Avoid "Void" in Geographic Routing for Data Aggregation in Sensor Networks

Shigang Chen[†] Guangbin Fan[‡] Jun-Hong Cui[§]

[†] Dept. of Computer & Information Science & Engineering, Univ. of Florida

[‡] Department of Computer Science, University of Mississippi

[§] Department of Computer Science & Engineering, University of Connecticut

Abstract — *Wireless sensor networks have attracted great attention in research and industrial development due to its fast-growing application potentials. New techniques must be developed for sensor networks due to their lack of infrastructure support and the constraints on computation capability, memory space, communication bandwidth, and above all, energy supply. To prolong the life time of a battery-powered sensor network, an energy efficient routing algorithm for data collection is essential. We propose a new geographic routing algorithm that forwards packets from sensors to base stations along efficient routes. The algorithm eliminates the voids that cause non-optimal routing paths in geographic routing. It replaces the right-hand rule by distance upgrading. It is fully distributed and responds to topology changes instantly with localized operations. We formally prove the correctness of the algorithm and evaluate its performance by simulations.*

I. Introduction

Sensor networks provide a critical infrastructure that can be quickly deployed to extract information from the surrounding environment [1]. Due to economic and deployment considerations, sensors are typically small with very limited storage and computation power. They run on batteries, which makes energy conservation a top issue. They communicate through short-range radio, which makes multi-hop routing a necessity in order for information to reach a remote destination.

For the vast majority of applications [2], [3], [4], [5], [6], the most important communication is from sensors to base stations, which is also the focus of this paper. A simple approach is to build a sink tree from the sensors to each base station. Specifically, a base station broadcasts a control message, and the reverse broadcast tree can be used by the sensors to route packets to the base station. The problem of this approach is the maintenance overhead due to topology changes. A sensor network evolves over time with existing nodes being demaged or running out of power while new nodes being deployed. A network may also be configured such that only a portion of the sensors are active at a time in order to conserve energy [7]. The sensors in the sleep mode do not participate in relaying packets. As sensors take turn to sleep and be active, the network topology changes. Therefore, periodic broadcasts are necessary to maintain the sink trees. There exists a tradeoff between the accuracy of routing information and the overhead of maintaining such information. Frequent broadcasts drain the power. Infrequent broadcasts may result in broken sink trees that are not in compliance with the underline topology. The same thing is also true for the flooding-based link state algorithms and the distance vector algorithms that employ progressive

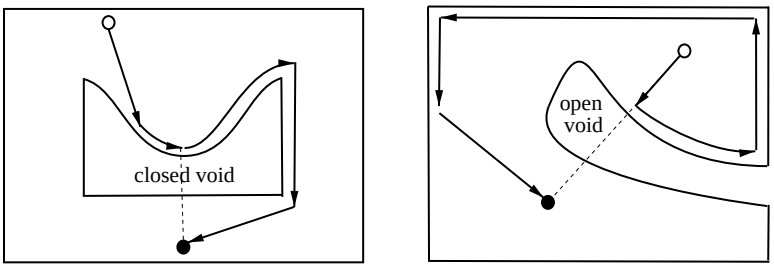

Fig. 1. Void and routing by the right-hand rule

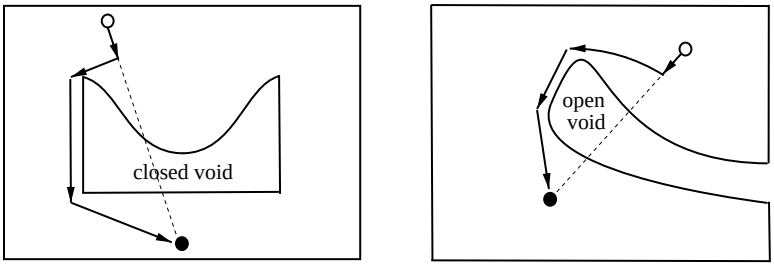

Fig. 2. Better routing paths

flooding with slow convergence.

One solution is geographic routing (BMSU [8], GPSR [9], SPEED [10], and FGG [11]), where each sensor knows the location coordinates of itself, its neighbors, and the base stations. The routing strategy of a sensor is simply to forward its packets to the neighbor that is closest to a base station. Because there is no routing information (other than the set of neighbors) to be maintained, the routing paths instantly adapt to the change of the topology. On the other hand, the geographic routing also has two major problems.

• First, it is not applicable if the sensors do not have the capability of knowing their location coordinates. This problem can be solved by virtual coordinate systems such as NoGeo [12], GEM [13], and BVR [14]. The virtual coordinates require sensors to know the hop distances to certain reference points, and the hop distances require periodic broadcasts to be kept up-to-date.

• Second, there may be void(s) between a source sensor and a base station, as illustrated in Figure 1. A void is an area that has no active sensors. A routing path based on geolocations may be blocked from moving closer to a base station due to the lack of relaying nodes to cross the void. The current solution is based on the "right-hand rule" [9], which routes a packet around the void. However, the right-hand rule may produce inefficient routing paths as shown in the figure, particularly for an open void.

This paper addresses the inefficiency problem of the right-hand rule in data aggregation from sensors to a set of base stations. We propose a new algorithm, called Distance Upgrading Algorithm (DUA), which avoids "voids" without the use of the right-hand rule. The basic idea is that, if a packet has to detour around a void from a sensor to a base station, then the effective distance between the sensor and the base station is larger than the Euclid distance. By appropriately upgrading the distances of some sensors, we are able to direct the packets along efficient routes towards the base stations, as illustrated in Figure 2.

The rest of the paper is organized as follows. Section II defines the sensor-network model and makes assumptions. Section III describes the void problem in geographic routing. Section IV presents the distance-upgrading algorithm. Section V presents the simulation results. Section VI draws the conclusion.

II. Assumptions

A sensor network consists of a set of sensors and a set of base stations. Two sensors are neighbors and can directly communicate if they are in the transmission range of each other. We assume each sensor knows the set of its neighbors via a neighbor discovering protocol. The sensors share the same wireless media, and each packet is transmitted as a local broadcast in the neighborhood. We assume the existence of a MAC protocol, which resolves the media contention and ensures that, among the neighbors in the local broadcast range, only the intended receiver keeps the packet and the other neighbors discard the packet. The sensors are statically located after deployment. We do not consider mobile sensors that form a dynamic ad-hoc network. We study the data packets sent from sensors to base stations. Assume that the base stations are connected via an external network to a data collection center. A data packet may be sent to any base station as long as there exists a routing path. Suppose a sensor knows the coordinates of its geolocation. This is a reasonable assumption because many monitoring applications require the knowledge of where the sensor data are generated, which in turn requires the sensors to know and report their geolocations [1].

III. The "Void" Problem in Geographic Routing

A. Geographic Routing

Consider a set N of sensors and a set B of base stations. After the sensors and the base stations are deployed, each base station broadcasts a welcome message including its location information. A sensor x learns the set of reachable base stations from the welcome messages. It calculates the distance $d(x)$ to the nearest base station. In the rest of the paper, when we say **"the distance of a sensor"**, we mean **"the distance of the sensor to its nearest base station"**.

Let N_x be the set of neighbors of x. Sensor x learns $d(y), \forall y \in N_x$, by exchanging the distance information with its neighbors. Consider the routing strategy of forwarding a packet to any neighbor with a smaller distance. This strategy makes sure that the packet will eventually reach one of the base station. The set of next-hop neighbors is defined as

$$R_x = \{y \mid d(y) < d(x), y \in N_x\} \tag{1}$$

We say there exists a *routing link* (x, y) if $y \in R_x$. [1] All routing links form the *routing graph*, which is acyclic. The graph allows multiple routing paths from a sensor to the base stations. x is called a *dead end* if $R_x = \emptyset$.

A greedy routing algorithm will always forward a packet to the neighbor with the smallest distance. Other routing algorithms may consider energy availability, congestion level, load balancing, and real-time requirements. These subjects are beyond the scope of this paper.

[1] Distance values of two neighbor nodes determine the direction of a routing link. For any routing link (x, y), $d(x) > d(y)$ by definition.

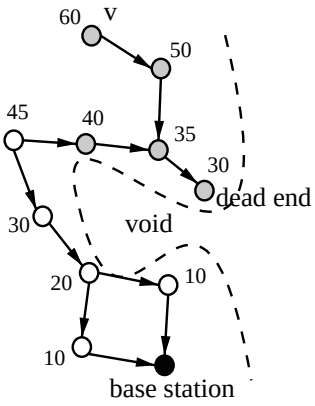

Fig. 3. Void and dead end

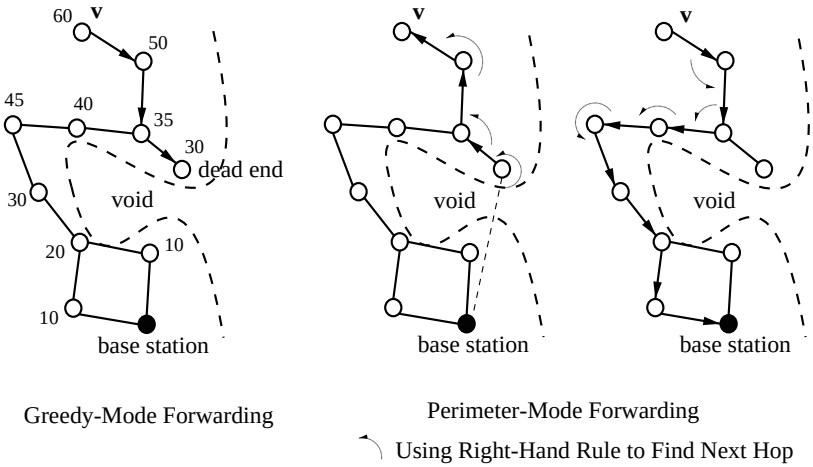

Fig. 4. Routing a packet in GPSR

B. Problem of Void

The problem of geographic routing is that packets may be routed to a dead end. Consider the example in Figure 3, where the white/gray circles represent sensors and the black circle represents a base station. The distance from a sensor to the base station is labeled beside the sensor. Suppose there is no sensor to the right of the dashed line, which outlines an open void. The void may be a disaster area where all sensors are destroyed, or it may be a bay where the sensors cannot survive. There is a single dead end, which does not have any neighbor that is closer to the base station. Once a packet is routed to the dead end, it cannot proceed any further. The packets from the gray nodes will be forwarded to the dead end based on geographic routing.

To solve the above problem, BMSU [8], GPSR [9], and FGG [11] use the "right-hand rule" to route packets along the boundary of the void until they reach the other side of the void. This approach finds a specific detour path out of the dead end. Figure 4 shows how a packet from v is routed by GPSR [9]. GPSR has two modes, the *greedy mode* and the *perimeter mode*. In the greedy mode, a sensor forwards a packet to the neighbor that is closest to the base station. When reaching a dead end, a packet enters the perimeter mode, where the routing is performed on a

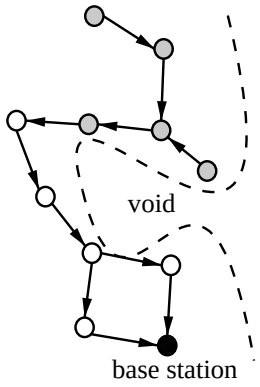

Fig. 5. Routing graph after the dead end is removed

planar graph embedded in the sensor network. The first forwarding link is identified by counter-clockwise rotating the line from the dead end to the base station. Then, the next forwarding link is identified by counter-clockwise rotating the previous forwarding link. The packet carries the location of the dead end. It resumes the greedy mode when reaching a sensor that is closer to the base station than the dead end. In Figure 4, the left-hand plot shows that a packet is routed from v to the dead end, where the greedy mode is switched to the perimeter mode. In the middle plot, the packet is routed back to v according to the "right-hand rule". In the right-hand plot, the packet is then routed from v to the base station. It was proved that the "right-hand rule" can always route a packet out of a dead end [9]. But it may create a routing path that is very inefficient. Ideally, the packet should directly take the path in the right-hand plot of Figure 4 without going from v to the dead end and then back to v. In other words, a good solution is not *to get out of a dead end* but *to not get into a dead end in the first place*.

IV. Distance Upgrading Algorithm (DUA)

A. Motivation

Our basic idea is to remove all dead ends by transforming the routing graph. This *global* objective must be achieved through a completely distributed process. Finding a solution requires a closer examination of the concept of distance. Refer to Figure 3. There is a fundamental difference between the distance from a white circle to the base station and the distance from a gray circle. The former is realizable by a path in the routing graph, but the latter is not because the void blocks the geography-based routing path. To reach the base station, a packet from the dead end has to take a detour along the boundary of the void. The actual distance traveled is much longer than the Euclid distance. Because the distance values of the sensors determine the routing paths, without changing the geographic routing algorithm itself, we can transform the routing graph to Figure 5 by artificially increasing the distance value of the dead end.

In the following, we investigate when a sensor should upgrade its distance and how much the distance should be increased. No sensor has global information. Each sensor has to make its own local decisions, while the net effect of all local decisions will remove the dead ends.

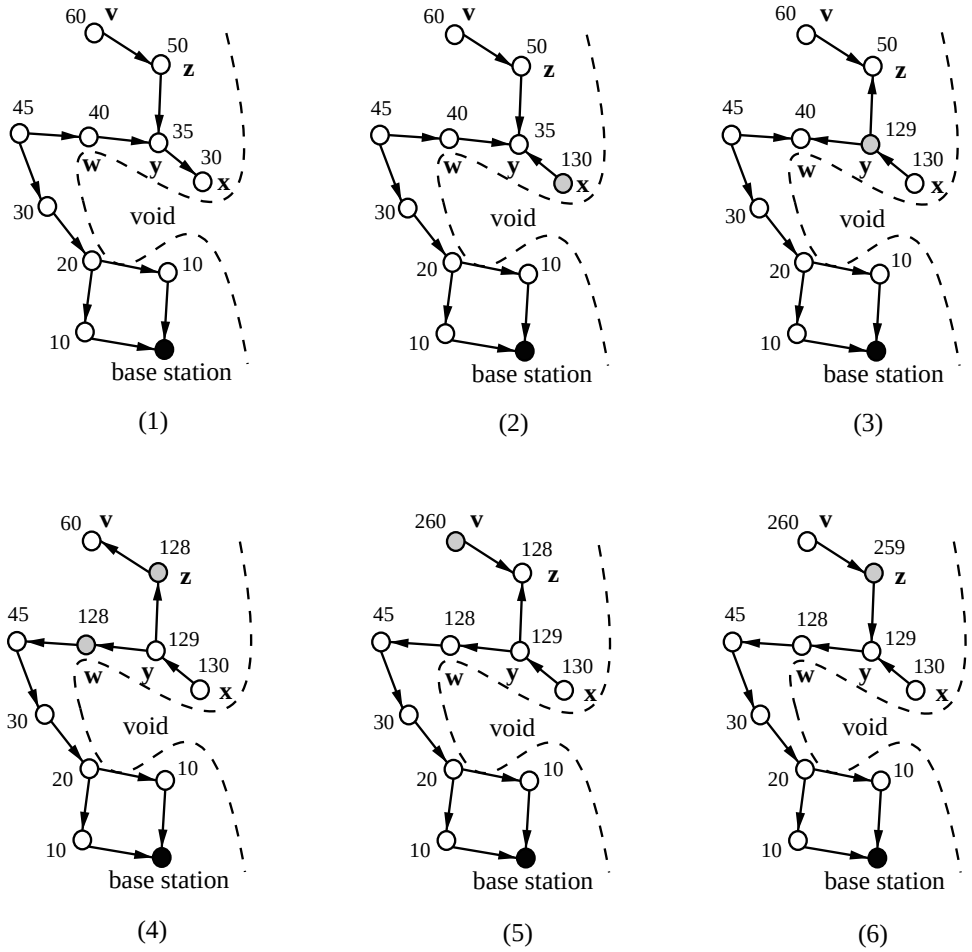

Fig. 6. Distance Upgrade

B. Distance Upgrading Algorithm

Distance Upgrade: The basic approach is simple: Whenever a sensor x finds itself becoming a dead end (i.e., $R_x = \emptyset$), it upgrades its distance such that R_x is non-empty. Let $d_e(x)$ be the Euclid distance from x to the nearest base station. We regard $d_e(x)$ as a *number* (not a variable) that can be calculated at any time based the coordinates of x and the base stations. The initial value of the distance variable $d(x)$ is the Euclid distance $d_e(x)$, but it may be modified by our distance upgrading algorithm (DUA). $d(x)$ can be regarded as a *virtual distance*, which determines the routing graph (as defined in Section III-A). Note that the *physical distance* $d_e(x)$ is never changed by DUA.

Figure 6 depicts the process of transforming a routing graph until it is free of dead ends. In each plot, the sensor that performs distance upgrade is represented by a gray circle. Let Ω be a constant that is greater than the largest Euclid distance from any sensor to its nearest base station. An easy way to choose Ω is to make it larger than the diameter of the area where the sensors are deployed. In this example, we let $\Omega = 100$.

Initially, there is only one dead end x in Plot (1). x increases its distance value by Ω, which reverses all incident routing links, as shown in Plot (2). Now y becomes a dead end. When y

performs distance upgrade, it wants to reverse links (z, y) and (w, y), but not (x, y). Hence, y chooses a distance value just under that of x, shown in Plot (3). Subsequently, w and z set their distance values just under that of y in Plot (4). Essentially link reversal is carried out in one direction, away from the original dead end x. After z's distance upgrade, v becomes a dead end. The only thing it can do is to reverse (z, v), which can be achieved by increasing its distance by 2Ω, as shown in Plot (5). Finally, in Plot (6) z sets its distance just under that of v, which reverses (y, z) only. After that, there is no dead end and the routing graph stabilizes.

The above example demonstrates three different cases of distance upgrade.

• Case 1. *A sensor x is a dead end after the initial deployment or becomes a dead end if all neighbors in R_x are not functional.* x in Figure 6 belongs to this category. The sensor increases its distance to reverse all adjacent links. Let "$\leftarrow$" be the assignment operator.

$$\begin{aligned} &\textbf{while } \exists y \in N_x, d(y) > d(x) \textbf{ do} \\ &\quad d(x) \leftarrow d(x) + \Omega \end{aligned}$$

Based on Lemma 1 that we will prove in Section IV-D, $\forall y \in N_x$, it is always true that $d(y) - d(x) < 2\Omega$. Hence, from the above pseudo code, $d(x)$ is increased by at most 2Ω.

• Case 2: *A sensor x becomes a dead end after some (but not all) neighbors have upgraded their distances.* y, z, and w in Figure 6 belong to this category. The sensor sets its distance just below the smallest value of the upgraded neighbor distances. It reverses links from the neighbors whose distances have not yet been upgraded. The pseudo code for this case is below.

$$\begin{aligned} &\alpha \leftarrow \lfloor \frac{d(x)}{\Omega} \rfloor \\ &\beta \leftarrow \min_{y \in N_x \wedge \lfloor \frac{d(y)}{\Omega} \rfloor > \alpha} \{d(y) \textbf{ mod } \Omega\} \\ &\text{choose a small positive number } \varepsilon (< \beta) \\ &d(x) \leftarrow (\alpha + 1)\Omega + \beta - \varepsilon \end{aligned} \tag{2}$$

Temporary variables, α, β, and ε are used to break up the expression into smaller sub-expressions.

• Case 3: *A sensor becomes a dead end after all neighbors have upgraded their distances.* v in Figure 6 belongs to this category. The sensor increases its distance by 2Ω, which reverses all adjacent links.[2] Namely,

$$d(x) \leftarrow d(x) + 2\Omega$$

Distance Downgrade: The above distance upgrade is an instance of the general class of Gafni-Bertsekas algorithms [15]. It may however produce inefficient routing paths as illustrated in Figure 7. Plot (1) shows the initial routing graph, where x is the only dead end. The distance upgrade based on the three cases transforms the routing graph to Plot (2), where the gray nodes are the sensors that perform distance upgrade. Most sensors follow an inefficient path from x clockwise around the void to reach the base station. The reason that causes this problem is that x' distance is raised too high. It is necessary to raise x's distance high enough in order to guarantee the link-reversal process to proceed outward until reaching a sensor that has a directed path to the base

[2]By Lemma 1 in Section IV-D, because $\forall y \in N_x$, it is always true that $d(y) - d(x) < 2\Omega$, x reverses all adjacent links after increasing $d(x)$ by 2Ω.

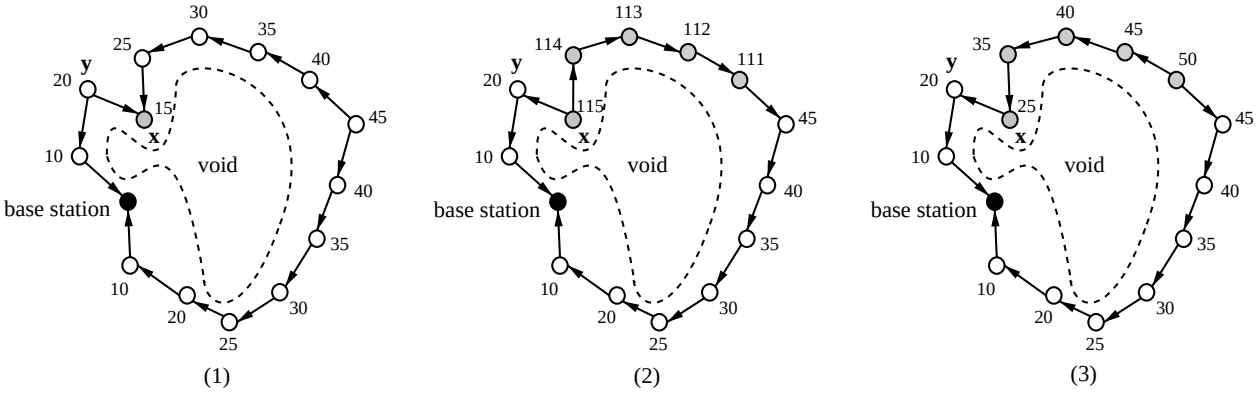

Fig. 7. Distance Downgrade

station, but there should be a mechanism to bring it down to an appropriate level after the dead ends are removed.

• Case 4: *Distance Downgrade:* For any routing link (x, y) on a directed path to a base station, if $d(x) > d(y) + |d_e(y) - d_e(x)|$, namely, $d(x)$ is larger than $d(y)$ by more than the absolute difference between their Euclid distances to a base station, then x downgrades its distance to

$$d(x) \leftarrow d(y) + |d_e(y) - d_e(x)|$$

To implement the distance downgrade, when a data packet is transmitted from x to y, if $d(x) > d(y) + |d_e(y) - d_e(x)|$, then the packet piggy-backs the locations (i.e., coordinates) of x and y. If the packet has already piggy-backed the location information of an upstream link, then it is overwritten with the locations of x and y. When the base station receives a packet with the piggy-backed information about x and y, we know that a) (x, y) is on a directed path to the base station, b) $d(x) > d(y) + |d_e(y) - d_e(x)|$, and hence x should downgrade its distance. The base station sends a control message DOWN(x, y) to x by GSRP.[3] When x receives DOWN(x, y) message, it sets $d(x)$ to be $d(y) + |d_e(y) - d_e(x)|$. Because $d(x)$ is decreased, some adjacent routing links may reverse their directions. x then finds those neighbors z with $d(z) > d(x) + |d_e(x) - d_e(z)|$, and sends a control message DOWN(z, x) to downgrade the distance of z.

After the distance downgrade is completed, the routing graph in Figure 7 is transformed from Plot (2) to Plot (3). Note that some links reverse their directions due to the distance changes.

Distance Recovery: When a void is removed, the routing graph may need to be modified in order to take advantage of shorter routing paths. Consider the example in Figure 8. Plot (1) shows the routing graph after the distance upgrade/downgrade remove the dead end x. In Plot (2), suppose a new sensor z is deployed in the void, establishing two new routing links (x, z) and (z, w). The routing graph remains correct but not optimal because there are unused, shorter paths, such as $y \rightarrow x \rightarrow z \rightarrow w \rightarrow$ base station. There needs a mechanism to "undo" the distance upgrade when additional sensors make the previous dead ends become transit nodes.

• Case 5: *Distance Recovery:* For any routing link (x, y), if sensor x finds that $d_e(x)$ is larger than $d(y)$, then it resets $d(x)$ back to $d_e(x)$.

[3]GSRP is used to deliver a one-time control message, not data packets, which are routed based on the routing graph, stored by R_x and maintained by DUA.

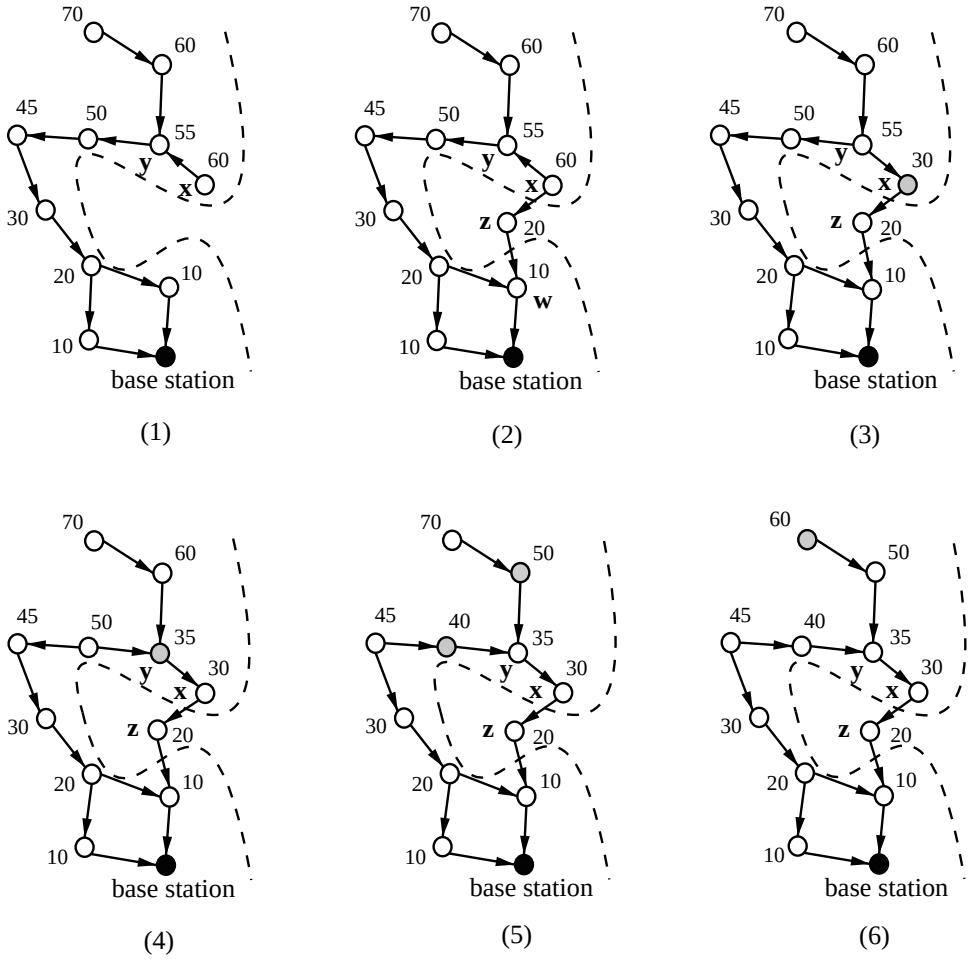

Fig. 8. Distance recovery

We continue with the example in Figure 8. After x receives the distance of the new sensor z, it recovers its original distance value (30) in Plot (3). After x notifies its neighborhood of this distance, y subsequently recovers its original distance (35) in Plot (4). More sensors recover their original distances in Plot (5) and Plot (6). After the distance recovery is completed, all sensors change their distance variables back to their original distances due to the addition of z.

Algorithm: Below we implement Cases 1, 2, 3, and 5 by three subroutines running at each sensor $x \in N$. They are Initialization(), Receive_Distance_Of_Neighbor(), and Upgrade_Distance(). Notations can be found in Table I for quick reference. Initialization() is executed after x boots up or when the set of neighbors change. The other two subroutines are invoked when x receives a notification from a neighbor about a new distance value. The distributed computation terminates when there are no more notification messages. Message loss may cause some dead ends left unremoved. This can be handled by neighbors periodically exchanging their distances and by an unremoved dead end executing Initialization() to restart the process. We implement Case 4 by Receive_DOWN_Message() that is initiated by the base station receiving a data packet that piggy-backs the location information of a routing link.

TABLE I

NOTATIONS

N	set of sensors
N_x	neighbors of a sensor x
$d(x)$	distance variable of x, initialized to be the Euclid distance to the nearest base station
$d_e(x)$	Euclid distance from x to the nearest base station
R_x	neigbors whose distances are smaller than $d(x)$
Ω	constant that is greater than the largest Euclid distance from any sensor to a base station
α, β, ε	local variables in the algorithm of DUP, whose meanings are apparent in the context of the algorithm

x.Initialization()
1. **if** $R_x = \emptyset$ **then**
2. **while** $\exists y \in N_x, d(y) > d(x)$ **do**
3. $d(x) \leftarrow d(x) + \Omega$ /* Case 1 */
4. recompute R_x
5. notify the neighborhood of the upgraded distance $d(x)$

x.Receive_Distance_Of_Neighbor(y) /* receive $d(y)$ */
6. **if** $d_e(x) > d(y)$ **then**
7. $d(x) \leftarrow d_e(x)$ /* Case 5 */
8. recompute R_x
9. notify the neighborhood with the recovered distance $d(x)$
10. **else**
11. recompute R_x
12. **if** $R_x = \emptyset$ **then**
13. Upgrade_Distance()
14. recompute R_x
15. notify the neighborhood of the upgraded distance $d(x)$

x.Upgrade_Distance()
16. **if** $\forall y, z \in N_x, \lfloor \frac{d(y)}{\Omega} \rfloor = \lfloor \frac{d(z)}{\Omega} \rfloor$ **then**
17. $d(x) \leftarrow d(x) + 2\Omega$ /* Case 3 */
18. **else**
19. $\alpha \leftarrow \lfloor \frac{d(x)}{\Omega} \rfloor$
20. $\beta \leftarrow \min_{y \in N_x \wedge \lfloor \frac{d(y)}{\Omega} \rfloor > \alpha} \{d(y) \bmod \Omega\}$
21. choose a small positive number $\varepsilon (< \beta)$
22. $d(x) \leftarrow (\alpha + 1)\Omega + \beta - \varepsilon$ /* Case 2 */

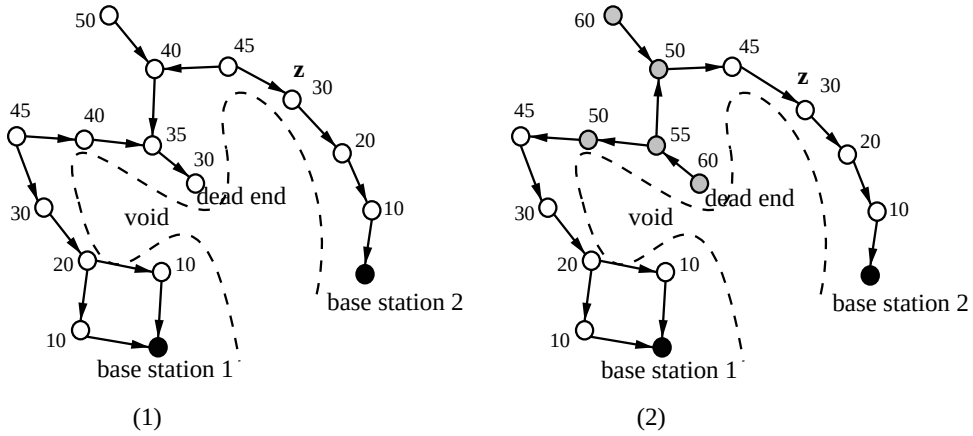

Fig. 9. DUA for multiple base stations

x.Receive_DOWN_Message(y) /* receive DOWN(x, y) */
23. **if** $d(x) > d(y) + |d_e(y) - d_e(x)|$ **then**
24. $d(x) \leftarrow d(y) + |d_e(y) - d_e(x)|$ /* Case 4 */
25. **if** $d(x)$ mod Ω = 0 **then**
26. $d(x) \leftarrow d(x)+$ a small positive number
27. **for** $z \in N_x$ **do**
28. **if** $d(z) > d(x) + |d_e(x) - d_e(z)|$ **then**
29. send DOWN(z, x) to z

In Section IV-D, we formally prove that the algorithm will remove all dead ends, which is also confirmed by our simulations. To ensure Line 21 is always doable, β must be greater than zero after Line 20. To see this is indeed the case, we only need to show that, $\forall x \in N$,

$$d(x) \text{ mod } \Omega \neq 0 \tag{3}$$

First, (3) is true initially because $0 < d(x) < \Omega$, $\forall x \in N$. Second, if (3) is true before a distance change at a sensor x, it obviously remains true after the change by Line 3, 7, 17, 22, or 24-26.

C. Examples

Figure 9 shows an example with two base stations. Initially, the distance of a sensor is the Euclid distance from the sensor to the nearest base station, as shown in the left-hand plot. For example, z is closer to Base station 2, with a distance of 30. The right-hand plot shows the routing graph after DUA. The gray nodes are the sensors that perform distance upgrade/downgrade.

Figure 10 gives a more complicated example. Sensors are deployed in a peninsula of a closed void (say, a lake) and a base station is located at the opposite side of the void. The left-hand plot shows the routing graph without DUA. At the tip of the peninsula, x is the only dead end. The packets from all sensors inside or above the peninsula will be routed to x and then routed back out if GPSR [9] is used. The right-hand plot shows the routing graph when DUA is used. The dead

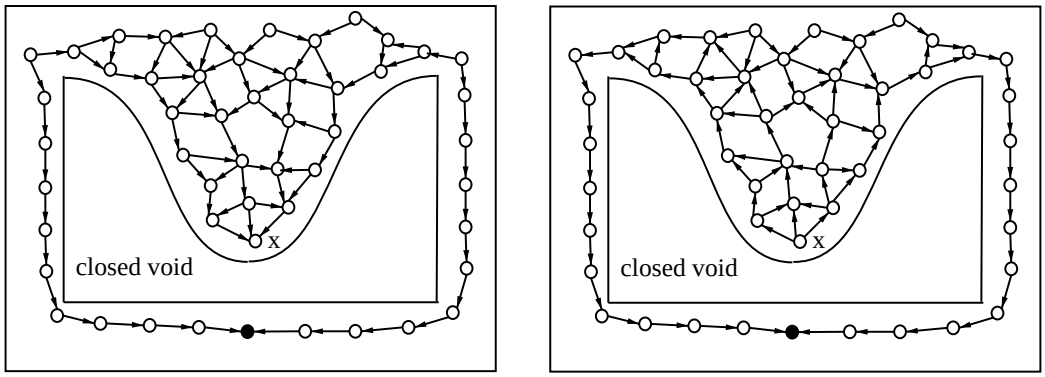

Fig. 10. Routing graph without or with DUA

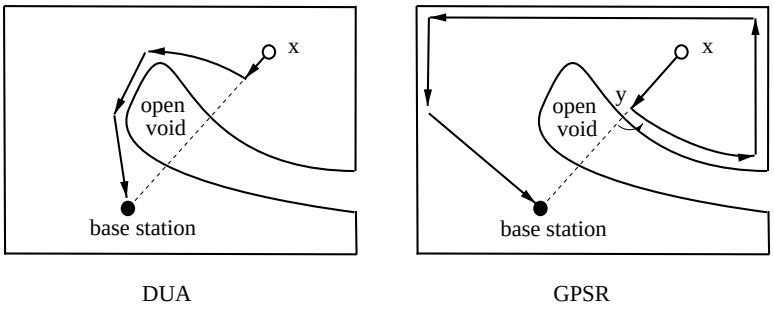

Fig. 11. Comparing DUA and GPSR with an open void

end is removed. The routing links inside the peninsula are reversed, which prevents the packets from going in the "wrong direction" towards x.

Figures 11-13 compares DUA and GPSR in how they route packets around an open void, a closed concave void, and a closed convex void, respectively. In Figure 11, suppose the sensors are densely deployed in a rectangle area except for an open void that is not completely surrounded by sensors. Consider a packet to be routed from a sensor x to a base station. The routing path generated by DUA is illustrated in the left-hand plot. The packet does not follow the dotted line to reach the edge of the void because the routing links in the area are reversed. The routing path by GPSR is illustrated in the right-hand plot. When a packet reaches the edge of an open void, the "right-hand rule" will guide the packet along the external boundary of the entire deployment area until the packet reaches a node that is closer to the base station than y. Therefore, GPSR can be extremely inefficient in the case of an open void.

Figure 12 examines a closed concave void. The routing paths by DUA are shown in the left-hand plots. Note that the routing links inside the peninsula are reversed by DUA, as shown in Figure 10. The routing paths by GPSR are shown in the right-hand plots. GPSR routes packets in far-longer paths than DUA. Figure 13 examines a closed convex void with a dead end at the top of the void. Again, DUA produces shorter routing paths than GPSR.

D. Correctness Proof

In the following, we prove that DUA will remove all dead ends in finite time.

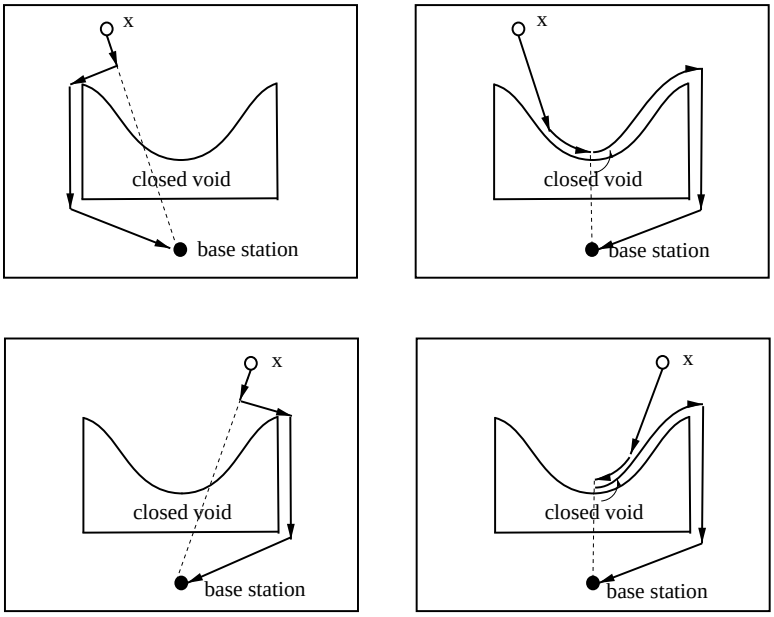

Fig. 12. Comparing DUA and GPSR with a closed concave void

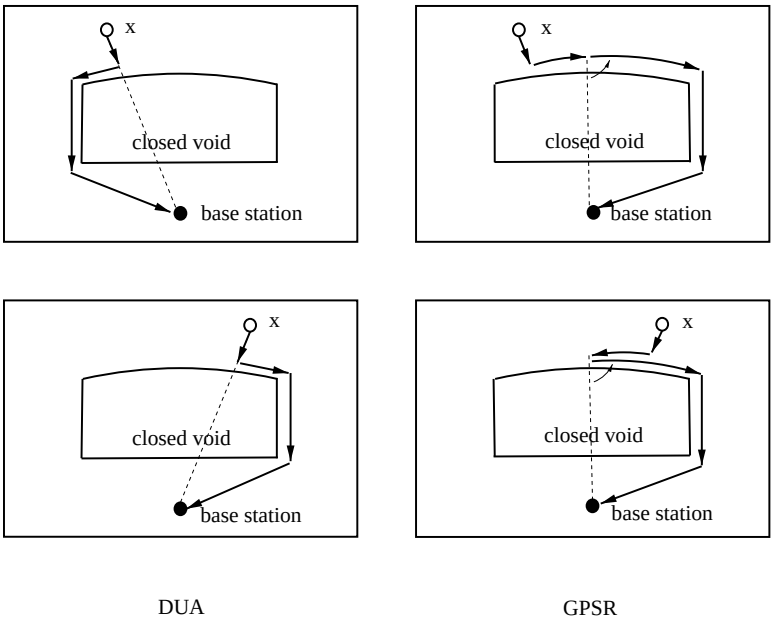

Fig. 13. Comparing DUA and GPSR with a closed convex void

Theorem 1: The routing graph of a sensor network is always loop-free.

Proof: By the definition of routing link (Section III-A), $d(x) > d(y)$ for **any** link (x, y) in the routing graph. Suppose there is a loop in the routing graph, $x \rightarrow y \rightarrow z \rightarrow ...w \rightarrow x$, which consists of routing links (x, y), (y, z), ..., (w, x). We then have, $d(x) > d(y)$, $d(y) > d(z)$, ..., $d(w) > d(x)$, which leads to $d(x) > d(x)$, a contradiction. □

Lemma 1: For any routing link (x, y), $d(x) - d(y) < 2\Omega$.

Proof: The lemma holds initially because $d(x) < \Omega, \forall x \in N$. Assume that the lemma holds before a distance change. We prove that it remains valid after the change. It is obviously true for distance downgrade (Line 24) and distance recovery (Line 7). Now consider the initialization, Lines 2-3. Before those lines, because $R_x = \emptyset$, $\forall y \in N_x, d(y) \geq d(x)$, and by our assumption, $\forall y \in N_x, d(y) - d(x) < 2\Omega$. After Lines 2-3, $d(x)$ is greater than $d(y), \forall y \in N_x$, with a distance increment of at most 2Ω, and therefore, $d(x) - d(y) < 2\Omega$. The only cases left are distance upgrades.

Without losing generality, consider a distance upgrade at a sensor x. Let $d_o(x)$ and $d_n(x)$ be the value of $d(x)$ before and after the upgrade respectively, where the subscribe "o" means "old" and "n" means "new". The upgrade only affects x's adjacent links. Before the update, all adjacent links point at x. After the update, there are two cases.

• Case one: *The upgrade of $d(x)$ reverses a routing link (y, x) to (x, y).* Prior to the upgrade, because the routing link is from y to x, we have

$$d(y) > d_o(x) \tag{4}$$

When x.Upgrade_Distance() is executed, if the branch of Line 17 is selected, $d_n(x) = d_o(x) + 2\Omega$. By (4), we must have

$$d_n(x) - d(y) < 2\Omega$$

Now suppose the branch of Lines 19-22 is executed. By Line 19, $d_o(x) \geq \alpha\Omega$. Hence, $d(y) > \alpha\Omega$ because of (4). By Line 20, $\beta < \Omega$. After Line 22,

$$d_n(x) = (\alpha + 1)\Omega + \beta - \varepsilon < d(y) + \beta - \varepsilon + \Omega < d(y) + 2\Omega$$

Hence, $d_n(x) - d(y) < 2\Omega$ after the execution of x.Upgrade_Distance().

• Case two: *The upgrade of $d(x)$ does not reverse the direction of a routing link (y, x).* Before the upgrade, $0 < d(y) - d_o(x) < 2\Omega$ by our assumption. After the upgrade, $d_n(x)$ is larger than $d_o(x)$ and hence we still have $0 < d(y) - d_n(x) < 2\Omega$.

The lemma holds. □

Lemma 2: $\forall x, y \in N,\ d(x) - d(y) < 2k\Omega$, where k is the number of hops on the shortest path from x to y.

Proof: A direct consequence of Lemma 1. □

Lemma 3: Two consecutive upgrades increase the distance of a sensor by at least Ω.

Proof: Each distance upgrade at x increases $\lfloor \frac{d(x)}{\Omega} \rfloor$ at least by one (Line 3, Line 17, or Line 22). There is no distance downgrade between the two upgrades because that would mean x was

on a directed path to a base station, which in turn would mean the second upgrade won't happen. Hence, two upgrades increases $\lfloor \frac{d(x)}{\Omega} \rfloor$ at least by two, and therefore increases $d(x)$ at least by Ω. □

Lemma 4: If the routing graph of a sensor network is connected, it will remain connected after some sensors change their distances.

Proof: The distance change by sensors will only change the direction of routing links but never remove any link. Hence, the routing graph remains connected. □

Theorem 2: Consider a connected sensor network. a) DUA will terminate. b) After it terminates, $R_x \neq \emptyset,\ \forall x \in N$, and any path in the routing graph leads to a base station.

Proof: Let the routing graph be $\langle N, E \rangle$, where N is the set of sensors and E is the set of routing links. Let M be the set of sensors that have at least one routing path to reach a base station. Once a sensor is in M, it will no longer perform any distance upgrade because it has at least one outgoing link. Before a sensor is in M, it will not perform any distance downgrade because its packets cannot reach a base station. Consider a sensor x that satisfies the following conditions, $x \notin M$ and $\exists z \in M, (z, x) \in E$. Because the routing graph is always connected (Lemma 4), there must exist such a sensor as long as $M \neq N$. Below we prove that link (z, x) will be reversed by DUA.

Let X be the set of sensors reachable by x and G_x be the subgraph consisting of nodes in $X + \{x\}$. X must not include any base station; otherwise, x would be in M. $\forall y \in X$, because there is a routing path from x to y, we must have $d(y) < d(x)$. The DUA algorithm continuously upgrades the distances of the dead ends (i.e., sinks in the graph other than the base stations) until there is no dead end. By Theorem 1, G_x is acyclic. An acyclic graph must have a sink. Hence, there are always some nodes in X upgrading their distances until X becomes empty. Furthermore, it takes finite time for this process to complete because of the following reasons. First, by Lemma 2, $\forall y \in X$, $d(x) - d(y)$ is finite and bounded by 2Ω multiplied by the length of the shortest path from x to y in G_x. Second, by Lemma 3, any two distance upgrades by y will increase $d(y)$ by at least Ω. Hence, it takes a finite number of upgrades for $d(y)$ to be greater than $d(x)$. Once $d(y)$ is greater than $d(x)$, y cannot be in X. As the sensors in X continuously upgrade their distances and exclude themselves from X once their distances become too big, X will eventually be reduced to an empty set, which makes x a dead end. When x upgrades its distance, it reverses link (z, x) and makes itself a member of M.

Because any sensor x that satisfies the condition, $x \notin M \wedge \exists z \in M, (z, x) \in E$, will become a member of M and because there exists such a sensor as long as $M \neq N$, it follows that M will eventually become N. Because no sensor in M will upgrade its distance and DUA is driven by distance upgrade, the algorithm will terminate when $M = N$.

When the algorithm terminates, there are no sinks in the routing graph except for the base stations. Hence, $R_x \neq \emptyset,\ \forall x \in N$. Because the routing graph is acyclic, any path leads to one of the only exit points — the base stations. □

Theorem 1 shows that there exists no routing loop. Theorem 2 shows that each sensor will eventually establish at least one routing path to a base station.

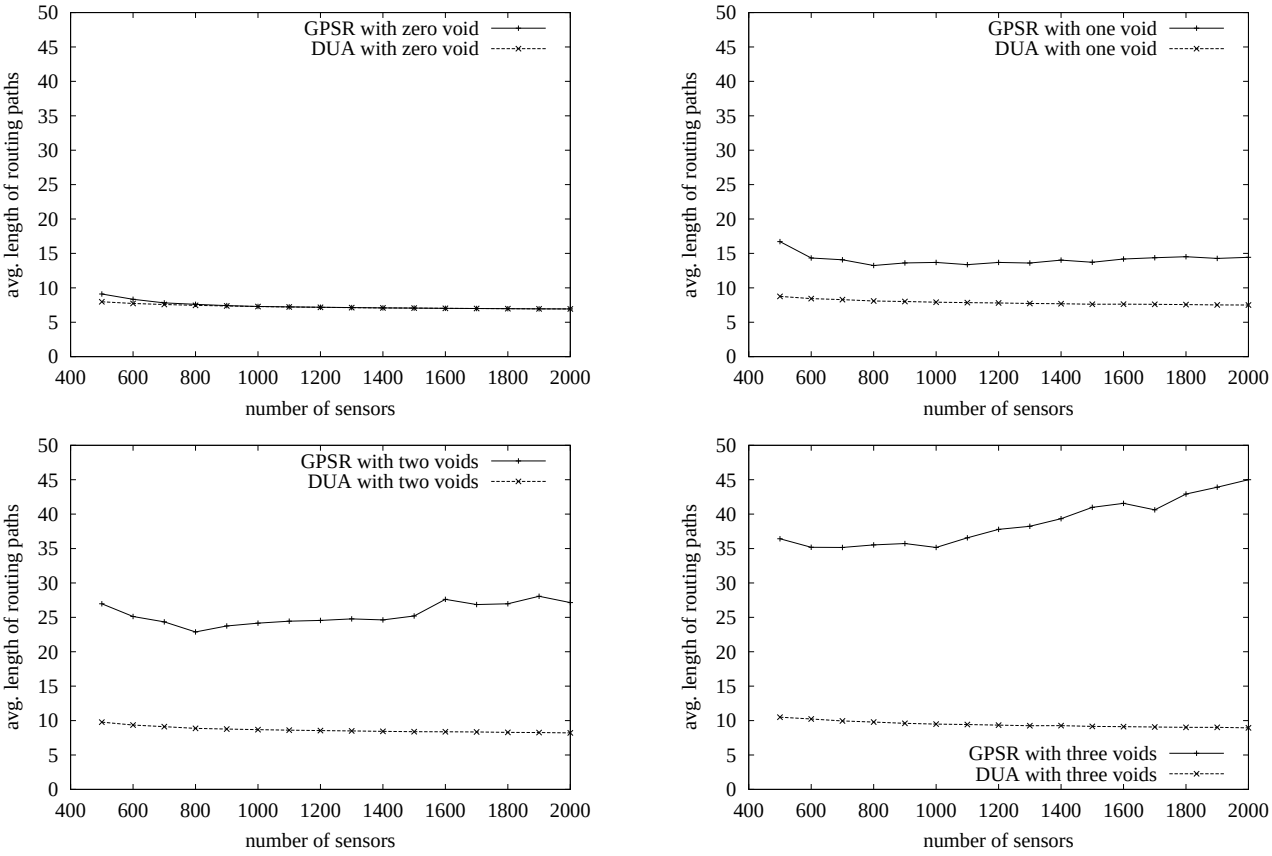

Fig. 14. average length of routing paths

V. SIMULATION

A. Simulation Setup

We use simulations to evaluate the performance and the overhead of the proposed DUA in aggregating data from sensors to a base station. We compare DUA with GPSR [9]. The simulation setup is described as follows. Sensors are randomly placed in a flat area of 1000 by 1000. The transmission range of a sensor is 100. The coordinates of the base station is $(500, 1000)$, i.e., the middle point of the upper boundary line. zero to three voids are randomly placed in the area. The void has a T shape with a default area size of 70,000. Our choice for the shape of the void does not carry particular significance. DUA is equivalent to GPSR when this is no void. The difference of the two algorithms manifests only when there are dead ends due to voids. A T shape void conveniently creates one or more dead ends, depending on its location relative to the base station.

B. Comparing Average Length of Routing Paths

When there are voids, the average length of routing paths in DUA can be considerably smaller than that in GPSR, as demonstrated by Figure 14. With zero to three voids respectively, the four plots show the average path length with respect to the number of sensors deployed. When there is zero T void, DUA and GPSR have similar path lengths. DUA is slightly better due to *naturally formed* small voids between sensors, which can happen when the number of sensors (and thus the

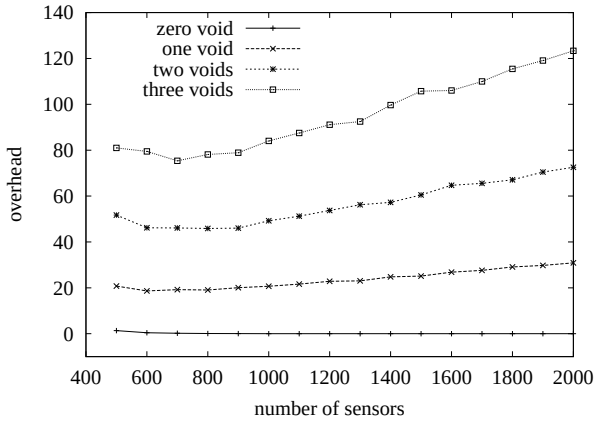

Fig. 15. overhead of DUA for deadend removal

node density) is relatively small. When the number of T voids increases, DUA outperforms GPSR significantly. With one void, the routing paths in DUA are 39% - 50% shorter than those in GPSR. With two voids, the routing paths in DUA are 60%-71% shorter than those in GPSR. With three voids, the routing paths in DUA are 71%-80% shorter than those in GPSR. It should be noted that, although only a very small number of sensors actually become dead ends, their impact can be significant in GPSR because data from many upstream sensors will be routed to them and then have to be routed out via long paths based on the right-hand rule.

All four plots indicate that the average path length decreases slightly with respect to the node density (proportional to the number of sensors deployed in the area). That is because DUA is a greedy routing algorithm that picks the sensor with the smallest distance in R_x as the next hop. When there are more nodes in the neighborhood, a sensor is more likely to find its next hop closer to the base station. The same thing is true for GPSR when there is no void. However, when there is one or more voids, on one hand, the average path length decreases with respect to the node density in the greedy mode; on the other hand, it increases with respect to the node density in the non-greedy perimeter mode. The net result is that, as the node density increases, the routing paths first become shorter and, after the node density reaches certain level, they start to grow longer.

C. Overhead of DUA

DUA requires additional overhead to remove dead ends. Because the radio transmissions dominate the sensors' energy spend, we measure the overhead of DUA by the total number of control-message transmissions befor all dead ends are removed. Here a transmission is a distance notification message sent by a sensor to its neighborhood immediately after the sensor performs a distance upgrade.

Figure 15 shows that DUA has modest overhead. First of all, when there is no void, DUA incurs no overhead. Secondly, the more the number of voids, the more the overhead for dead-end removal. For three voids, the number of transmissions is 123 for 2000 sensors, namely, 0.06 transmission per sensor, which is very small. The reason is that dead-end removal only involves localized operations, which has limited one-time overhead in a small region.

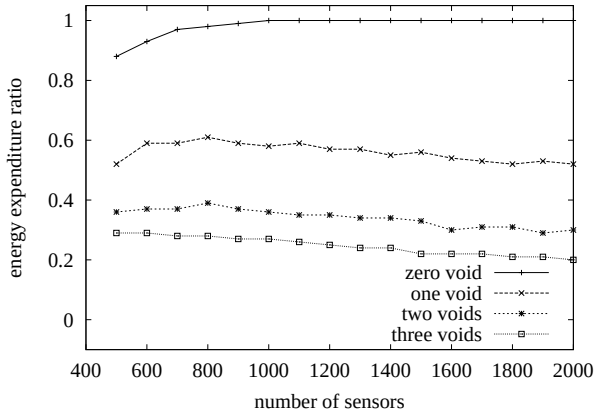

Fig. 16. energy expenditure ratio of DUA and GPSR for delivering a packet

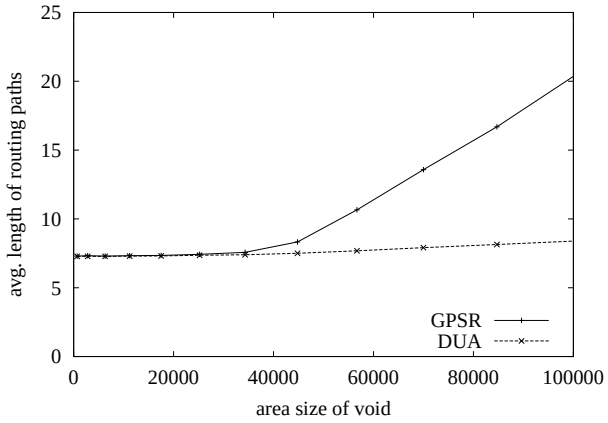

Fig. 17. impact of void size

D. Comparing Energy Expenditure of Data Aggregation

Because DUA and GPSR have different average lengths of routing paths, their energy expenditure in data aggregation (characterized by the number of transmissions for delivering one packet to the base station) will also differ. Figure 16 shows the ratio of energy expenditure by DUA and that by GPSR. When there is zero T void, the ratio is close to one because DUA and GPSR are virtually identical in this case. The more the number of voids, the smaller the ratio becomes, which means more energy saving by DUA. Moreover, this saving is long term, as long as data are sent from sensors to the base station.

E. Impact of Void Size

Figure 17 shows the impact of void size on the average length of routing paths. There is a single T void. Increasing the size of a void may not increase the number of dead ends, but will increase the number of upstream sensors whose packets fall into the dead ends. Therefore, the path length increases with void size in GPSR. It also increases in DUA but much more lowly. The performance gap between DUA and GPSR increases when there are more voids.

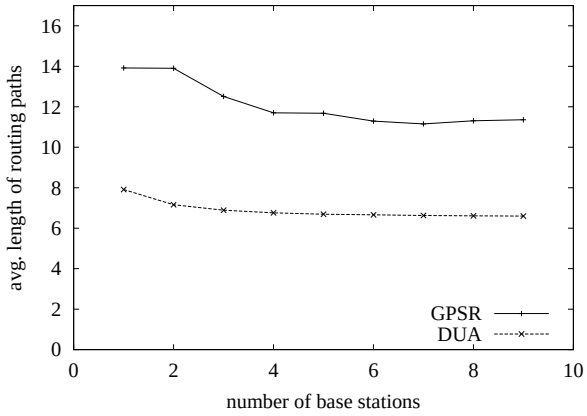

Fig. 18. multiple base stations

F. Multiple Base Stations

We also simulated the case of multiple base stations. One through ten base stations are placed with even spacing along the upper boundary of the deployment area. We ran DUA and GPSR, respectively. Figure 18 compares their average lengths of routing paths from sensors to the nearest base stations. For both DUA and GPSR, the routing paths are shorter when there are more base stations. DUA consistently outperforms GPSR by 40%-50%. A single T void was used in the simulation of Figure 18. With more voids, the performance gap between DUA and GPSR will increase.

VI. Conclusion

This paper proposed a new geographic routing algorithm that removes the interference of voids on the routing paths from sensors to the base stations. It produces more efficient routing paths than the traditional right-hand rule. No global periodic broadcasts are required to maintain the routing paths. The algorithm responds to topology changes instantly with localized operations.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communication Magazine*, August 2002.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Drive for Wireless Communications Technology," *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[3] L. Schwiebert, S. Gupta, and J. Weinmann, "Research Challenges in Wireless Networks of Biomedical Sensors," *Mobile Computing and Networking*, pp. 151–165, 2001.

[4] E. Biagioni and K. Bridges, "The Applications of Remote Sensor Technology to Assist the Recovery of Rare and Endangered Species," *International Journal of High Performance Computing Applications, Special Issue on Distributed Sensor Networks*, April 2003.

[5] H. T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," *in Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'03)*, March 2003.

[6] R. Brooks, P. Ramanathan, and A. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, 2003.

[7] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," *in Proceedings of IEEE INfocom'02*, June 2002.

[8] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," *3rd Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM'99)*, August 1999.

[9] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *in Proceedings of ACM MobiCom'00*, August 2000.

[10] T. He, J. A.Stankovic, C. Lu, and T. F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," *in Proceedings of International Conference on Distributed Computing Systems (ICDCS'03)*, May 2003.

[11] Q. Fang, J. Gao, and L. J. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," *in Proceedings of IEEE INFOCOM'04*, March 2004.

[12] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information," *in Proceedings of 9th Annual International Conference on Mobile Computing and Networking (Mobicom'03)*, September 2003.

[13] J. Newsome and D. Song, "GEM: Graph Embedding for Routing and Data-Centric Storage in Sensor Networks without Geographic Information," *in Proceedings of First International Conference on Embedded Networked Sensor Systems (SenSys'03)*, November 2003.

[14] R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica, "Beacon Vector Routing: Scalable Point-to-Point in Wireless Sensornets," *Technical Report IRB-TR-04-12, Intel Research Berkeley*, May 2004.

[15] E. M. Gafni and D. P. Bertsekas, "Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology," *IEEE Transactions on Communications*, January 1981.