Neighbor Oblivious Link Reversal over Duty-Cycled WSNs

Santosh Ramachandran, S.V.R.Anand, Malati Hegde, Anurag Kumar, Rajesh Sundaresan
ECE, Indian Institute of Science, Bangalore, India
Email: santosh, anand, malati, anurag, rajeshs@ece.iisc.ernet.in

***Abstract*—We propose partial and full link reversal algorithms to bypass voids during geographic routing over duty-cycled wireless sensor networks. We propose a distributed approach that is oblivious to one-hop neighbor information. Upon termination of the algorithm, the resulting network is guaranteed to be destination-oriented. Further, to reduce the delays incurred under reactive link reversal, we propose the use of 'pseudo-events', a preemptive link reversal strategy, that renders the network destination-oriented before the onset of a real event. A simulation study of the effectiveness of pseudo-events is also provided.**

***Index Terms*—partial link reversal, full link reversal, pseudo-events, sensor network, geographic routing, distributed algorithm**

I. Introduction

We consider a wireless sensor network (WSN) with a designated sink node. The purpose of the WSN is to generate an alarm when certain events are detected. An alarm packet originating at a node has to be routed to the sink node. For such problems, *geographic routing* [1] is a popular protocol for packet delivery. It is scalable, stateless and reactive without the need for prior route discovery. The protocol employs geographic greedy routing, where a node forwards a packet to another node within the communication range and closer to the destination. Such a protocol requires a node with a packet to be aware of its geographical location, and those of its neighbors. This next hop node in the direction of the destination is defined as a *greedy node*. Clearly, the resulting routing graph is a directed acyclic graph (DAG). A DAG is said to be *destination-oriented* when there is a directed path in the DAG from any node to the sink. A DAG is *destination-disoriented* if there exists a node other than the sink that has no outgoing links. Such a node is said to be *stuck*. A destination-oriented network under geographic greedy routing may be rendered destination-disoriented in the presence of node failures. The failure of geographic greedy routing in the presence of stuck nodes is commonly referred to as *the local minimum condition* [2]. Solutions such as face routing [1], convex hull routing [3] and link reversal routing [4] were proposed in the literature to pull the network out of a local minimum condition. All these algorithms require knowledge of one-hop neighbors. Maintenance of one-hop neighbor information, in general, requires periodic transmissions of *Keep Alive* packets. Nodes however are duty-cycled to enhance the lifetime of the network. The intricacies introduced by duty-cycled sensors in the maintenance of one-hop neighbor information involves several message exchanges between nodes and its one-hop neighbors, associated access issues and collision resolution mechanisms. This can be both time and energy consuming.

This work was supported by a research grant on Wireless Sensor Networks from DRDO.

This work is motivated by the question: is there a distributed, *neighbor oblivious* protocol (i.e., a protocol with zero overhead for neighbor discovery) that can pull a network out of its local minimum condition and render it destination-oriented?

Gafni and Bertsekas in [4] propose two general classes of *link reversal algorithms* for converting a destination-disoriented DAG into a destination-oriented DAG (see section II). Henceforth we refer to their algorithms as *GB algorithms*. Both classes of algorithms require one-hop neighbor information. In this paper we propose neighbor oblivious link reversal algorithms. We then embed them into the GB framework to show that our proposed algorithms render the network destination-oriented, but in a neighbor oblivious fashion.

Executing link reversal reactively (i.e., when an actual alarm meets a stuck node), to deliver an alarm packet to the sink, can lead to significant end-to-end delay [5]. We propose the use of *pseudo-events* to maintain the network in a destination-oriented state before the onset of real events. Pseudo-events create virtual events distributed across space and time. This initiates link reversal at stuck nodes while relaying these pseudo-alarm packets to the sink. Our proposed maintenance technique is likely to be more energy efficient and works well even in a duty-cycled network since the forwarding protocol itself repairs the network (see section VI).

Our main contributions are the following :

- We provide neighbor oblivious partial and full link reversal schemes (see section III) that fit within the framework of the GB algorithms (see section V).
- We propose the use of pseudo-events, generated across time and space, that maintains the network in a destination-oriented state.

The analyses of the proposed algorithms is presented in Section IV. Section VII contains some concluding remarks.

Related Literature

The distributed planar graph traversal technique, commonly known as face routing and presented in [1] and [6], guarantees delivery if a path exists. The technique requires apriori knowledge of the full neighborhood. Kalosha et al. [7] addressed a beaconless recovery problem where the local planar subgraph

is constructed on the fly. They did not consider the duty-cycling of sensors. The RGP protocol [8] results in a shortest path routing protocol to bypass voids. The protocol requires communication among neighbors. Yu et al. [9] discussed a void bypassing scheme when both source and destination nodes are mobile. Leong et al. [3] presented a new geographic routing protocol called GDSTR. GDSTR uses convex hulls which requires maintaining topology information, an onerous task in a duty-cycled environment. The ALBA-R protocol [10] is a nonplanar routing across voids. Routing is based on hierarchy of colors and is designed to work with ALBA [11], another greedy forwarding protocol for WSN. Chen et al. [12] proposed partial link reversal under the assumption that neighbor information is available. As opposed to the existing literature, we propose a neighbor oblivious technique to bypass voids.

II. Overview of GB Algorithms

Gafni and Bertsekas [4] propose two classes of distributed renumbering algorithms, *full link reversal* and *partial link reversal* that transform a destination-disoriented DAG into destination-oriented DAG.

The renumbering scheme can be used in geographic greedy forwarding by assigning numbers to nodes ($n_1,\ n_2, \ldots$) which are totally ordered by the relation $<$ such that for any two nodes with node Id's 1 and 2 there holds $n_1 < n_2$ or $n_2 < n_1$ but not both. These numbers, corresponding to either hop counts or distances to sink, are used in assigning directions to routing links.

In GB algorithms, each node is associated with a set of integers, (α_i, i) for full reversal and (α_i, β_i, i) for partial reversal where α_i and β_i are integers and i is the node index. The tuples are ordered lexicographically. The direction of the links between nodes are always oriented from a node with a higher tuple to a node with a lower tuple i.e., a link is oriented from i to j if $(\alpha_i^k, i) > (\alpha_j^k, j)$ for full reversal and $(\alpha_i^k, \beta_i^k, i) > (\alpha_j^k, \beta_j^k, j)$ for partial reversal, where k is the iteration index. Hence a node i is *stuck* if for every neighbor j of i we have $(\alpha_i^k, i) < (\alpha_j^k, j)$ (similarly for partial reversal). The GB algorithms gives new values for the integers α_i and β_i to stuck nodes iteratively and distributively so that a destination-oriented DAG is obtained.

Full link reversal: In this algorithm, a stuck node reverses the direction of all the incoming links. Let C_i denote the neighbors of i. A stuck node i at iteration k increases α_i^k to

$$\alpha_i^{k+1} = \max\{\alpha_j^k \mid j \in C_i\} + 1$$

while all other nodes preserve their values. The algorithm terminates when a destination-oriented DAG is obtained.

Partial link reversal: In this algorithm, every node i keeps a list of its neighboring nodes j that have reversed theirs links to i. If node i is stuck then it reverses the directions of all neighbor links to nodes j not in the list. If such a j does not exist, i.e., the list contains all the neighbors of i, then i reverses all its links and empties the list. The algorithms terminates when a destination-oriented DAG is obtained. Here a stuck node i at iteration k sets α_i^k and β_i^k to

$$\alpha_i^{k+1} = \min\{\alpha_j^k \mid j \in C_i\} + 1$$

$$\beta_i^{k+1} = \begin{cases} \min\{\beta_j^k \mid j \in C_i \text{ with } \alpha_i^{k+1} = \alpha_j^k\} - 1 \\ \qquad \text{if there exists a j with } \alpha_i^{k+1} = \alpha_i^k \\ \beta_i, \qquad \text{otherwise} \end{cases}$$

All other nodes j maintain the same integers for α_j and β_i i.e., $\alpha_j^{k+1} = \alpha_j^k,\ \beta_j^{k+1} = \beta_j^k$. Gafni and Bertsekas [4] show the following

1) *The algorithm terminates in a finite number of iterations regardless of the timing and order of reversals and results in the same destination-oriented DAG.*
2) *Only those nodes that do not have a greedy path to the sink undergo link reversal in obtaining a destination-oriented DAG.*

III. Neighbor Oblivious Link Reversal

A. Overview of the algorithm

The updates at a stuck node, in both the full and partial link reversal GB algorithms, depend on knowledge neighbor states (the numbers at each of the neighbors). If the stuck node does not know these, it has to establish a link with each of its neighbors and gather these values in a reliable fashion. We now see how to avoid this exchange (neighbor obliviousness).

In the sequel, the parameter h for height (or potential) will play the role of the α in GB algorithms. Suppose that the update algorithm is such that a node, at any stage and given its own current state (height), knows the entire *range* of all neighbors' heights. Then it may execute a full reversal by raising its height to a value one more than the maximum in the range. It may also execute a partial reversal by doing this raising in two steps. The first step enables a reversal of links not already reversed. The second step enables a reversal of all links, if needed. All these must be possible without requiring the exact knowledge of neighbors' heights.

In both algorithms the number of full reversals at any stage is indicated by the integer t_i for node i. The state of partial reversal is indicated by α_i for node i.

The heights are initialized to either hop counts or to distances from destination (evaluated either from actual or virtual locations [13]), with the destination at zero height.

Notations

- $[N] = \{1, 2, \ldots, N\}$, set of node indices
- $i \in [N]$, node index
- $t = \{t_1, t_2, \ldots, t_N\} \in \mathbb{Z}_+^N$, count of number of full reversals at a node
- $z = \{z(1), z(2), \ldots\}$, bounds on heights of all nodes after the indicated number of full reversals.
- C_i, list of one-hop neighbors of i

Full reversal

- $h = (h_1(t_1), h_2(t_2), \ldots, h_N(t_N))$ set of heights; $h \in \mathbb{R}_+^N$
- $h_{\max} = \max\{h_i(0) : i \in [N]\}$
- $GF_i(h) = \{j \in [N] : \ h_j(t_j) < h_i(t_i)\}$, greedy forwarding set for node i, given h

Partial reversal

- $h = (h_1(t_1, \alpha_1), h_2(t_2, \alpha_2), \ldots, h_N(t_N, \alpha_N))$ set of heights; $h \in \mathbb{R}_+^N$
- $h_{\max} = \max\{h_i(0,0) : i \in [N]\}$
- $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\} \in \{0,1\}^N$, indication of whether a partial reversal was done after the last full reversal.
- $\lambda = \{\lambda(1), \lambda(2), \ldots\}$, bounds on the heights of nodes after the indicated number of full reversals.
- $GF_i(h) = \{j \in [N] : \ h_j(t_j, \alpha_j) < h_i(t_i, \alpha_i)\}$, greedy forwarding set for node i, given h

B. Full reversal algorithm

The state of each node, i, has two components: t_i and $h_i(t_i)$ which are initialized respectively to 0 and to the height of the node during node deployment. The sequence $z(t_i)$ satisfies the following recursion

$$z(0) = h_{\max} + 1, \quad (1)$$
$$z(t_i + 1) = 2z(t_i) + 1. \quad (2)$$

Algorithm 1. Full Link Reversal
For any node i,
if $(GF_i(h) = \emptyset)\{$

- $h_i(t_i + 1) = z(t_i) + h_i(t_i)$
- $t_i = t_i + 1$

$\}$

From the algorithm, it is clear that only stuck nodes upgrade their heights. The choice of $z(\cdot)$ and the initial heights are such that every time t_i and h_i are updated, node i would fully reverse all its links, without knowing neighbors heights. This eliminates the need for communicating with neighbors to compute new height such that it is greater than all its neighbors heights.

C. Partial reversal algorithm

The state at each node has three components: the number of full reversals t_i, indication of whether the node is currently in partial reversal or not α_i and current height $h_i(t_i, \alpha_i)$. The α_i and t_i are initialized to 0, while $h_i(t_i, \alpha_i)$ is initialized to the initial height of the node as determined during node deployment. The λ and z sequences satisfy the following recursion

$$z(0) = h_{\max}; \lambda(0) = 2z(0) + 1 \quad (3)$$
$$z(t_i + 1) = \lambda(t_i) + z(t_i) \quad (4)$$
$$\lambda(t_i + 1) = 2z(t_i + 1) + 1. \quad (5)$$

Algorithm 2. Partial Link Reversal
For any node i,
while $(GF_i(h) = \emptyset)\{$

- $u = (1 + \alpha_i)\lambda(t_i) - h_i(t_i, \alpha_i)$
- $t_i = t_i + \alpha_i$
- $\alpha_i = 1 - \alpha_i$
- $h_i(t_i, \alpha_i) = u$

$\}$

Again, only stuck nodes upgrade their states. The partial link reversal state is highlighted in the figure 1. When entering $\alpha_i = 1$ from state 0, the heights are chosen such that links already reversed since the last full reversal are left as they were. When entering state 0 from state 1, all links are reversed. So the t, which counts the number of full reversals, in incremented.

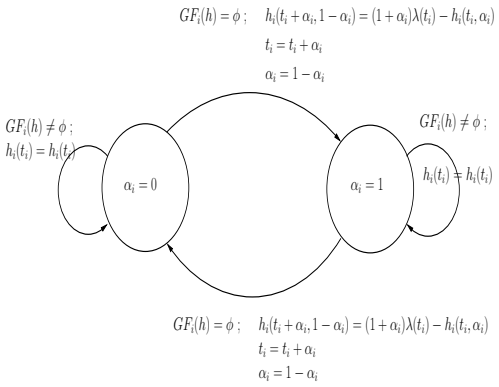


Fig. 1. Projection of partial link reversal on state α_i at any node i

IV. Analyses of Neighbor Oblivious Link Reversal

An execution is defined as a sequence of link reversals. A full execution is defined as the number of link reversals required to transform a destination-disoriented DAG to destination-oriented DAG. Given the same initial destination-disoriented DAG, any deterministic link reversal algorithms exhibit equivalence of execution. A formal proof on this fact is given by Costas et al. in [5]. Both the full and partial link reversal algorithm exhibits equivalence of execution.

A. Properties for Algorithm 1

The following property is an immediate consequence of the update because the z sequence is strictly positive.

Lemma IV.1. *Each height upgrade by a stuck node i strictly increases h_i.*

The proof of Costas et al. [5] on equivalence of execution implies the following result

Proposition IV.1. *Given the same initial destination-disoriented DAG, if a full execution of the GB algorithm requires k iterations, then Algorithm 1 also takes exactly k iterations for a full execution.*

B. Delay analysis for Algorithm 1

By proposition IV.1 the number of iterations incurred by Algorithm 1 and full reversal GB algorithms are same. A node determines it is stuck if it has not received any probe ACK in one duty-cycle (D) time. The link reversal algorithm is initiated only when a node knows it is stuck, for which an additional delay of D is incurred to gather all neighbors heights. Hence the total end to end delay (d_e) for k iterations of GB algorithms is given by

$$d_e = 2kD + \text{forwarding delay} \quad (6)$$

Algorithm 1 due to its oblivious nature can avoid the extra step to gather neighbor information in each iteration. The end-to-end delay (d_e) for k iterations of our algorithm is given by

$$d_e = kD + \text{forwarding delay} \tag{7}$$

Furthermore, the reduced communications result in a reduced energy expenditure.

C. Properties for Algorithm 2

Similar properties hold for Algorithm 2 as well. Proofs are omitted.

Lemma IV.2. *Each height upgrade by a stuck node i strictly increases h_i.*

Proposition IV.2. *Given the same initial destination-disoriented DAG, if the full execution of the GB algorithm requires k iterations, Algorithm 2 takes at most $2k$ iterations.*

D. Delay Analysis for Algorithm 2

By proposition IV.2 and analysis similar to full reversal the end to end delay of the GB algorithms for k iterations is

$$d_e = 2kD + \text{forwarding delay} \tag{8}$$

The number of iterations incurred by Algorithm 2 is at most $2k$. Hence the worst case end to end delay is

$$d_e = 2kD + \text{forwarding delay} \tag{9}$$

But the average case end to end delay is $kD +$ forwarding delay $< d_e < 2kD +$ forwarding delay. Despite the increase in the number of iterations, the average case end to end delay is still lower than the GB algorithms.

E. Implementation Issues

In practice, height values will be stored using finite bit-width representations. The heights during link reversal however have exponential growth in the number of link reversals. One approach to address the imminent issue of overflow is to keep track of number of overflows and hence have a dynamic bit width for heights. Another approach is for the node with an overflowed height value to reinitiate network self-organization so as to renormalize height values.

V. Embedding into the general class of GB algorithms

From the GB paper, let V is the set of N-tuples, $v = (a_1, a_2, \ldots, a_N)$, where $(a_1, a_2, \ldots, a_N)$ correspond to the height of the nodes. A sequence $\{v_k\}$ corresponds to a sequence of acyclic graphs subject to the definition of link orientation [4]. The set $S(v)$ gives the set of stuck nodes. Height increase function g assigns height to any node i such that the set of pairs $\{(h_i, i)\}$ for $i = \{1, 2, \ldots, N\}$ are ordered lexicographically. The function g should satisfy the assumptions (A1-A3) given by GB algorithms. For any node i we define neighbor oblivious functions g_i for both full and partial link reversal that coheres with the general class of the GB algorithms.

In the full reversal method, the function g_i is defined for all $v = ((h_1, 1), \ldots, (h_N, N))$ by

$$g_i(v) = \begin{cases} z(t_i) + h_i(t_i) & \text{if } i \in S(v) \\ (h_i, i) & \text{if } \notin S(v) \end{cases} \tag{10}$$

$z(t_i)$ satisfies the recursion given by equations (1)-(2).

In the partial reversal method, the function g_i is defined for all $v = ((h_1, 1), ..., (h_N, N))$ by

$$g_i(v) = \begin{cases} (\overline{h_i}, i) & \text{if } i \in S(v) \\ (h_i, i) & \text{if } \notin S(v) \end{cases} \tag{11}$$

where $\overline{h_i} = (1 + \alpha_i)\lambda(t_i) - h_i(t_i, \alpha_i)$, such that $\lambda(t_i)$ satisfies the recursion given by equations (3)-(5)

For each $v = (a_1, \ldots, a_N) \in V$ and $i = 1, \ldots, N$ both the functions for g_i satisfy
$g_i(v) > a_i \;\; if \;\; i \in S(v)$, by Lemmas IV.1 and IV.2.
$g_i(v) = a_i \;\; if \;\; i \notin S(v)$

Also for each $i \in \{1, \ldots, N\}$, and each sequence $\{v_k\} \subset V$ for which $i \in S(v_k)$ for an infinite number of indices's k, the sequence

$$\{a_i^0 + \sum_{r=0}^{k} [g_i(v_r) - a_i^r]\}$$

is unbounded in A_i, where a_i^r denote the coordinates of v_r and A_i is a countably infinite set which is totally ordered by a relation $<$ in the sense that for any two distinct elements a_1 and a_2 of A_i there holds $a_1 < a_2$ or $a_2 < a_1$, but not both. Therefore the function g_i satisfies the assumptions of the GB algorithms. Thus we have embedded our algorithms in the GB framework and have checked that the conditions A1, A2 and A3 for propositions P1 and P2 of GB hold. These in turn imply the claims we made just before Section III.

VI. Pseudo-events: Preemptive Link Reversal

Pseudo-events are generated via an independent Poisson process of rate (λ/N) at each node, where λ is the net rate at which pseudo-events are generated in the network and N is the number of the nodes in the network. Our results show that the end to end delay is significantly reduced by avoiding the voids. Further only a few pseudo-events are required to circumvent a local minimum condition.

A. Simulation setup

We use Qualnet 4.5 simulator to perform the simulations. The simulation setup is described as follows. Area of 100m x 100m is divided into N grids of equal size where N is the number of sensors in the network. Within each grid one sensor is placed uniformly. The sink is located at the origin (0,0) and the transmission range is considered to be 10m. The pseudo-events are generated at the rate of one event every 10 minutes. Area of the void is around 2600 sq.m. Simulations are run by changing the number of nodes. Figures 4(a) and 4(b) represents one instant of the node layout with 1000 nodes. The dark nodes are dead and will not participate in any network activity. We consider a transmitter initiated duty-cycled sensor

network where the probes are sent periodically to find greedy nodes. The other simulation parameters are

- Radio duty-cycle = 3.22 %
- Probe interval = 20 ms
- Pseudo-events generation rate = One every 10 minutes

B. Simulation results

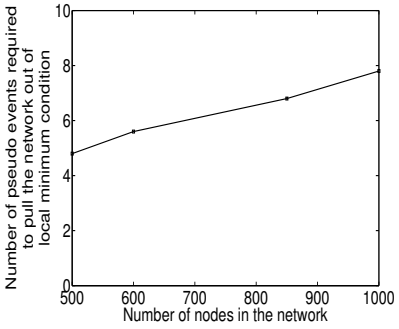


Fig. 2. Number of pseudo-events required to circumvent a local minimum condition

Simulations are run for various seeds and the y-axis of figure 2 represents the average number of pseudo-events required to pull the network out of a local minimum condition. A single pseudo-event does not guarantee link reversal at all the stuck nodes because of the reactive approach to link reversal. Only if a stuck node receives a packet will it initiate link reversal. It was generally observed that very few pseudo-events are sufficient to circumvent a local minimum condition.

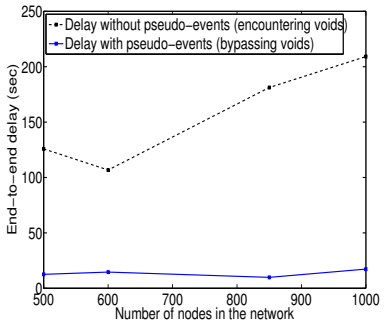
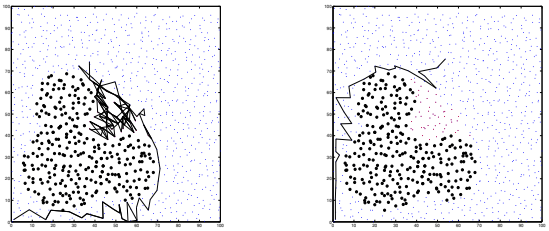


Fig. 3. End to end delay with and without pseudo-events

Figure 3 gives the end to end delay incurred by the packet to reach the sink with and without pseudo-events. This plot is for a particular hole given by figure 4(a). Figures 4(a) and 4(b) show paths with and without pseudo-events respectively.

VII. Conclusion

We proposed a neighbor oblivious link reversal scheme to get a network out of the local minimum condition in geographic routing without neighbor information. Our proposed algorithms fall within the class of GB algorithms [4]. We then saw the usefulness of pseudo-events in keeping the network in a prepared state in case of faults. Rough indications of number



(a) Path taken before pseudo-events (b) Path taken after pseudo-events

Fig. 4. The light dots in the figure 4(b) corresponds to the stuck nodes that have reversed their links during pseudo-events.

of the pseudo-events needed and the attendant gains in end-to-end delay were obtained via simulations.

References

[1] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2000, pp. 243–254. [Online]. Available: http://dx.doi.org/10.1145/345910.345953

[2] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing holes in sensor networks," *Mob. Netw. Appl.*, vol. 11, no. 2, pp. 187–200, 2006.

[3] B. Leong, B. Liskov, and R. Morris, "Geographic routing without planarization," in *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 25–25.

[4] E. M. Gafni, S. Member, Dimitri, P. Bertsekas, and S. Member, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Transactions on Communications*, vol. 29, pp. 11–18, 1981.

[5] C. Busch, S. Surapaneni, and S. Tirthapura, "Analysis of link reversal routing algorithms for mobile ad hoc networks," in *SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*. New York, NY, USA: ACM, 2003, pp. 210–219.

[6] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005, pp. 217–230.

[7] H. Kalosha, A. Nayak, S. Ruhrup, and I. Stojmenovic, "Select-and-protest-based beaconless georouting with guaranteed delivery in wireless sensor networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008, pp. 346–350.

[8] C.-Y. Chang, K.-P. Shih, S.-C. Lee, and S.-W. Chang, "Rgp: Active route guiding protocol for wireless sensor networks with obstacles," in *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, Oct. 2006, pp. 367–376.

[9] F. Yu, S. Park, Y. Tian, M. Jin, and S.-H. Kim, "Efficient hole detour scheme for geographic routing in wireless sensor networks," in *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, May 2008, pp. 153–157.

[10] P. Casari, M. Nati, C. Petrioli, and M. Zorzi, "Efficient non-planar routing around dead ends in sparse topologies using random forwarding," in *Communications, 2007. ICC '07. IEEE International Conference on*, June 2007, pp. 3122–3129.

[11] ——, "Alba: An adaptive load - balanced algorithm for geographic forwarding in wireless sensor networks," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, Oct. 2006, pp. 1–9.

[12] S. Chen, G. Fan, and J. Cui, "Avoid "void" in geographic routing for data aggregation in sensor networks," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 1, no. 4, pp. 169–178, 2006.

[13] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," 2003.