Localized Routing Approach to Bypass Holes in Wireless Sensor Networks

A. Mostefaoui, M. Melkemi and A. Boukerche

Abstract—Geographic greedy forwarding (GF) technique has been widely used by many algorithms for routing in sensor networks because of its high efficiency resulting from its local and memoryless nature. Hence, it ensures scalability which is a fundamental requirement for protocol applicability to large-scale sensor networks with limited resources. Nevertheless, GF suffers from a serious drawback when packets, based on geographic distance, cannot be delivered; i.e., the so called "local minimum phenomenon". This problem has been tackled in previous research works to guarantee packet delivery by routing around the boundaries of the hole but at an excessive consumption of control overheads. In this paper, we propose a novel approach that exploits GF technique and guarantees at the same time packet delivery (handles the local minimum situations). Our approach is of a local nature that does not retain memories and performs better than the state-of-the-art approaches in terms of its ability to guarantee packet delivery and to derive efficient routing paths. We provide in this paper proof of its correctness (packet delivery guaranty) while showing, through simulations, its performance effectiveness in terms of reducing path lengths, average end-to-end delays and overall energy consumption.

Index Terms—Geographic greedy routing, local minimum problem, hole traversal, distributed algorithms.

1 Introduction

Optimizing communications in Wireless Sensor Networks (WSNs), which are known to be the most energy consuming components [1], leads naturally to designing efficient localized multihop routing protocols capable of delivering packets originating from a source node and sent to the destination node with the lowest overhead possible. By localized, we mean that the decision to select the next hop performed by each node within the routing path, is based only on local knowledge (i.e., 1 hop neighbor information). To this end, GF strategy [2], as a simple, efficient and scalable approach fulfills this requirement. In this approach, every node is supposed to be aware of its geographical location, by means of a location service [3] or by deriving it from the localization phase [4], [5], initiated just after deployment. Hence, a source node, knowing the location of the destination node, sends the packet to its 1-hop neighbor which is the closest node to the destination among all neighbors. Knowing the destination location from the received message, every node in the path repeats this process until the message reaches the destination node.

It has been shown that geographic routing protocols are not only efficient in the average case; they are also optimal for worst-case localized routing protocols with the hop-counts of the resulting paths bounded by $O(d^2)$, where d is the distance between the source and destination [6].

This simple process does not induce any additional communication overhead since it requires that every node knows only its 1-hop neighbors. However, it suffers from a serious drawback called *local minimum phenomenon* [7]. This "case" occurs when the packet cannot be delivered to the next hop, when the current node is the closest one to the destination. Figure 1 shows the local minimum phenomenon, where node N_{local}, receiving the packet from node N_4 (originated from N_s) could not deliver it in the direction of N_d.

Many related routing algorithms were developed in the literature to solve or at least to reduce the local minimum problem [8], [9], [10]. These works fall under two categories: (a) graph based approaches and (b) non-graph-based approaches (see Section 7 for more details). Though these approaches resolve the local minimum problem, they require the entire network graph or parts of it to be locally stored for graph based approaches which leads thus to poor scalability. However, the non-graph-based approaches derive either long routing paths or make use of expensive techniques in terms of energy consumption (flooding for instance) to meet the requirement for no memory.

To illustrate, we consider the two most efficient approaches without memory, namely, Boundhole [11] and GAR (Greedy Anti-Void Routing) [12]. The be-

- *A. Mostefaoui is with FEMTO-ST Institute at Franche Comte University*
  1 Rue Engel Gros, Belfort 90000, France.
  E-mail: Ahmed.Mostefaoui@univ-fcomte.fr
- *M. Melkemi is with LMIA laboratory at Haute Alsace University,*
  4 rue des freres Lumiere, 68093, Mulhouse,
  E-mail: mahmoud.melkemi@uha.fr
- *A. Boukerche holds the Canada Research Chair at the University of Ottawa,*
  800 King Edward Avenue Ottawa, Ontario, Canada, K1N 6N5
  E-mail: boukerch@site.uottawa.ca

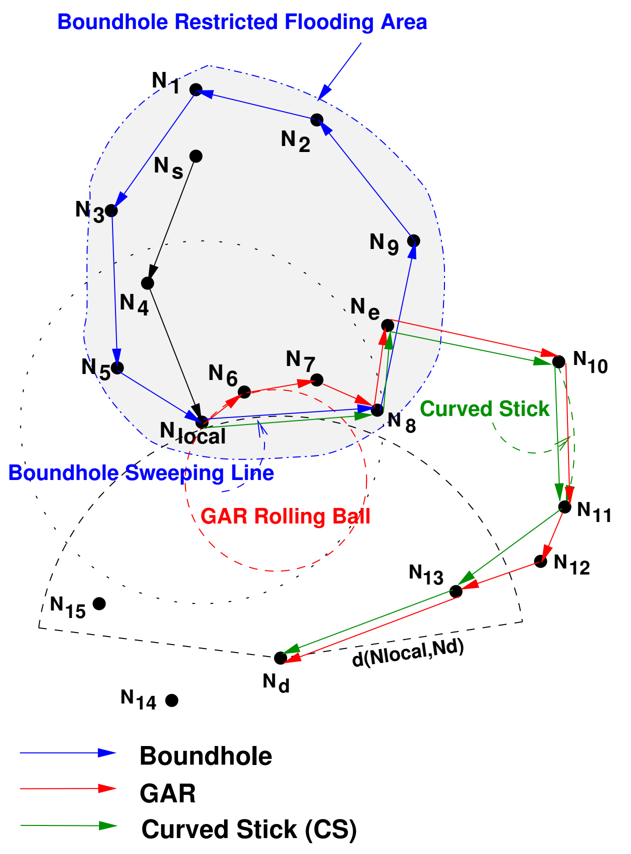

Fig. 1. Examples of constructing routing paths when local minima situation is met.

havior of these two protocols is illustrated into Figure 1. At each step, Boundhole approach uses a sweeping line to select the next hop. By doing so, it presents a great risk of succumbing to looping. This situation, also reported by the authors, comes from the fact that Boundhole could not, by construction, handle *intersection situations*. For instance, in our example, N_e could communicate with N_{10} while neither N_8 nor N_9 could communicate with N_{10}. That is why when node N_8 gets the packet, it selects node N_9 as the next hop rather than selecting node N_e, creating hence a false boundary. By repeating this process on all visited nodes (i.e., nodes: $N_{local}, N_8, N_9, N_2, N_1, N_3, N_5, N_{local}$), the packet returns to node N_{local} which is considered as the initiator node of Boundhole process.

Hence, in spite of the existence of a path, Boundhole is not able to deliver the message. This is particularly problematic for critical event surveillance applications for instance. To overcome the false boundary detection problem, the authors proposed a "brute force" technique which consists in **flooding the network** until the message gets out of the false boundary. In addition to its inherent huge overhead, flooding can also be initiated even when a path does not exist. In other terms, Boundhole is not able to decide when a path exists or not. As a consequence, flooding is used each time the message gets back to the initiator. Obviously, this proposed solution may generate a considerable routing overhead leading thus to a dramatic degradation in the overall network performance, and in particular on end-to-end delays and energy consumption.

GAR protocol has been specially designed to alleviate this situation. It operates differently by making use of a rolling-ball instead of a sweeping line. Similarly to Boundhole approach, GAR uses GF until a packet reaches a node that is a stuck node; i.e., the local minimum situation is met. From this node, GAR uses a rolling ball, with a range equal to half of the communication range of the nodes ($R/2$). This rolling ball (dashed red circle in Figure 1) is hinged at the current node and moves counterclockwise. The first node hit by the rolling ball among the neighbors of the current node is selected as the next hop. The rolling ball is then hinged at this node and this process is repeated while the distance between the current node and the destination node is greater than the distance between the node which is stuck and the destination node. For instance, in the example of Figure 1, when the packet, originated from node N_s, arrives at node N_{local}, the next hop selected by GAR protocol is N_6. The rolling ball is then hinged at node N_6 and moves again selecting node N_7 as the next hop and so on until it arrives at node N_{13} which is closer to node N_d than N_{local}. From this node, GF is used. The generated routing path is then the following: $\{N_s, N_4, N_{local}, N_6, N_7, N_8, N_e, N_{10}, N_{11}, N_{12}, N_{13}, N_d\}$.

Even though GAR protocol resolves the false boundary detection problem, as shown in Figure 1, between nodes N_8 and N_9 by selecting node N_e, it visits however, by construction, unnecessary nodes like N_6, N_7 and N_{12}. The generated paths are then longer which will impact negatively the overall network performance in terms of routing efficiency (end-to-end delays for instance) and energy consumption.

Our contribution in this paper is to propose a novel approach, named ***Curved Stick*** (CS) that prevents, by construction, the false boundary detection problem observed previously within Boundhole approach, while deriving more efficient routing paths than GAR protocol. We outline the basic idea behind our approach in Figure 1. Unlike Boundhole approach, we used a curved stick instead of a sweeping line to select the next hop. For instance, in Figure 1, node N_{local} will select node N_8 as its next hop, avoiding hence the selection of unnecessary nodes as is the case within GAR protocol. Similarly, node N_8 will select node N_e as its next hope rather than selecting N_9 as was the case in Boundhole approach. By doing so, CS ensures that it will not succumb to looping (avoiding the false boundary detection problem) while it limits highly the number of visited nodes. The generated routing path with CS is then the following: $\{N_s, N_4, N_{local}, N_8, N_e, N_{10}, N_{11}, N_{13}, N_d\}$.

We provide, in this paper, proof of the correctness of our proposed approach and show through simulation its effectiveness compared to the state-of-the-art approaches; we demonstrate that our approach derives shorter routing paths, reduces the average end-to-end delays observed between communicating nodes and decreases the overall energy consumption of the network.

The rest of the paper is organized as follows: Section 2 provides details about the used network model and outlines the problem of interest. The proposed CS approach is detailed in Section 3. Section 4 presents proof of the correctness of CS whilst Section 5 details its implementation. In Section 6, we present the results for simulations we have conducted in order to evaluate the proposed approach and to compare its performance to the state-of-the-art approaches. Section 7 provides a literature overview of works that have addressed routing through holes in sensor networks. Finally, Section 8 concludes this paper.

2 Network Model and Problem Statement

We consider a geographic wireless network in which all nodes are homogeneous and static with R as their communication range. The network can then be represented by a Unit Disc Graph $G(N, E)$, where $N = \{N_1, N_2, \dots, N_n\}$ is a finite set of sensor nodes and $E = \{E_{i,j} \mid d(N_i, N_j) \leq R\}$ is a finite set of links. Note that $d(N_i, N_j)$ stands for the Euclidean distance between nodes N_i and N_j. The locations (x, y) of the set N are known (each node is aware of its location by means of a positioning system like GPS or as a result of the localization process [4], [5]). We also note for every node N_i, its neighbors set $V(N_i) = \{N_j \mid E_{i,j} \in E\}$; i.e., nodes are also aware of their neighboring nodes with the corresponding locations. We further assume that the location of the destination node is known by the source node. Thus, intermediate nodes can only know the location of the destination node by receiving the packet from the source node.

Under these assumptions, when two nodes want to communicate (and ask for a routing path), GF approach could be performed by each intermediate node based only on local information until the packet reaches the destination node or falls into the local minimum problem as mentioned before. When the local minimum situation is met (i.e., the packet is on the boundary of a hole) CS approach is used to get the packet out of the hole by traversing hole boundary nodes. Hence, the main issue is how to design a boundary traversal algorithm that ensures true boundary detection while at the same time deriving efficient communication paths in terms of the number of hops?

Before diving into the details of our proposal, we first introduce some definitions and clearly state the false boundary detection problem.

Definition 1: (Hole)
*We define a hole as a cyclic sequence of nodes: $H = \{N_i, N_{i+1}, ..., N_{i+j}, N_i\}$ so that the closed region bounded by this **non self-intersecting** polygonal sequence is empty of any node.*

For instance, in our stated example, the sequence $S_1 = \{N_{local}, N_6, N_7, N_8, N_e, N_9, N_2, N_s, N_4, N_{local}\}$ is considered as a hole, whereas the sequence $S_2 = \{N_{local}, N_6, N_7, N_8, N_9, N_2, N_s, N_4, N_{local}\}$ is not; this is because node N_e is located inside the polygon S_2.

Definition 2: (Boundary Node)
We define a node as a boundary node if it is located on the boundary of the network or of a hole inside the network.

Now we introduce the communication intersection situation which is behind the false boundary detection problem in Boundhole approach.

Problem 1: (Communication Intersection Problem)
A Communication intersection situation occurs when a link $E_{a,b}$ exists between two boundary nodes N_a and N_b which is "crossed" by another link $E_{c,d}$ where N_d is not visible neither to N_a nor to N_b.

Figure 2 illustrates the false boundary detection problem.

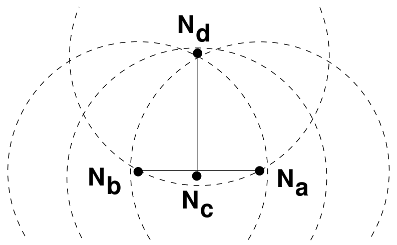

Fig. 2. Communication intersection problem.

Our objective here is to develop an approach that handles **by construction** the intersection problem while it guarantees at the same time packet delivery and generates efficient routing paths.

3 Proposed CS approach

The main idea behind our proposed approach is to characterize well nodes responsible for intersection situations when hole/network boundaries are traversed so that these nodes will be directly selected by the previous hop node. Of course, the localized nature of the proposed approach must hold; i.e., no extra-information other than the information mentioned before is needed. This means that each node knows only its immediate neighbors and their locations and based on this it will select the next hop. To this end, we have not made use of a sweeping line which generates, as mentioned before, the intersection situation; but, we used a curved stick that selects nodes behind intersection situations as shown in Figure 1 (node N_e is selected instead of node N_9).

3.1 Curved Stick Form

The concerns that follow immediately are: the need to determine the optimum form of this curved stick in order to select, on one hand, nodes only responsible for intersection situations; and, on the other hand, the question of whether or not this curved stick will "miss" intersection cases. In other terms, there is a need to determine how to reduce path lengths while preventing intersection situations. We begin by introducing the following definitions.

Definition 3: (Potential-Exit-Gate Node)
We say a node N_c is a Potential-Exit-Gate (PEG) node in the direction $\angle N_aN_cN_b$ if there exists a location $X \in \mathbb{R}^2$ in the transmission range of N_c and outside the transmission ranges of N_a and N_b which are boundary nodes.

For instance, in Figure 3, node N_c is considered as a PEG. The locations X are represented by the grey region in Figure 3(a).

Definition 4: (Exit-Gate Node)
We say a node N_c is an Exit-Gate (EG) in the direction $\angle N_aN_cN_b$ if there exists a node N_d in the transmission range of N_c and outside the transmission ranges of N_a and N_b.

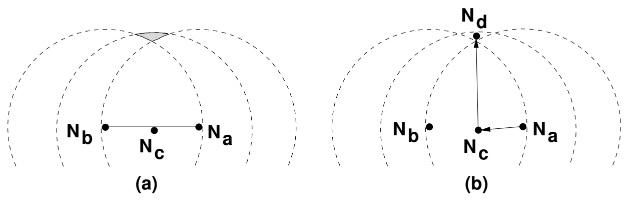

Fig. 3. (a) N_c is a PEG, if the grey region is empty of nodes; (b) N_c is an exit-gate.

Ideally we are looking for a distributed approach that allows nodes to make distinctions between PEG and EG nodes and to select only the latter as next hops. Unfortunately this cannot be possible without providing nodes with extra-information, such as 2-hop neighbors for instance. However, constructing 2-hop tables at each node is an energy consuming process and more importantly is not adequately scalable. For this reason, no extra-information is considered in our proposed approach. Rather, we will not make distinctions between PEG and EG nodes. We note that EG nodes are a sub-set of PEG nodes; i.e., each EG node is also a PEG node.

Hence, our objective is to characterize PEG nodes and to select them when performing next hop selection, without missing any one, otherwise intersection problems occur. Later on, we will use the notation of Figure 4.

Property 1: (Angle Characterization of a PEG Node)
Let N_c be a node in the left side of the line oriented from N_a to N_b. Node N_c is a PEG if and only if $\angle N_aN_cN_b > \theta_0$ where $\theta_0 > \pi/2$ and

$$\sin\theta_0 = \frac{d(N_a, N_b)}{2R}. \qquad (1)$$

Proof: Let us refer to Figure 4. Consider a point N_0 such that the discs of radii R, centered at N_a, N_b and N_0, intersect at the same location X_0. Therefore, we have:

$$d(X_0, N_b) = d(X_0, N_a) = d(X_0, N_0) = R \qquad (2)$$

We first show that $\angle N_aN_0N_b = \theta_0$: we observe that $\gamma = (\pi - 2\alpha) + (\pi - 2\beta) = 2\pi - 2(\alpha + \beta) = 2\pi - 2\angle N_aN_0N_b$. Then $\sin\frac{\gamma}{2} = \sin(\angle N_aN_0N_b)$, where $\alpha = \angle X_0N_bN_0 = \angle X_0N_0N_b$, $\beta = \angle X_0N_cN_a = \angle X_0N_aN_c$ and $\gamma = \angle N_aX_0N_b$.

On the other hand, as the triangle $N_aN_bX_0$ is isosceles then

$$\sin\frac{\gamma}{2} = \frac{d(N_a, N_b)/2}{R} = \sin\theta_0 \qquad (3)$$

We obtain $\sin(\angle N_aN_0N_b) = \sin\theta_0$. As $\angle N_aN_0N_b > \pi/2$, therefore $\angle N_aN_0N_b = \theta_0$.

Now, let us prove the first implication. If N_c is a PEG node then the communication range of N_c contains the point X_0; therefore, N_c is necessarily in the disc centered at X_0 (the dotted blue disc); thus $\angle N_aN_cN_b > \angle N_aN_0N_b = \theta_0$.

Conversely, if the angle $\angle N_aN_cN_b > \theta_0 = \angle N_aN_0N_b$ then N_c is in the disc centered at X_0. Consequently, the communication range of N_c contains the point X_0; hence, N_c is a PEG. □

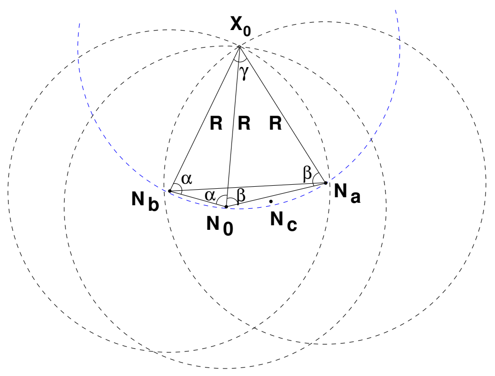

Fig. 4. Characterization of a PEG node N_c.

This property could be used by the current node (which has to select its next hop) as a criterion to reduce the number of candidates for the next hop (i.e., specifically PEG nodes). This is not the case for instance, for GAR approach in which the use of a rolling ball of $R/2$ radius is not relevant since it considers non PEG nodes. This important drawback of GAR approach is illustrated in Figure 5 where the rolling ball passes through many nodes that are not PEG nodes leading therefore to unnecessarily longer routing paths. The derived path using GAR approach is $N_a, N_1, N_2, \ldots, N_b$ whereas the use of Property 1 ensures that none of these nodes are PEG nodes and then avoids selecting them.

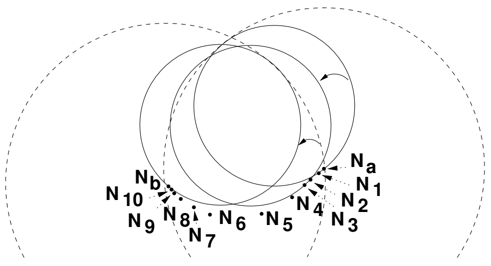

Fig. 5. The solid circles are examples of GAR; where the generated path is $\{N_a, N_1, ..., N_{10}, N_b\}$.

Property 2: (Region Characterization of PEG nodes)
All PEG nodes are located in the left side of the segment $[N_a, N_b]$ and inside the disc of radius R passing by nodes N_a and N_b.

Proof: When $\angle N_a N_c N_b = \theta_0$, the node N_c is on the boundary of the disc D of radius R and passing by N_a and N_b (it is the disc centered at point X_0, as shown in Figure 4). Therefore, all the nodes N_c that are located in the left side of the segment $[N_a, N_b]$ and inside the disc D form an angle $\angle N_a N_c N_b > \theta_0$. On the other hand, Property 1 establishes that a PEG node N_c is characterized by $\angle N_a N_c N_b > \theta_0$. Hence, the statement of this property is valid. □

The interest in Property 2 comes from the fact that it precisely defines the location region of all PEG nodes with regards to the current hop node. Therefore, it is sufficient for this node to just *"sweep"* this region when looking to a PEG node to select. If this region is empty, this means that there is no node that could rise to an intersection situation; however, if there is at least one PEG, the sweeping process will certainly select it. The important issue then becomes how the current hop node could **sweep exactly** this region?

Our answer is to use a *Curved Stick (see Figure 6) which has the form of an arc with a radius of R and which is hinged at the current node*. Hence, we ensure that only this region is visited and more importantly that if there is a PEG node, it is selected first avoiding therefore the intersection situations.

Definition 5: (Curved Stick)
We define the Curved Stick, hinged at node N_a and noted $CS(N_a, N_b)$ (see Figure 6), as the arc of radius R and passing by the two points N_a and N_b.

Theorem 1:
CS approach resolves the false boundary detection problem.

Proof: We use the notation of Figure 2. Assume there exists a node N_c which is a PEG node in the direction $\angle N_a N_c N_b$ and that this node has not been

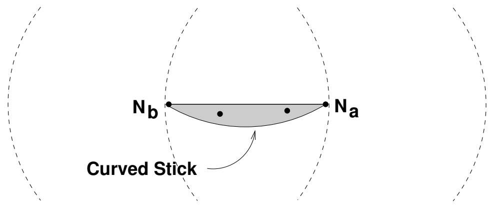

Fig. 6. The grey region is the PEG locations.

selected by CS. This means that the next hop of node N_a is N_b. However, from the description of CS, the region delimited by the segment $[N_a, N_b]$ and the arc of the circle of radius R which has its center in the direction $\angle N_a N_c N_b$ and passes by N_a and N_b is **empty** of nodes. Consequently N_c is located outside this region. This means that the set $\{X \in \mathbb{R}^2 \mid d(X, N_c) \leq R \;\; and \;\; d(X, N_a) > R \;\; and \;\; d(X, N_b) > R\}$ is empty, which contradicts the assumption that N_c exists. □

3.2 CS Routing Algorithm

Having proved that the proposed CS resolves the false boundary detection problem, we provide details below on the proposed CS routing algorithm. Three phases characterize our proposal: (a) engaging phase, (b) CS boundary traversal phase and (c) termination phase.

3.2.1 *Engaging phase*

Like other approaches, CS algorithm is of a greedy nature: the message between two communicating nodes is geographically forwarded hop by hop until it reaches the destination node or falls into the local minimum situation problem. When the message gets stuck (i.e., it cannot be delivered) the current hop node, called *the initiator node (N_{init})*, starts applying CS boundary traversal rule.

3.2.2 *CS Boundary Traversal Rule*

To apply CS boundary traversal rule, each visited node needs the location of the initiator as an input from the previously visited hop node. The node in question first computes the two distances e.g., its own distance to destination and the one from the initiator to destination. If it is closer to the destination than the initiator, this means that the message will get out of the local minimum problem. In this case, the current node starts applying GF policy. Otherwise, the message is still on the boundary of a hole and CS rule has to be applied. To this end, the current node determines its *Starting Point (SP)* (see below), from which point the curved stick is swept until a node is hit. The latter is then selected as the next hop and notified by the current node.

Definition 6: (Starting Point)
We define the Starting Point of node N_{i+1}, noted SP_{i+1}, as the intersection point between the two circles of range

R, centered at N_i and N_{i+1} and following the direction $\angle N_{i-1}N_iN_{i+1}$ where N_{i-1} is the previous CS hop of node N_i and N_{i+1} is the next CS hop of N_i.

The Starting Point is used to prevent the generation of link intersections, as illustrated in Figure 7, and to then guarantee the progress at each step. However, for the initiator node that does not have a previous CS hop (the message comes from a previous GF hop), its corresponding starting point, noted as ISP_{init}, is considered as the intersection point between the circle of range R, (centered at this node) and the segment between N_{init} and the destination (see Figure 7). Note that we differentiate between SP_{init} which corresponds to the starting point of node N_{init} when the message comes from a previous CS hop (node N_{cs} in Figure 7) and ISP_{init} when the message comes from a previous GF hop (node N_g in Figure 7). This difference is essential for the termination of the algorithm as explained below.

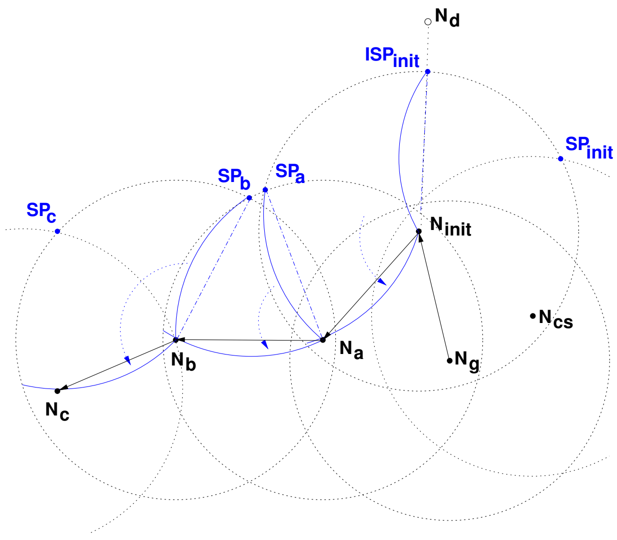

Fig. 7. CS Rule

3.2.3 Termination Phase

For the termination of the algorithm, there are only two possible cases:

1) In the first case, the message is received by the destination. This means that there is at least one routing path between the source node and the destination node.
2) In the second case, the message travels the whole boundary using CS rule and returns back to the initiator node. In this case, the initiator sweeps the curved stick from its SP_{init} to its ISP_{init}. If a node is hit, this node is determined as the next hop and then CS process continues; otherwise the initiator node cannot apply CS rule and it deduces that there it does not exist any route from the source node to the destination node (see the demonstration in the next section). In this case, the initiator node informs the source node that the message could not be delivered.

Algorihm 1 summarizes the proposed routing algorithm.

Algorithm 1 CS routing algorithm.

```
Require: Receive message M from the previous hop
 1: if (M.Initiator.ID = null) then
 2:     // GF policy is used
 3:     select the next hop based on GF;
 4:     if (Local minimum problem occurs) then
 5:         set M.initiator.ID to MyID
 6:         select the next hop based on CS rule
 7:     end if
 8: else
 9:     if M.Initiator.ID $\neq$ MyID then
10:         compute: $d(N_{current}, N_{dest})$ and $d(N_{init}, N_{dest})$
11:         if $d(N_{current}, N_{dest}) < d(N_{init}, N_{dest})$ then
12:             // the message is out of the hole
13:             set M.initiator.ID to $null$
14:             select the next hop based on GF
15:         else
16:             select the next hope based on CS rule
17:         end if
18:     else
19:         // the message traveled the whole boundary and
            returns to the initiator
20:         sweep the curved stick from $SP_{init}$ to $ISP_{init}$
21:         if a node is hit then
22:             select the next hop based on CS rule
23:         else
24:             inform the source node that the message could
                not be delivered
25:         end if
26:     end if
27: end if
28: send the message
```

3.3 Hop Count Reduction

In the previously provided description of CS routing algorithm, we have considered only one direction of sweeping for the curved stick (e.g., counterclockwise). Nevertheless, exploring the other sweeping direction (clockwise) could be in some cases more effective in terms of hop count reduction, as illustrated in Figure 8. For instance, the path derived by counterclockwise sweeping at node N_{local_1} is longer (11 hops) than the other two paths (9 and 10 hops) derived by adopting a clockwise sweeping direction. To include this optimization in our approach, we have adopted the following rule: each time the message falls into the local minimum problem, the initiator node *"clones"* it into two messages; the initiator node includes in each message the hop count and starts applying CS rule, one in each direction; i.e., the initiator node selects one next hop in counterclockwise direction and the other in the clockwise direction. In Figure 8, for example, two local minimum situations are encountered when routing from node N_s to node N_d. At the first stuck node N_{local_1}, the two next hops are selected: N_{10} in the clockwise direction and N_2 in the counterclockwise direction. Similarly, at node N_{local_2}, where the local minimum phenomenon occurs again, two nodes are selected as next hops with their associated sweeping sense (N_{16} and N_{14}). Finally, when the destination node receives all possible paths, it selects the shortest one in terms of hop count and sends back the information through this path to the source node. In our example, the selected path is: $\{N_s$,N_1, N_{local_1},N_{10},N_{11},N_{12},N_{local_2},N_{14},N_{15}, $N_d\}$.

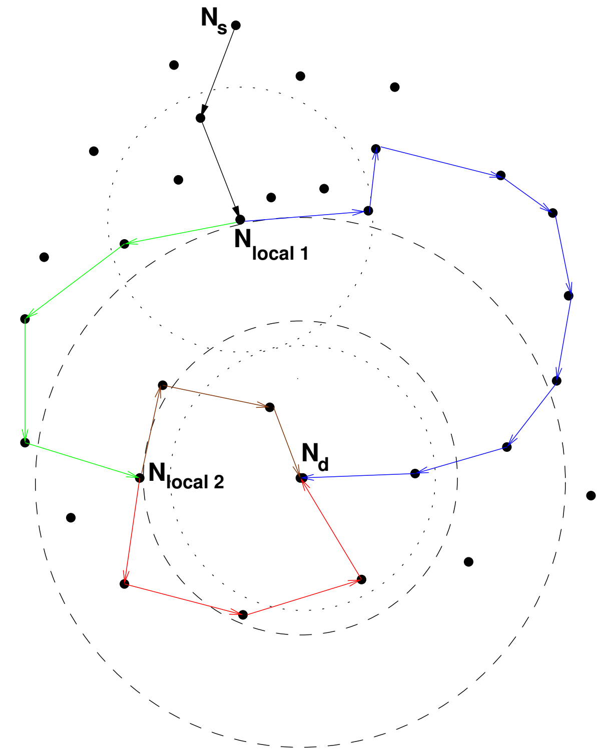

Fig. 8. Hop count reduction.

4 Proof of Correctness

In this section, the correctness of CS approach is proven. In order to guarantee packet delivery, we first prove that CS approach generates a finite sequence of links which means that the algorithm terminates. From this point, we secondly prove that CS is able to find at least one path connecting the source node to the destination node where such path exists. Conversely, if any path between the two communicating nodes does not exist, CS is able to detect and to notify the source node.

Property 3: (Link Intersection)
Two links generated by CS scheme do not intersect.

Proof: Assume that two links $E_{i,j}$ and $E_{k,l}$ generated by CS approach intersect. In this case, one node, lets say N_k, could neither communicate with N_i nor with N_j; otherwise it would have been hit previously by the curved stick when N_i or N_j performed CS approach. This case is the situation studied before in Figure 2 and from which we have proven (see Theorem 1) that CS approach resolves by generating two links that do not intersect; i.e., $E_{i,j}$ could not intersect with $E_{k,l}$ which contradicts the initial assumption. □

Property 4: (Consecutive Angles)
The angle between two consecutive links in CS sequence $\angle N_{i-2}N_{i-1}N_i$ is greater than or equal to $\pi/3$.

Proof: We have to consider two cases illustrated in Figure 9:

- **Case (a):** node N_i is visible to node N_{i-2}; i.e., N_i is within the communication range of N_{i-2}. In this situation, illustrated in Figure 9(a), the angle $\angle N_{i-2}N_{i-1}N_i$ is greater than θ_0 (see Property 1) which in turn is greater than $\pi/3$.
- **Case (b):** node N_i is not visible to node N_{i-2}. Here, the smallest angle associated with the possible location of node N_i is the intersection point between the communication zones of N_{i-2} and N_{i-1} (see Figure 9(b)). Hence, the distance between the three nodes is equal, which means that the smallest value of $\angle N_{i-2}N_{i-1}N_i$ is $\pi/3$.

□

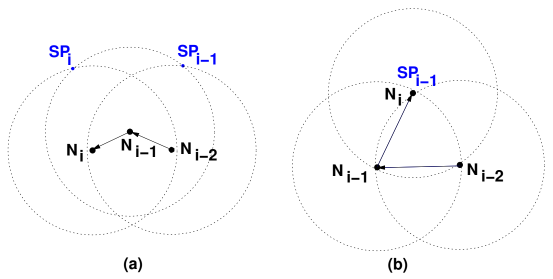

Fig. 9. Two possible locations of the next CS hop.

Theorem 2:
CS scheme terminates and generates a finite sequence of links.

Proof: CS algorithm terminates if any node will not appear in the sequence infinitely. In fact, each node can only appear in the sequence at most 6 times. Consider the sequence $S = \{N_0N_1\cdots N_iN_{i+1}\cdots N_jN_{j+1}\}$ $(i < j)$. Assume that $N_i = N_j$, that is the message gets back to N_i which it has already visited. CS scheme ensures that the angle $\angle N_{i-1}N_iN_{i+1}$ is smaller than the angle $\angle N_{i-1}N_iN_j$, as illustrated in Figure 10 (the grey area swept by the curved stick is empty of nodes). Also node N_{j+2} does not belong to the grey area, consequently the angle $\angle N_{i-1}N_iN_{i+1}$ does not overlap with the angle $\angle N_jN_iN_{j+2}$. Two illustrations of the possible locations of N_{j+2} are shown in Figure 10: one on the right side and the other on the left side of the line directed from N_j to N_i. Since the two angles $\angle N_{i-1}N_iN_{i+1}$ and $\angle N_jN_iN_{j+2}$ do not overlap and are at least equal to $\pi/3$, as shown in Property 4, N_i can then appear at most 6 times in the sequence. □

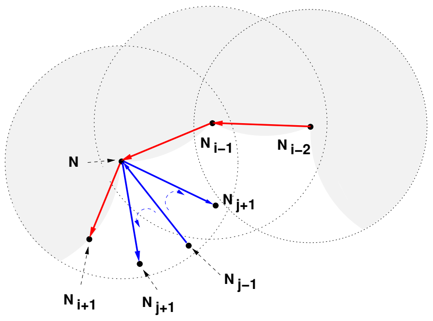

Fig. 10. Related to proof of Theorem 2

Theorem 3:
Given a source node N_s and a destination node N_d, if there exists at least one path between N_s and N_d then CS algorithm guarantees packet delivery.

Proof: Because of the greedy nature of CS algorithm, if there does not exist any local minimum between N_s and N_d, then the packet is guaranteed to be delivered. Now, consider the case where GF algorithm meets a stuck node N_{local} which satisfies $d(N_{local}, N_d) < d(N_k, N_d)$ for all $N_k \in V(N_{local})$ where $V(N_{local})$ is the set of N_{local} neighbors. Node N_{local} then starts CS boundary traversal rule. Assume that the generated sequence of links is: $N_{local} \ldots N_{end}$. As proven before, Theorem 2 ensures that this is a finite sequence. Now there are only two cases to consider:

- **Case 1:** $N_{local} \neq N_{end}$. By construction, this means that $d(N_{local}, N_d) > d(N_{end}, N_d)$. Hence GF could be used again to reach N_d.
- **Case 2:** $N_{local} = N_{end}$. In this case, the curved stick draws a closed curve which divides the plane into two simply connected regions Z and its complementary $\bar{Z}$ (see Figure 11). The region Z contains N_{local} and $\bar{Z}$ contains node N_d. In addition, all the nodes connected to N_{local} are in Z. Consequently Z contains N_s. Also, since the area swept by the curved stick (the grey area in Figure 11) is empty, there is no connection then between nodes belonging to the region Z and between those located outside Z. Therefore, there is no path connecting N_s to N_d.

□

5 CS Approach Implementation

While it is not difficult to explain the continuous sweeping of the curved stick, additional efforts must be done to develop a concrete and efficient method to implement this mechanism. In this section, we provide details on our implementation of CS approach, in particular we formally prove its correctness. Furthermore, we study its complexity and show that it easily fits the requirements of limited devices such as sensor nodes.

As explained in previous sections, CS approach relies on GF mechanism until a local minimum situation

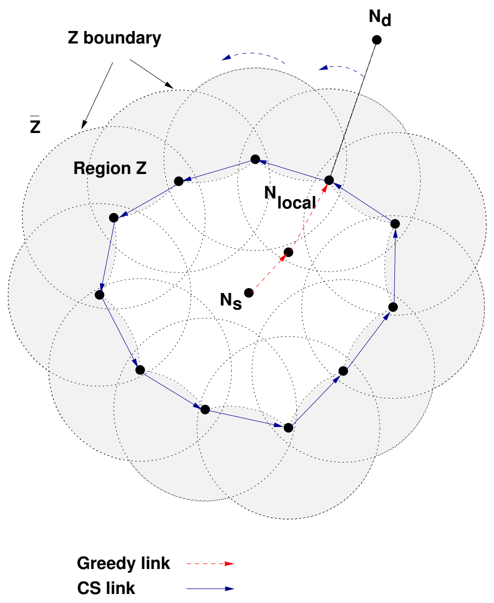

Fig. 11. CS approach sweeps the bounded region Z when there is no path connecting N_s to N_d

is encountered. The greedy algorithm implementation is straightforward and requires only one hop neighbor tables already available at each node, as stated earlier. We then shall here skip details about GF implementation.

We recall that CS boundary traversal algorithm is triggered once the local minimum situation is met, as shown in Section 3.2 (cf. phase 2). We consider node N_{init} as CS boundary traversal initiator, e.g., local minimum node (see Figure 7 for illustration). Furthermore, we consider for the rest of this section that the current boundary traversal orientation is counterclockwise[1]. Hence, node N_{init} starts CS sweeping from position ISP_{init} until a node, (e.g., node N_a), is hit. This means that N_a is considered as the next hop and it has to repeat the same procedure (CS sweeping) again, and so on. Before diving in implementation details, we first introduce the following observations and definitions.

Observation 1:
For each node N_i, involved in CS process, the region delimited by segment $[N_i, SP_i]$ and Curved Stick $CS(N_i, SP_i)$ is empty of nodes.

This observation remains valid for all nodes on the boundary traversal, including the initiator node. In fact this region was already swept by the previous hop and hence is empty of nodes, as shown in Figure 7. For the particular case of node N_{init}, which has no previous CS hop, this remains true because it is a local minimum node and therefore this region, as defined previously, is empty of nodes. The basic idea behind this observation is to state that when a node starts CS sweeping from this position, it has not *"forgot"* any

1. The generalization to clockwise orientation is straightforward.

node behind it (ensuring hence boundary traversal). From this position, each node sweeps the curved stick until a neighbor is hit. This position will be used later on as a reference for the comparison of angles presented below.

To explain how the current node selects only one neighbor as the next hop, we introduce an order relation between all its neighbors, defined by the sweeping of the curved stick. Starting from SP_i, an entire rotation of the curved stick ensures that all neighbors will be hit in an order related to their localization in the current node communication range.

Definition 7: (Hit Order)
For any node N_i, we define **Hit Order** *relation, noted $\prec$, over the set $V(N_i)$ (set of N_i neighbors) as the order in which neighbors are hit by the curved stick starting from SP_i.*

For instance in Figure 12, we have determined the following hit order: $N_1 \prec N_2 \prec N_3 \prec N_4 \prec N_p$. Hence, our objective is to find a method that is able to select the first element, subject of our interest, from this set. To achieve this, we have used what we call Projection Point.

Definition 8: (Projection Point)
For every neighbor N_j of the current node N_i, we define its corresponding Projection Point, noted P_j, as the point located on the circle of radius R and centered at point N_i so that Curved Stick $CS(N_i, P_j)$ contains node N_j.

Figure 12 illustrates an example of node N_i neighbors with their corresponding projection points.

Observation 2:
Several neighbors could have the same projection point.

This could happen when two or more nodes are located on the same curved stick as is the case for nodes N_4 and N_5 of Figure 12.

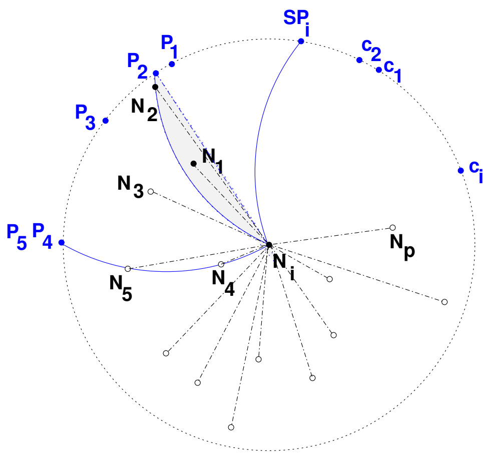

Fig. 12. Projection Point Illustration.

We now present how we have implemented CS approach. The main steps are summed up in Algorithm 2.

The next subsections deal with the correctness proof of our implementation as well as its time computation complexity.

Algorithm 2 CS next hop selection.

```
Require: Previous CS hop $N_p$, current hop $N_i$
1: Compute the corresponding $SP_i$.
2: Order all the neighbors, $N_j \in V(N_i)$, according to their increasing angles $\angle SP_iN_iN_j$ using counterclockwise direction.
3: Select the neighbor, $N_f$, corresponding node to the smallest angle. In the case where there are more than one corresponding to the smallest angle, select then the node which is farthest away.
4: Compute the projection point $P_f$ of node $N_f$.
5: For every neighbor $N_j$ belonging to the circle $C_f$ and located on the left side of the directed line $[N_i, P_f]$, compute its corresponding projection point, $P_j$. Recall that $C_f$ is the circle of radius R passing by nodes $N_i$ and $N_f$ and the position $P_f$.
6: Select the projection point $P_s$ corresponding to the smallest angle $\angle SP_iN_iP_s$ among the angles $\angle SP_iN_iP_j$ using counterclockwise direction. $P_s$ is the projection point of the next hop. In the case where there are more than one node having $P_s$ as their projection point, select the node which is the farthest away.
```

5.1 Proof of Correctness of Algorithm 2

To prove the correctness of our implementation using projection points, we have to prove first that it preserves the order relation *Hit Order* defined previously. In other terms, we have to prove the following property:

Property 5: (Order Preservation)
$\forall$ $N_x, N_y \in V(N_i)$, we have: $N_x \prec N_y$ if and only if $\angle SP_iN_iP_x \leq \angle SP_iN_iP_y$

Proof: We prove the first implication; i.e.,

- $N_x \prec N_y \Rightarrow \angle SP_iN_iP_x \leq \angle SP_iN_iP_y$

We denote by C_k the circle passing by nodes N_i, N_k and the position P_k and centered at position c_k. In the same way, we denote C_i as the circle containing $CS(N_i, SP_i)$ and centered at position c_i (See Figure 12 for illustration). Here, we have to consider two cases: the case where $P_x \neq P_y$ and the case where $P_x = P_y$. In the first case we have: $N_x \prec N_y \Rightarrow N_y \notin C_x$ as N_x was hit first. Consequently, we have:

$$\angle c_iN_ic_x < \angle c_iN_ic_y \tag{4}$$

On the other hand, if c_x is located before SP_i, we have:

$$\angle c_iN_ic_x + \angle c_xN_iSP_i = \angle c_xN_iSP_i + \angle SP_iN_iP_x = \pi/3 \tag{5}$$

Otherwise, if c_x is located after SP_i, we have:

$$\angle c_iN_ic_x - \angle SP_iN_ic_x = \angle SP_iN_iP_x - \angle SP_iN_ic_x = \pi/3 \tag{6}$$

which results in: $\angle c_iN_ic_x = \angle SP_iN_iP_x$. Similarly, we can derive that: $\angle c_iN_ic_y = \angle SP_iN_iP_y$. Finally, we obtain:

$$\angle SP_iN_iP_x < \angle SP_iN_iP_y \tag{7}$$

In the second case, where $P_x = P_y$, we have $\angle SP_iN_iP_x = \angle SP_iN_iP_y$ which ends the proof of the first implication.

We now prove the second implication; i.e.,

- $\angle SP_iN_iP_x \leq \angle SP_iN_iP_y \Rightarrow N_x \prec N_y$.

Inversely, $\angle SP_iN_iP_x \leq \angle SP_iN_iP_y \Rightarrow \angle SP_iN_ic_x \leq \angle SP_iN_ic_y$, which leads to $N_x \prec N_y$. $\square$

Theorem 4:
Algorithm 2 returns the first neighbor hit by CS sweeping.

Proof: Assume that Algorithm 2 returns node N_x which is not hit first by CS sweeping. We denote N_y as the node which is hit first. This means that $N_y \prec N_x$. Algorithm 2 returns N_x which implies that $\forall N_j \in V(N_i)$, $\angle SP_iN_iP_x \leq \angle SP_iN_iP_j$, and in particular $\angle SP_iN_iP_x \leq \angle SP_iN_iP_y$. However, Property 5 states that: $\angle SP_iN_iP_x \leq \angle SP_iN_iP_y \Rightarrow N_x \prec N_y$ which contradicts the initial assumption. $\square$

5.2 Projection Point Computation and Time-Complexity Analysis

Step 1 implementation in Algorithm 2 (e.g., Starting Point) consists first in finding the two intersection points between two circles centered in the previous hop and in the current hop with the same radius R. After, by using angle comparison, the starting point is selected among the two points; i.e., the smallest angle starting from the previous node if the current orientation is counterclockwise or otherwise from the greatest angle. Similarly, the computation of projection points (Step 4 and 5) consists also in computing the intersection of circles and in using angle comparison to select the correct points.

We note that the starting point (Step 1) and the projection point P_f (Step 4) are computed in constant time (they need only a calculation of the intersection between two discs). Step 2 and Step 3 perform sorting of n angles and select N_f which corresponds to the smallest one (e.g. N_2 in Figure 12). Thus, the required time is $\mathcal{O}(n \log n)$, where n stands for the number of the current node neighbors. Sorting the angles allows us to avoid scanning all the neighbors in order to identify, in Step 5, the nodes belonging to the narrow grey area illustrated in Figure 12. Indeed, these nodes are present in the top of the sorted list. In the worst case which occurs when all the neighbors are in the grey region, Step 5 costs at most $\mathcal{O}(n)$.

Furthermore, when the nodes are randomly distributed, we observe that only a very small subset (computed in Step 5) of the current node neighbors are concerned with projection point computation. For instance, in the example of Figure 12, only N_1 and N_2 are candidates for projection point computation, the rest of the nodes are ignored. Finally, Step 6 computes the smallest projection point among those computed in Step 5; hence the required time to find this point does not exceed $\mathcal{O}(n)$ in the worst case.

With the exception of Step 2 accomplished in $\mathcal{O}(n \log n)$ time, all the other steps are accomplished either in at most $\mathcal{O}(n)$ (e.g., Steps 5 and 6) or in $\mathcal{O}(1)$ (e.g., Steps 1, 3 and 4). The time complexity of Algorithm 2 is then $\mathcal{O}(n \log n)$.

6 Performance Evaluation

The performance of our proposed routing approach is evaluated and compared to other existing localized protocols, namely Boundhole approach [11] and GAR approach [12], through simulations using CASTALIA simulator [13]. This simulator has the interesting feature of being based on realistic wireless channel and radio models in addition to capturing realistic sensor node behavior.

We have implemented the three protocols and conducted several series of simulations under several settings. Three main comparison metrics have been studied: (a) the length of generated routing paths between communicating pairs of nodes (i.e., hop counts), (b) the average End-to-End delay (i.e., the time elapsed for successfully delivering a data packet) between two communicating nodes and (c) the energy consumption for the entire network.

6.1 Simulation Scenarios and Settings

Our simulation scenarios were constructed as follow: we firstly randomly placed artificial holes in the network area in order to emulate practical obstacles or dead-zones (nodes that have run out of energy). The height of those holes was varied in order to increase the occurrence of minimum local problem situations. Once the holes were placed, we generated several networks with randomly placed nodes within the network area; however, no nodes were placed in the holes. All nodes are considered as having the same communication range. In our scenarios, we have varied this parameter in order to simulate dense networks when the communication range is large and sparse networks when the communication range is low. It has to be noted that small holes other than the artificial ones might occur in regions where node density is low.

For each scenario (i.e., same parameters), we generated 30 networks to repeat the simulation. For each simulation, we selected source nodes as those nodes that are localized on the left side of the network area (e.g., their x coordinates are below a *"Source Limit"*). Similarly, we selected destination nodes as those that are localized on the right side of the network area (e.g., their x coordinates are beyond a *"Destination Limit"*). After this, we generated all communication pairs composed of those source nodes and destination nodes. The objective behind this is to increase the rises in local minimum problem situations as frequently as possible in order to compare the three protocols.

Among all these communicating pairs, we have selected only those that fall into a local minimum problem situation, since the three considered protocols are of greedy nature and hence have the same behavior when there is no local minimum problem situation. This means that in the absence of local minimum problem situations, the three approaches derive the same routing paths.

To measure the average end-to-end packet delivery delay and energy consumption, we have generated for every two communicating pairs a constant bit rate traffic of 256Kb/s using a data packet of 512 Bytes for the duration of 180 seconds. The CASTALIA radio model used is CC2420 [13] with a transmission power level set to -10dBm.

Unless explicitly mentioned in the text, we used the parameters reported into Table 1. It has to be noted that we used the same parameters as those used in [12].

Parameter	Value(s)
Network Area (m^2)	1000 x 800
Number of nodes	500 nodes
Transmission range (m)	50, 100 and 150
Number of holes	3
Holes width (m)	300
Holes height (m)	100, 200, 300, 400 and 500
Source limit (m)	100
Destination limit (m)	900

TABLE 1
Simulation parameters

Figure 13 plots the number of local minimum communicating pairs with regards to the height of the holes. As expected, the number of local minimum pairs increases with the increase of hole height. Similarly, the number of local minimum cases is greater in sparse networks (e.g., communication range equals to 50m) than in dense networks (e.g., communication range equals to 150m for instance). This is due to the fact that in dense networks each node has many neighbors and consequently many routing possibilities. As illustrated in the figure, we also note that in dense networks, small artificial holes (with height equal to 100m) generate very few local minimum problem situations (18 situations over 1978 communicating pairs when the communication range is equal to 100m).

Furthermore, we have observed that not all considered communicating pairs lead to valid routing paths; i.e., some of them could not deliver data because no valid routing path exists (network partition). In these configurations, while GAR and CS could, by construction, detect this network partition (i.e., destination node is not reachable any more) and then inform the source node, Boundhole was not able to detect this

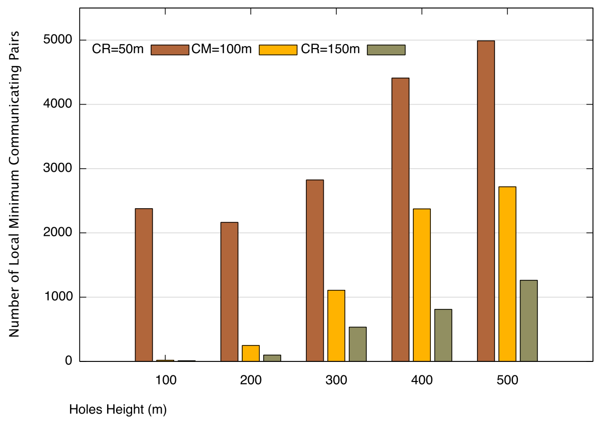

Fig. 13. Local minimum number evolution.

without the use of flooding (in this case, flooding of the entire network is required). This defect of Boundhole was so dramatic that we have excluded those configurations in the remainder of our simulations in order to get a clear view on the behavior of Boundhole without the occurrence of network partitions.

6.2 Simulation Results

We have considered three kinds of network topologies: sparse networks (communication range equals to 50m), medium networks (communication range equals to 100m) and dense networks (communication range equals to 150m).

6.2.1 Sparse Networks

In Figure 14, we plotted the average routing path length generated by the three approaches in the function of hole height when the communication range is set to 50m. As expected and confirmed in the figure, generated routing path lengths increase with the increase in the height of holes and this remains true for the three compared approaches. We also noted that GAR and CS approaches outperform Boundhole approach because we have observed that the latter failed in *intersection situations*, as reported previously, more frequently in sparse networks than in dense networks. Hence this situation leads Boundhole approach to generate longer routing paths. Similarly, CS approach experienced better performance, compared to GAR approach, because the latter visits unnecessary nodes as explained in previous sections. Consequently, we have observed a mean improvement of about 3 hops.

Figures 15 and 16 present the results for average End-to-End delays and average energy consumption respectively. The same observation concerning the impact of hole height remains valid; i.e, performances are better in networks with small holes than in those with big ones. As shown in the figures (e.g., 15 and 16), CS protocol outperforms the two other approaches because it derives better routing paths.

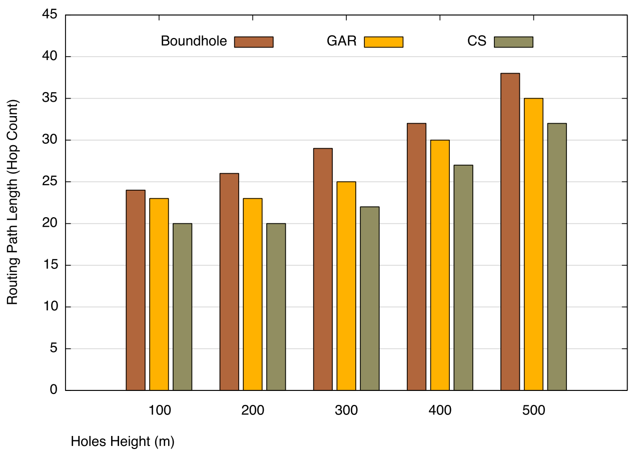

Fig. 14. Routing Paths Lengths.

We have noted however that even GAR protocol behaves on average better than Boundhole; in some particular cases, when intersection situations are not encountered, Boundhole gives better results than GAR because it does not visit unnecessary nodes. But in general its performance decreases dramatically for the rest of the simulation series when at least one such situation is met because of its very costly recovery mechanism.

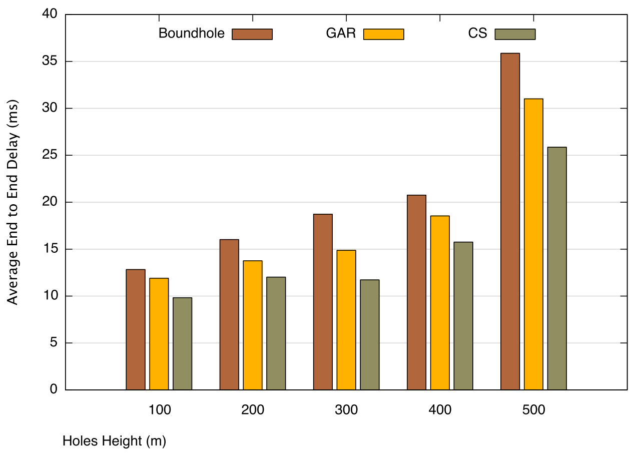

Fig. 15. Average End to End Delays.

6.2.2 *Medium and Dense Networks*

We have conducted the same simulation series, by using two other values for the communication range, 100m and 150m, to better capture different network densities. The results are plotted into Figures 17, 18 and 19. It has to be noted that the lengths of the generated routing paths are, to a certain extent, "inversely proportional" to the communication range. In fact, the greater the communication range is, the smaller the lengths of routing paths are.

Again, we have noticed that CS approach outperforms GAR approach. The mean improvement ranges

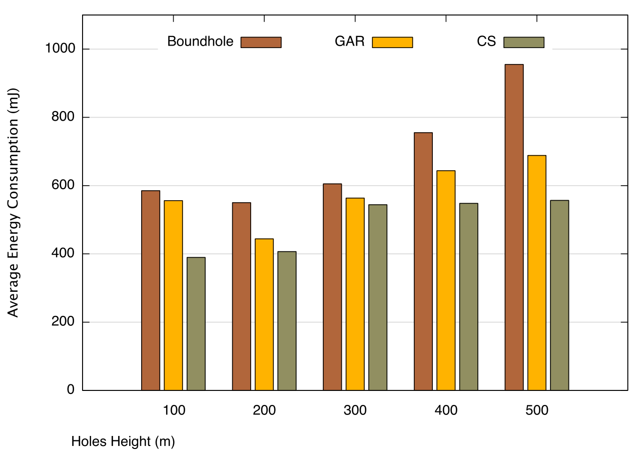

Fig. 16. Average Energy Consumption.

from 1 to 3 hops, depending on the hole heights. We observed that important improvement was reached when the networks are more constrained; i.e., when large holes are present, as reported by the figures.

We also observed that in some cases, Boundhole approach outperforms GAR approach. We then conducted a detailed analysis of the obtained results for these scenarios and we have found that the intersection situations, responsible for the bad performance of Boundhole, are less frequent in dense networks than in sparse ones. That is why figures show a roughly comparable performance between CS approach and Boundhole approach when the hole height is moderate. The situation changes when the holes are important. Furthermore, we have noticed for Boundhole protocol that its recovering cost incurred from intersection situations is very expensive in terms of average End-to-End delays and energy consumption due primarily to the partial flooding technique adopted, as explained in Section 1.

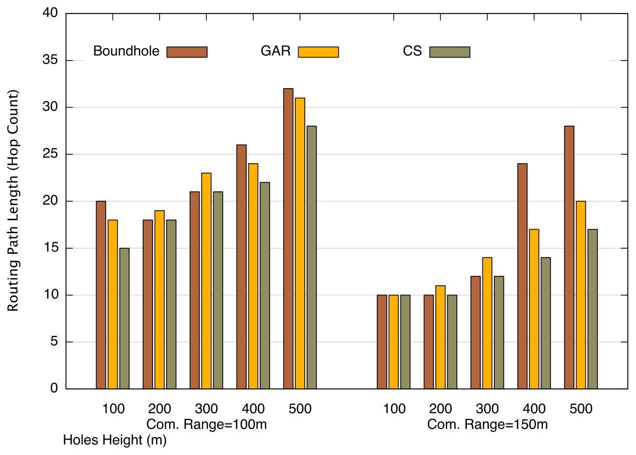

Fig. 17. Routing Paths Lengths.

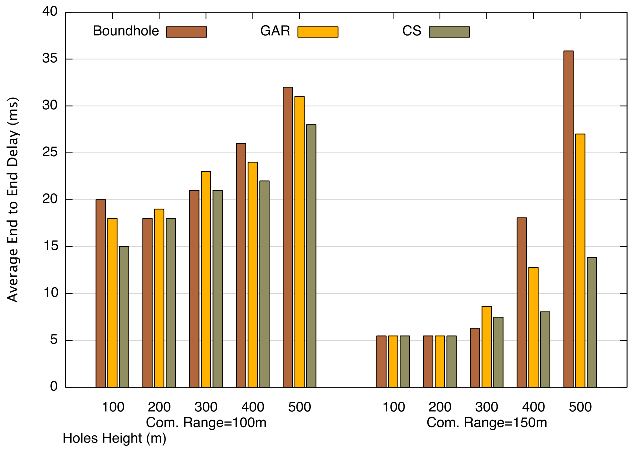

Fig. 18. Average End to End Delays.

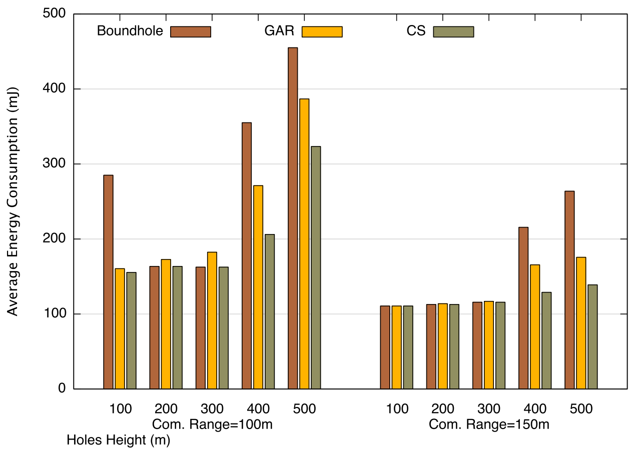

Fig. 19. Average Energy consumption.

7 Related Work

Over the recent years, many geographic routing protocols have been proposed to address the routing hole problem occurring in MANETs [10] in general and in WSNs in particular [14], [15], [16], [17]. Most of these began with GF and recovered from local minimum problems through different strategies. Based on these strategies, previously completed work falls into two categories: (a) graph-based strategies and (b) non graph-based strategies.

Graph-based strategies: in these approaches, such as GPSR [18], GOAFR [19], GOAFR+ [6], Compass Routing II [20], etc., recovery from local minimum scenarios are performed by routing along the boundary of holes, according to a local planar graph. The *right-hand rule* is then adopted which states that when arriving to node N_x from node N_y, the next link traversed is the next link located sequentially counterclockwise and hinged at N_x from link $E(N_x, N_y)$. This rule requires, however, that all links are non-crossing. For this reason, planarization techniques such as Gabriel Graph (GG) [21], the Relative Neighborhood Graph (RNG) [22], etc., are usually used to derive a planar graph from the Unit Disk Graph of the underlying network. A planar graph represents the same connectivity as the original network with non-crossing links. While those techniques have a high success rate, they introduce however an extra-overhead; this is due mainly to the maintenance of a local planar graph at each node [23]. Moreover, though all nodes maintain the planar graph all the time, this information is used only by a sub-set of nodes; i.e., those facing local minimum situations.

Non graph-based strategies: in these approaches, the basic idea is to first localize nodes on hole boundaries and to then derive a detour path so to avoid routing in the direction of holes. In [24], Jia et al. presented HAIR (Hole Avoiding In advance Routing) protocol to bypass holes in advance. HAIR operates as follows: during the first step, nodes recognize themselves as nodes facing holes (i.e., local minimum nodes). They then, during the second step, direct their neighbors to mark them as hole nodes. Routing is then performed on non-hole nodes when possible. This process is then repeated, leading thus to larger hole perimeters. Taking advantage of earlier knowledge about hole positions, HAIR achieves shorter routing paths, and thus reduces energy consumption. However it suffers from a serious drawback: great energy depletion of nodes along the detour paths. Moreover the holes become larger, requiring new detour paths and so on. GEAR (Geographic and Energy Aware Routing) protocol [25] also works in two phases; in phase 1, energy aware next hop selection is performed when routing a packet toward the region of interest; in phase 2 flooding or recursive geographical forwarding is used to disseminate the packet inside the region. In [26], a probabilistic approach, named INF (Intermediate Node Forwarding), is proposed for non Unit Disk Graph networks (i.e., non-uniform radio ranges). To overcome local minimum scenarios, Negative Acknowledgment packets (NAKs) have been proposed to provide feedback to the source node. Based on these NAKs, the sender selects randomly intermediate nodes that do not drop packets. Either flooding or incorporating the NAKs in GF process will increase these protocol overheads and consequently affect negatively their performances.

8 Conclusion

In this paper, we presented a novel geographic routing approach that efficiently resolves the local minimum problem compared to the state-of-the-art approaches. Our approach is localized without retaining memory; i.e., no extra-information other than the localization of nodes is needed and no information is maintained by nodes (contrary to graph-based approaches for instance). This makes our approach scalable and well suited for large distributed WSNs. We provided formal proof of its correctness (free of looping and data delivery guaranteed) and presented an optimized method for its implementation; taking into account the limited capabilities of sensor nodes.

The simulation evaluation and comparison that we have conducted shows clearly the effectiveness of our approach in terms of deriving shorter paths and thus reducing the average end-to-end delays as well as the overall energy consumption of the network in comparison to the state-of-the-art approaches. We have noted that the improvements are more noticeable in constrained networks; i.e., networks with many holes.

In the near future, we plan to extend our work to 3D networks where GF remains very efficient (hop count bounded by d^3, where d is the distance between the source node and the destination node) and still suffers from the local minimum problem.

References

[1] A. Boukerche, *Algorithms and Protocols for Wireless Sensor Networks*, Wiley Series on Parallel and Distributed Computing, October, 2008.

[2] R. Jain, A. Puri, and R. Sengupta, ”Geographical Routing Using Partial Information for Wireless Ad Hoc Networks,” *IEEE Personal Comm. Magazine*, vol. 8, no. 1, pp. 48-57, 2001.

[3] J. Li, J. Jannotti, D. S. J DeCouto, D. Karger, and R. Morris, ”A scalable location service for geographic ad-hoc routing,” *Proc. of 6th Annu. International Conference on Mobile Computing and Networking,*, pp.120-130, 2000.

[4] J. Bahi, A. Makhoul and A. Mostefaoui, ”Localization and Coverage for High Density Sensor Networks,” *Computer Communications,* vol. 31, no. 4, pp. 770-781, 2008.

[5] A. Savvides and M. B. Strivastava, ”Distributed Fine-grained localization in ad-hoc networks”, *IEEE Transactions on Mobile Computing*, 2003.

[6] F. Kuhn, R. Wattenhofer, and A. Zollinger. ”Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing,” *Proc. of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc03),* pp. 267-278, 2003.

[7] A. Boukerche, B. Turgut, N. Aydin, M.Z. Ahmad, L. Bölöni and D. Turgu ”Routing Protocols in Ad Hoc Networks: a Survey,” *Computer Networks,* vol. 55, no. 13, pp 3032-3080, 2011.

[8] N. Ahmed, S.S. Kanhere, S. Jha, ”The Holes Problem in Wireless Sensor Networks: A Survey,” *Mobile Computing and Communications Review*, vol. 9, no. 2, pp 4-18, 2005.

[9] L. Zou, M. Lu, and Z. Xiong, ”A Distributed Algorithm for the Dead End Problem of Location Based Routing in Sensor Networks,” *IEEE Trans. Vehicular Technology,* vol. 54, no. 4, pp. 1509-1522, 2005.

[10] W.J. Liu and K.T. Feng, ”Largest Forwarding Region Routing Protocol for Mobile Ad Hoc Networks,” *Proc. IEEE Global Comm. Conf. (GLOBECOM 06),* pp. 1-5, 2006.

[11] Q. Fang, J. Gao, and L. Guibas, ”Locating and Bypassing Routing Holes in Sensor Networks,” *Proc. 23d AnnualJoint Conf. of the IEEE Comp. and Comm. Soc. (INFOCOM 04),* vol. 4, pp. 2458-2468, 2004.

[12] Wen-Jiunn Liu, Kai-Ten Feng, ”Greedy Routing with Anti-Void Traversal for Wireless Sensor Networks”, *IEEE Trans. on Mobile Computing,* vol. 8, no. 7, pp. 910-922, 2009.

[13] http://castalia.research.nicta.com.au/

[14] N. Arad and Y. Shavitt, ”Minimizing Recovery State in Geographic Ad-Hoc Routing,” *IEEE trans. on Mobile Computing*, vol. 8, no. 2, pp. 203 - 217, 2009.

[15] C. Liu and J. Wu. ”Destination-region-based local minimum aware geometric routing.” *Proc. of the 4th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2007)*, pp. 1-9, 2007.

[16] R. Abe, S. Honiden ”Adaptive geographic routing in wireless sensor networks” *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems (MSWIM 2010)*, pp. 91-100, 2010.

[17] C.-H. Chou, K.-F. Ssu, H. C. Jiau, W.-T. Wang, C. Wang, ”A Dead-End Free Topology Maintenance Protocol for Geographic Forwarding in Wireless Sensor Networks,” *IEEE Transactions on Computers*, vol. 60, no. 11, pp. 1610-1621, 2011.

[18] B. Karp and H.T. Kung, ”GPSR: Greedy Perimeter Stateless Routing for Wireless Networks,” *Proc. of 6th ACM intern. conf. on Mobile computing and networking (MobiCom'00)*, pp. 243-254, 2000.

[19] F. Kuhn, R. Wattenhofer, and A. Zollinger, ”An algorithmic approach to geographic routing in ad hoc and sensor networks,” *IEEE/ACM Transactions on Networking,* vol. 16, no. 1, pp. 51-62, 2008.

[20] E. Kranakis, H. Singh, and J. Urrutia, ”Compass Routing on Geometric Networks,” *Proc. Canadian Conf. Computational Geometry (CCCG 99),* pp. 51-54, 1999.

[21] K.R. Gabriel and R.R. Sokal, ”A New Statistical Approach to Geographic Variation Analysis,” *Systematic Zoology,* vol. 18, no. 3, pp. 259-278, 1969.

[22] G.T. Toussaint, ”The Relative Neighborhood Graph of a Finite Planar Set,” *Pattern Recognition,* vol. 12, no. 4, pp. 261-268, 1980.

[23] Nadeem Ahmed , Salil S. Kanhere , Sanjay Jha, ”The holes problem in wireless sensor networks: a survey”, *ACM SIGMOBILE Mobile Computing and Communications Review*, vol.9, no.2, pp. 4-18, 2005.

[24] W. Jia, T. Wang, G. Wang, M. Guo, ”Hole Avoiding in Advance Routing in Wireless Sensor Networks”, *in Proc. of Wireless Communications and Networking Conference (WCNC)*, pp. 3519 - 3523, 2007.

[25] Y. Yu, D. Estrin, and R. Govindan, ”Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks,” *UCLA Computer Science Department Technical Report*, UCLA-CSD TR-01-0023, May 2001.

[26] Douglas S. J. De Couto and Robert Morris, ”Location proxies and intermediate node forwarding for practical geographic forwarding.” *Technical Report MIT-LCS-TR-824*, MIT Laboratory for Computer Science, June 2001.

Ahmed Mostefaoui is currently an associate professor at the University of Franche Comte, France, since 2000. He received the M.S. and Ph.D. degrees in computer science from Ecole Normale Suprieure de Lyon (France) in 1996 and 2000, respectively. His research interests are in distributed algorithms in wireless ad-hoc and sensor networks emphasizing both practical and theoretical issues, multimedia systems and networking, in particular, distributed architectures.

Mahmoud Melkemi received his PhD in applied mathematics from the University of Grenoble, France in 1992. From 1993 to 2004, he worked as an Associate Professor in the Claude Bernard University, Lyon, France. He joined the Haute Alsace University, Mulhouse, France, in 2005 as a Professor. His main research interests are in the fields of pattern recognition, computer graphics and computational geometry with applications in ad hoc and sensor networks.

Azzedine Boukerche is a Full Professor and holds a Canada Research Chair position in distributed simulation and wireless and mobile networking at the University of Ottawa. He is the Founding Director of PARADISE Research Laboratory at Ottawa U. His current research interests include sensor networks, mobile ad hoc networks, mobile computing, wireless multimedia and distributed computing. Dr. Boukerche has published more than 460 papers in these areas.