacement node can always be found for a failed non-redundant node.*
Proof: Redundant nodes register themselves with all the non-redundant node within its predefined vertical registration zone. The neighbors of a failed node inquire all the non-redundant nodes inside their horizontal request zones for redundant node. The request zones intersect with a number of registration zones. The non-redundant node in the intersection area is able to provide redundant node information. In case that no available redundant node can be found, the size of the request zones will be increased. When the size is larger than the area of the sensor field, all the non-redundant nodes will be asked in the network. According to the assumption of sufficient redundant nodes, an available redundant node will be eventually discovered. Hence, Lemma 1 holds.

Lemma 2: *A failed non-redundant node can be replaced once a replacement node is discovered.*
Proof: When a replacement node is determined by the two default discovery process initiators, the communication path between the replacement node and its discoverer forms a nature relocation path. By the shifting relocation method, after the relocation process, the failed node is replaced by the replacement discoverer. Hence, Lemma 2 holds.

In a real-world scenario where failure may occur at any node at any given time:

Lemma 3: *The ZONER tolerates node failures.*
Proof: In a registration process, node failure will not affect the initiator's registration operation since the network is always connected. In a discovery process, if a NR-node inside the request zone fails, it will not appear in the relocation path of the finally discovered R-node; if the initiator fails, its discovery partner is still going to find a R-node anyway; if both of the two default initiators fail, two other functioning NR-nodes will take over their discovery and relocation responsibility; if all the NR-node neighbors of the failure node fail, the replacements of these nodes are required to find a R-node for the failure node. In a relocation process, if a relocating node fails before arriving its target location, the NR-nodes around its target location can find out and then find a R-node to cover the blank spot. Hence, Lemma 3 holds.

Lemma 4: *The ZONER is able to solve node collision.*
Proof: Its correctness follows the fact that no collision will occur if a node initiates a relocation process iff it is sure about that there is no other relocation process is being or will be conducted for the same failure node.

By Lemma 1, 2, 3 and 4, we have the following theorem: **The ZONER is able to effectively patch sensing holes due to node failures.**

VI. Conclusion and Future Work

In this paper, we presented a distributed zone-based sensor relocation protocol, ZONER, for mobile sensor networks (MSNs) on the base of a restricted flooding technique, i.e., ZFlooding. The ZONER has a number of advantages: 1) benefiting from the zone-based node discovery strategy, it is able to quickly and efficiently discovery pre-deployed redundant nodes; 2) by using the shifting relocation method, energy consumption for node replacing is distributed into multiple nodes, and thus prolonging network lifetime; 3) due to the void-area penetration property of the ZFlooding technique, it is adaptive to obstacles and unbalanced node distribution, and it is able to accommodate dynamically added redundant nodes as well (this situation is not explored in this paper though); 4) thanks to the effective fault tolerance mechanism, it is robust against node failures; 5) because of the zero knowledge about the sensor field, it has strong availability in unknown environment.

In the future, we are going to implement the ZONER and evaluate its performance in comparison with the Grid-Quorum based protocol[9] by experiments.

Acknowledgments

The authors would like to thank Dr. Ivan Stojmenovic for his valuable discussions on improving this work, and the anonymous reviewers for their useful comments.

References

[1] X. Li and N. Santoro. "An Integrated Self-Deployment and Coverage Maintenance Scheme for Mobile Sensor Networks", 2006. Submitted for publication.

[2] 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. of DARS*, pp. 299-308, 2002.

[3] Y. Zou and K. Chakrabarty. "Sensor deployment and target localization in distributed sensor networks". *ACM Tran. on Embedded Computing Systems*, 3(1):61-91, 2004.

[4] G. Wang, G. Cao, and T. L. Porta. "Movement-Assisted Sensor Deployment". In *Proc. of IEEE INFOCOM*, vol. 4, pp. 2469-2479, 2004.

[5] N. Heo and P. K. Varshney. "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks". *IEEE Tran. on Systems, Man, and CyberNetics - Part A: Systems and Humans*, 35(1):78-92, 2005.

[6] J. Wu and S. Yang. "SMART: A Scan-Based Movement-Assisted Sensor Deployment Method in Wireless Sensor Networks". In *Proc. of IEEE INFOCOM*, vol. 4, pp. 2313- 2324, 2005.

[7] G. Wang, G. Cao, and T. L. Porta. "Proxy-Based Sensor Deployment for Mobile Sensor Networks". In *Proc. of IEEE MASS*, pp. 493-502, 2004.

[8] G. Wang, G. Cao, and T. L. Porta. "A Bidding protocol for deploying mobile sensors". In *Proc. of IEEE ICNP*, pp. 315-324, 2003.

[9] G. Wang, G. Cao, T. L. Porta, and W. Zhang. "Sensor Relocation in Mobile Sensor Networks". In *Proc. of IEEE INFOCOM*, pp. 2302-2312, 2005.

[10] I. Stojmenovic. "A scalable quorum based location update scheme for routing in ad hoc wireless networks". In *Technical Report*, SITE, Univ. of Ottawa, TR-99-09, 1999.

[11] D. Liu, I. Stojmenovic and X. Jia. "A scalable quorum based location service in ad hoc and sensor networks". In *Proc. of IEEE MASS*, 2006. To appear.

[12] F. Aurenhammer and R. Klein. "Voronoi Diagrams". Available online at "http://www.pi6.fernuni-hagen.de/publ/tr198.pdf".

[13] J. W. Jaromczyk and G. T. Toussaint. "Relative neighborhood graphs and their relatives". In *Proc. of the IEEE*, vol. 80, pp. 1502-1517, 1992.

[14] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks". In *Proc. of ACM DIALM*, pp. 48-55, 1999.

[15] H. Frey and I. Stojmenovic. "On Delivery Guarantees of Face and Combined Greedy-Face Routing Algorithms in Ad Hoc and Sensor Networks". In *Proc. of ACM MobiCom*, 2006. To appear.