New Virtual Coordinate System for Improved Routing Efficiency in Sensor Network

AKSA Karima[1], BENMOHAMMED Mohammed[2] and BILAMI Azeddine[3]

[1] Department of Computer Science, El hadj lakhdar University , Batna, 05000, Algeria

[2] Department of Computer Science , Mentouri University Constantine, 25000, Algeria

[3] Department of Computer Science, El hadj lakhdar University, Batna, 05000, Algeria.

Abstract

Geographic routing has attracted attention since it is a memory-less and scalable approach. Two types of geographic routing protocols are proposed in the literature; those relying on exact position information using GPS, and other algorithms using virtual coordinate system (VCS). In recently proposed VCS, the unique identity cannot be easily assigned to any node due to the amount of reference points needed to produce a unique reference framework. This paper proposes a fundamentally different way of designing a new virtual coordinate system for localization in wireless sensor network which relies only on local connectivity information and per-neighbor communication. Using this new virtual coordinate system, we build a logical topology on which we apply a new fully simple lightweight geographic routing protocol. It is designed to be scalable, loop free, energy efficient and it guarantees delivery with the selection of the shortest path using hops' count.

Keywords: *virtual coordinate system, localization, geographic routing, wireless sensor network.*

1. Introduction

Wireless Sensor Networks (WSN) are foreseen to become ubiquitous in our daily life and they have already been a hot research area. They play an important role in monitoring and collecting data from difficult geographical terrains. They find useful applications in various fields ranging from environmental monitoring to monitoring the patients' conditions in hospitals. Briefly Sensor networks have been identified as being key technology in monitoring and detecting threats.

Routing in wireless sensor networks is a challenging task due to the networks' variable topology and limited resources. Two distinct classes of routing protocols are presented in the literature; the first one is the classical non-geographic routing protocols being either proactive, reactive or hybrid. This class suffers from a huge amount of overhead for route setup and maintenance due to the frequent topology changes; moreover the route discovery or link state updates affect negatively the traffic network and limit its scalability and efficiency. The second class is reserved for geographic routing (called geo-routing) protocols. They are very efficient due to their ability to find new routes in both static and mobile networks with frequent topology changes by using only local topology information. In this kind of protocols, nodes need to know their geographic locations (using some localization mechanisms like GPS), destination location and direct neighbors' locations.

The localization 1 problem is for individual sensors to determine, as closely as possible, their geographic coordinates in the area of deployment. The practical deployment of many wireless sensor networks results in sensors initially unaware of their location. The task of localization is performed immediately after deployment. Further, due to limitations in form factor, cost per unit and energy budget, individual sensors are not expected to be global positioning system (GPS)-enabled. Moreover, in many probable application environments, including those inside buildings, hangars, or warehouses, satellite access is drastically limited [18].

In order to reduce the cost, a subset of the nodes, in the network, can be equipped with a GPS and used to infer the position of the remaining nodes (like in [20, 21]). In such case it suffices to know the distance relative to this selected subset nodes using techniques such as time difference of arrival [24], angle of arrival [23], or signal strength [22]. In theory, if the position of all nodes, in the network, is estimated, any geographic routing algorithm can be used. These approaches introduce, however, drawbacks to the geographic routing. It is possible, for

[1] For a good survey of localization protocols for wireless sensor networks, we refer the reader to the following reference [9]

example, that two nodes obtain the same coordinates leading to delivery failures [12].
Other approaches rely exclusively on the relative distances (or hop counting) to a set of nodes in the network, without the intervention of external location services. The general idea is to define a virtual coordinate system (VCS) and use it to induce a routing protocol based on the virtual coordinates [14].
In general, a VCS overlays virtual coordinates on the nodes in the network based on their network distance from some fixed reference points (called anchors or landmarks); the coordinates are computed via an initialization phase. The virtual coordinates serve in place of the geographic location for purposes of geographic forwarding. Because it does not require precise location information, VCS is not sensitive to localization errors. It is purely algorithmic and very interesting since it reduces dead-ends in critical sparse networks. Furthermore, the virtual distance reflects the real distance: $dv \approx c.dr$ where c: constant, dv: virtual distance, dr: real distance.

Basing on the use of anchors, all geographic routing, proposed in [25], [26], [24], [27], [28], [29], [13], [30], [31], failures of greedy forwarding on VCS are caused by several nodes with the *same identity* occurring in the network: the more the network is dense, the more the appearance of several nodes with the same identity is increasing. Therefore; the principal drawback of VCS is the lack of naming uniqueness. A good routing protocol must set up on a good naming base, which should give each node a *unique identity (unique virtual coordinates)*.
Unlike the vast majority of existing virtual coordinate systems that rely heavily on the existence of a minimum of three anchors without known geographic position, our protocol only requires the use of a sink-node as a starting point to attribute virtual coordinates to sensor nodes in the network. The key idea of our protocol is to allow each sensor to get its position (its virtual coordinates) basing on the use of the cluster method.

In this paper, we propose a method of constructing a virtual coordinate system in wireless sensor networks where location information is not available. It is characterized by its simplicity and its efficiency in assigning virtual coordinates to all sensor nodes by using two principles: the creation of levels in the network (creation of the first coordinate), and the affectation of virtual coordinates to all sensor nodes in each level basing on clustering creation (creation of the second and the third coordinates). A new simple greedy routing algorithm, basing on the use of this new virtual coordinate system, is similarly proposed. It is designed to be scalable, loop free (since it is a greedy routing which always makes any sender node on the path forwarding to a node closer to the destination), energy efficient (since it selects the node which minimizes the energy consumption by maximizing the power transmission towards the destination). It guarantees delivery and finally it selects the shortest path using hops' count.
The remainder of this paper is organized as follows. In section 2, we present an overview about the geo-routing algorithms using exact position information (real coordinates). In Section 3, we have shifted to cover briefly some related works through a quick survey of the geographic routing algorithms that rely on virtual coordinate system. Section 4 introduces, in detail, our new virtual coordinate system named VCSCClockwise. Section 5 is devoted to the presentation of a new greedy routing algorithm based on the use of the virtual coordinates offered by the new VCS presented in the precedent section. Simulation and evaluation of the proposed greedy routing are presented in section 6. Finally, in section 7, a conclusion resumes the essential points and main results reached through this work along with some guided perspectives.

2 Geographic routing algorithms based on real coordinates system

As mentioned above, geographic routing is receiving more and more attention since it is a memory-less and scalable approach. Thus many proposals and protocols have mushroomed in this field almost all over the world. From 1984 to 1986, three progress based geo-routing protocols have been gaining importance, were already proposed for packet radio networks and were lately rediscovered for mobile ad-hoc and sensor networks. These approaches are based on the progress which is defined as the projection of the distance traveled over the last hop from S (source) to any node A onto the line from S to the final destination D. MFR (Most Forward within Radius) proposed by Takagi and Kleinrock [1] was the first geographic routing algorithm and the first protocol using this system and. It was introduced as an attempt trying to minimize the number of hops by selecting the node with the largest progress from its neighbors. The objective of this protocol was to obtain the optimum transmission radius in a contention-based channel. Three months later, a new protocol called RPM (Random Progress Method) was proposed by Nelson and Kleinrock [2]. In fact, it is based on the MFR principle with a slight modification. In RPM packets destined towards D are routed with equal probability towards one intermediate neighboring node that has a positive progress. The rationale for the method is that; if all nodes are sending packets frequently, probability of collision grows with the distance between the nodes, and thus there is a trade-off between the progress and transmission success. Two years later, Hou

and Li [3] have proposed NFP (**N**earest **F**orward **P**rogress) which contributed in minimizing the interference with other nodes and the overall power consumption by transmitting the packet to the nearest node with forward progress. MFR, RPM and NFP, though they are simple localized algorithms, highly efficient in a dense graph and loop-free, are nevertheless, incapable to guarantee delivery, whenever there exist voids in a network.

In 1987 Finn, in [4], had proposed a localized greedy scheme called GF (**G**reedy **F**orwarding) as variant of random progress method, where the node, currently holding the message, will forward it to the neighbor that is closest to destination. Only nodes closer to destination than the current node are considered. This greedy algorithm is greedy forwarding with limited flooding; and therefore; it has the same advantages and insufficiencies as the precedent algorithms.

In 1998 and 1999 was born a type of routing protocols called directional routing protocols. Two of its protocols Dream [5] and LAR [6], appeared in the same year. Dream (**D**istance **R**outing **E**ffect **A**lgorithm for **M**obility) was proposed to use the location information for data delivery in ad hoc network. In this routing protocol the packet is forwarded to all nodes in the direction of the destination. Based on the destination location and its velocity, the source determines an expected zone for the destination and forwards the packet to all nodes within an angle containing the expected zone. If the sender has no neighbors in the direction of the destination, a recovery procedure using partial flooding or flooding is invoked. LAR (**L**ocation **A**ided **R**outing) uses the location information of nodes for route discovery and not for data delivery. It improves the performance of non-geographic ad hoc routing protocols by limiting discovery floods to a geographic area around the destination expected location. One year later, Kranakis, Singh and Urrutia have proposed in [7] DIR (**DIR**ectional) or Compass routing where a node forwards the packet to the neighbor whose edge has the closest slope to the line between that node and the destination; that is the neighbor with the closest direction to the destination. We notice that all these algorithms based directional have very high delivery rates for dense networks, but low for spars graphs. In addition to that the compass routing is not a loop-free. Another local algorithm called Compass Routing II appeared in [8], to mark its contribution in confirming the attainment of the packet to its right destination. Compass Routing II, which becomes known as face routing or perimeter routing, works in planar unit graphs by traversing the faces intersecting the line between the source and the destination consecutively until reaching the destination. This algorithm was considered as the First correct algorithm.

3. Related work

Routing protocols based on virtual coordinate system are proposed in wireless sensor networks without GPS assistance. They rely exclusively on virtual coordinates, derived from either relative distances or hop counting to a set of anchor nodes in the network, without the intervention of external location services. The general idea is to define a virtual coordinate system and use it to induce a routing protocol based on the virtual coordinates [14].

A. Caruso et al. have proposed an algorithm called Glider (**G**radient landmark based routing) [10] where nodes are partitioned into tiles and a set of well dispersed nodes are identified as anchors. Virtual coordinates are then given to each node based on their centered square-distance, otherwise known as the variance, from each anchor. We notice that a good set of anchors greatly impacts the efficiency of GLIDER. However, no theoretical methods are capable of selecting a good set of anchors. In [11] J. Bruck et al. have proposed a routing protocol which uses a medial axis graph (MAG) as a guide for routing without the need for the selection of anchors; this protocol is named MAP (**M**edial **A**xis **P**rotocol). In GLIDER and MAP, each node is required to memorize a graph having the global topological features, which is demanding in terms of message communication and memory overhead. In LTP [12], the authors introduce a new coordinate system, based on a tree construction. Each node is assigned a label which embeds the path between this node to any other node in the network, based on the path in the tree which is unique (see figure 1). LTP ensures the delivery of the message and the success of the routing but is not energy aware and may provide paths which are much longer than the optimal one [15].

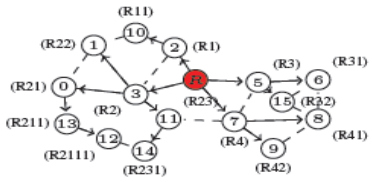


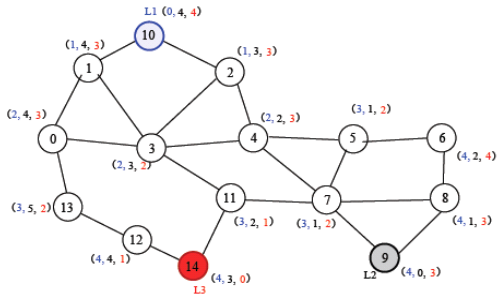
Fig.1 Illustration of labeling in LTP [15]

GEM [33] proposed routing based on a virtual coordinate system. A virtual polar coordinate space (VPCS) is used for localizing each node in the network. A tree style overlay is then used for routing. Thus, GEM is not stateless. Further, GEM works only as a localization algorithm, generally does not provide guaranteed uniqueness of node identity based on coordinates. Rodrigo et al proposed beacon vector routing (BVR) [31], which forms a VCS with a large number of anchors (typically 10 to 80). BVR uses Manhattan-style distance to measure

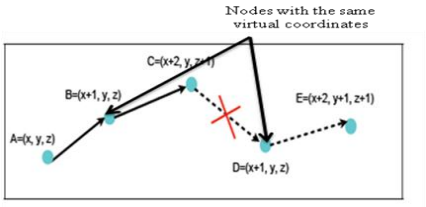
distance between two given coordinate points. BVR uses such a large number of anchors to increase the possibility of BVR routing success in the greedy mode. However, even with so many anchors, BVR fails frequently to forward packets back to the reference node closest to the destination. VCap (**V**irtual **C**oordinate **A**ssignment **P**rotocol) [13] proposed by A. Caruso et al. is a system of virtual coordinates based on hop counts from three anchors. Nodes are assigned a triplet of coordinates given as the number of hops the node is distant from each anchor [13]. Then, nodes use a greedy routing, like *MFR* with the Hamming distance computed on these coordinates (instead of the Euclidean distance). Note that even though, an efficient coordinate system can be generated using three anchors, a more accurate one can be established as the number of anchors increases as claimed in [14] [15]. With hop counting, the identifiers or virtual coordinates obtained are not unique while the routing algorithms rely on the uniqueness of the labels. Several similar protocols to VCap are also proposed [14], [25], [26], [24], [27], [28], [29], [30], [31] with the same inconvenience: several nodes have the same virtual coordinates (the same identity).

With the following practical example (see figure 2), we can understand clearly the drawback that hinders the construction of an efficient virtual coordinate system used both in Vcap and several similar protocols:

Three sensor-nodes, in the sensor network, are distinguished as anchors (landmarks) (see figure 2(a)): L1 for node 10, L2 for node 9 and L3 for node 14. Each anchor broadcasts a beacon in the network incremented at each hop. From it, an arbitrary node x knows its virtual coordinate vector V (x) = (h_1, h_2, h_3) where h_i is the hop-distance between x and Li. Thus, every node has a 3-dimensional vector as coordinates constituted by the number of hops between itself and every anchor. For instance, node 0 can reach L1 (node 10) in 2 hops, L2 (node 9) in 4 hops and L3 (node 14) in 3 hops. Its virtual coordinate is thus V (0) = (2, 4, 3).



(a)



(b)

Fig.2 (a) Example of a virtual coordinate system used in VCap, (b) Illustration of failed routing due to the presence of nodes with the same virtual coordinates in the network [14].

Obviously, using only these coordinates does not guarantee delivery since the node coordinates are not and thus do not identify a single node [15] (see figure 2 (b)). This is for example the case for nodes 5 and 7 on which are both labeled with (3, 1, 2).

A consequent observation is that a VCS constructed only by the network distance (number of hops) to anchors cannot provide naming uniqueness for general graphs [26].

4. A new virtual coordinate system "VCSCClockwise" based on clustering method

4.1 The selected architecture to apply VCSCClockwise

Because of the scarce energy supply of a wireless sensor network, the task of choosing network architecture becomes a prerequisite to optimize the energy consumption. Since, in a multi-hop network, a sensor spends most of its energy in relaying data packet. It is important to shorten the distance spent by the packet until reaching the sink. These distances can be reduced seriously by fixing a single sink in the center of the sensor field as it is illustrated in figure 3. Due to the efficiency of this architecture, several researchers have selected them like [17].

For this most reason (optimizing the energy consumption), we have chosen this kind of architecture to apply VCSCClockwise. This latter will use a sink as a starting point to attribute virtual coordinates to sensor nodes.

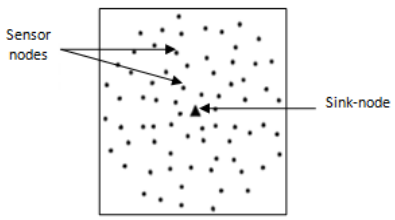


Fig.3 Architecture of sensor network with a central sink

4.2 Building the first coordinate

To build the first coordinate, we will focus on hop's count to create levels as follows:
- Attribute to the sink-node level zero (*Level=0*).
- After one hop, from the sink, all sink-node's neighbors (the set of nodes in communication range of the sink-node) will have level equal to one (*Level=1*).
- After two hops, from the sink, the level will be incremented to get a new one equal to two (level=2). Assign this level's number to all neighbors of the precedent nodes in level one.
- The same operation will be repeated until all nodes in the network will be identified by the hops count representing their levels as it is illustrated in figure 4.

We can summarize the building of this first coordinate in the following algorithm:

Building the first coordinate –Algorithm

```
Let L=0  {/*L: Level*/}
Let U the current node
For each sensor node Do
  If  U=sink Then
     U(First coordinate) ←L {/* Level=0 */}
      Send L to sink's neighbors {/* next hop*/}
  Else
     Receive L from sender
     If U has no level Then
        L ←L+1
        U(First coordinate) ← L
        Send L to neighbors of the current node U {/* next hop*/}
    End If
  End If
End For
```

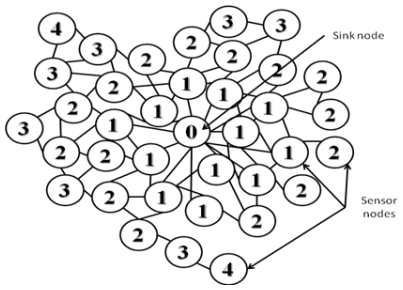


Fig. 4 Creating levels (first coordinate) using hops count from the sink-node.

4.3 Building the second & the third coordinates

To build the second and the third coordinate, we will focus on two basic notions: the *cluster's number* and the *position of the node in the cluster*. The *cluster's number* will represent the second coordinate, and the *position of the node in the cluster* will represent the third coordinate. The third coordinate can be either a *head-cluster* or a *simple sensor-node* in the cluster.

In our virtual coordinate system, **if** a node **U** with the third coordinate equals to **1**, it means that **U** is a head-cluster; **else U** is a simple node in the cluster.

Example: U(first coordinate, second coordinate, third coordinate) :

U(1,2,**1**) or U(3,4,**1**) or U(1,1,**1**) → U is a head-cluster.

U(1,2,**2**) or U(2,4,**4**) or U(1,1,**5**) →U is a simple node in the cluster.

The following steps will show in detail the creation of the second and the third virtual coordinates:

*-Level **0** (case of the Sink node):*

In this level, we will find two operations:

Add to the first coordinate of the sink-node two other coordinates equal to zero representing their second and third coordinates. So, the sink-node will have *three* virtual coordinates which are **(0,0,0):**

The first **0** indicates the first coordinate (Level=0 as it is explained previously),

The second **0** indicates the number of the cluster,

The third **0** indicates its position in its group.

The first cluster, in the network, will be constructed in the first level (level=1). To achieve this operation, the sink-node will select the farthest node situated in level one (level=1) to which it will assign the two coordinates (1,1) (as it is illustrated in figure 5); the first "**1"** is the cluster's number (the first cluster in the level 1 and, entirely, in the network) and the second "**1"** indicates the head-cluster.

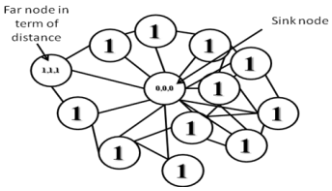


Fig. 5 The first head-cluster in the sensor network.

*- Level **1**(all sensor nodes which have the first coordinate **" 1")**:*

In this level, four operations can be executed: (see figure 6)

The farthest sensor-node selected by the sink-node will receive the two coordinates (1,1). This farthest node will add these two new coordinates to the first already existing one (level number) to get the final coordinates: (1,1,1). The way of reading the set of the virtual coordinates is from the right to the left: **First** node (head-cluster) of the **First** cluster in the **First** level.

This head-cluster, with (1,1,1) coordinates, will:

- Order all its neighbors (N neighbors) according to their distances from the nearest to the farthest (neighbors which are in its range, and in the same level).
- Attribute a number to each neighbor from 2 to N-1 (number 2, number 3, … number N-1), beginning by

the nearest to the farthest. Each number indicates the position of the single node in the cluster (third coordinate). So, the nearest will get "2", the following will get "3", … until the before last which will get "N-1".

- Send two coordinates to each neighbor, which are: "1" (cluster's number), and the position of the node calculated previously. Each neighbor, in the same level, will receive the two additional coordinates to get as following:

The nearest node will have (1,1,2): the **Second** node of the **First** cluster in the **First** level

The following nearest node will have (1,1,3): the **Third** node of the **First** cluster in the **First** level

The N-1 neighbor node will have (1,1,N-1): the **N-1** node of the **First** cluster in the **First** level

- Send the two coordinates (1,1) to the farthest node in the next level (level 2) (repeating the same operation (b) of the precedent level (level 0)). So the first node in level 2 will receive the two additional coordinates to get as virtual coordinates: (2,1,1); **First** node (head-cluster) of the **First** cluster in the **Second** level.

The neighbor **N** (the farthest neighbor according to the node (1,1,1)) will be a head-cluster of the second cluster in the same level (level 1). So the node (1,1,1) will send to this neighbor two coordinates (2,1): "1" for the head-cluster, and "2" for the second cluster. This farthest neighbor will add these two received coordinates to its first one to get as virtual coordinates (1,2,1): the **First** node (head-cluster) of the **Second** cluster in the **First** level. This node, with (1,2,1) coordinates will repeat the same operations fulfilled by the previous head-cluster with its neighbors except for the last step of the operation (b)(this operation concerns only the head-cluster of the first cluster in each level). The same operations will be repeated with each new head-cluster until all sensor-nodes within their clusters in the same level are identified by the three virtual coordinates.

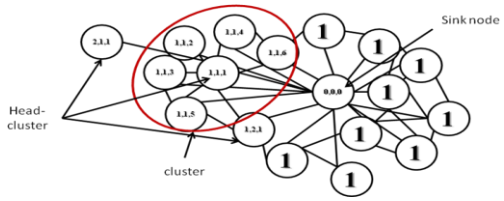


Fig. 6 Creation of the first cluster in the first level.

In level **2,** the same operations will be repeated as in level **1,** to get all nodes in the level **2** with 3 virtual coordinates as it is illustrated in figure 7. The same operations will be repeated in each level until the last one in the network.

Remark

As mentioned before, only the sink-node and the head-cluster of the first cluster which will select the following first head-cluster in the following level. This operation is executed only one and one time.

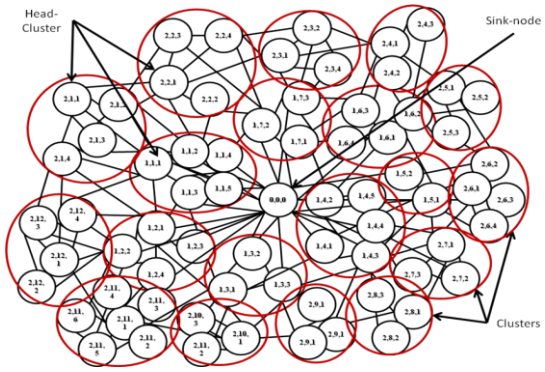


Fig. 7 Cluster-based topology

We can summarize the building of the second and the third virtual coordinates in the following algorithm:

Building the second & the third virtual coordinates -Algorithm

Let U the current node, **L** the current level
Let N neighbors number of **U** in the **same level** (L), **Let M** neighbors number of **U** in the **next level** (L+1)
Let A array with length=N containing **N** neighbors, of level L, ordered from the nearest to the farthest from **U** *{/A[1],A[2],……A[N-1],A[N]) */}*
Let B array with length=M containing **M** neighbors, of level L+1, ordered from the nearest to the farthest from **U** *{/B[1],B[2],……B[M-1],B[M]) */}*
Let F: first coordinate *{/* get it from the precedent algorithm*/}*, **S:** second coordinate, **T:** third coordinate, **VC**: **V**irtual **C**oordinates.
For each sensor node **Do**
  If U=sink **Then**
    S←0
    T←0
    U(VC)← (**F**,**S**,**T**) *{/* add the second and the third coordinate to the first one, (F,S,T)=(0,0,0) */}*
    Send (1,1) to B[M] *{/* the farthest neighbor in the next level*/}*
  Else
    Receiving (S,T)
    U(VC)← (**F**,**S**,**T**) *{/* add the second and the third coordinate to the first one*/}*
    If T=1 Then
      Send (S+1,1) to A[N] *{/*Farthest neighbor in the current level that will be the head-cluster of the next cluster*/}*
      If S=1 Then *{/* operation executed only by the head-cluster of the first cluster in each level*/}*
        Send (1,1) to B[M] {/* *Farthest neighbor in the next level*/}*
      End If
      For i ←1 **to** N-1 **Do** *{/* creating cluster*/}*
        Send to A[i] the coordinates (**S**,**i+1**) *{/* i+1=T which is the third coordinate */}*
      End For
    End If
  End If
End For

We can discover an important point in this section (see figure 8): it is the direction induced from assigning virtual coordinates to sensor nodes in each level, where: each level has its own direction either counterclockwise or clockwise. It is the task of the head cluster in each level to decide whether the direction of the level will be clockwise or counterclockwise depending on the position of the farthest node in its cluster (depending on the following head-cluster). The nomination of our proposed virtual coordinate system comes from this interesting remark.

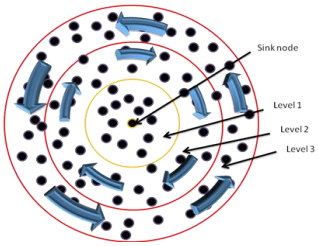


Fig. 8 The directions of creating clusters.

4.4 Discussion

1- Using VCSCClockwise, all nodes in the sensor network are identified with a unique identity (ID); it is impossible to get two nodes with the same virtual coordinates: the first coordinate classifies the nodes into levels (several nodes are similar in the same level), the second one classifies them into cluster (sensor-nodes are similar in the same cluster, they have the same first and second coordinates), the last one produces a unique reference in the cluster and entirely in the sensor network. So VCSCClockwise provides a universal unique identity to any node in the sensor network.

2- All VCS based anchors, mentioned previously, use at least three virtual coordinates, where the number of virtual coordinates has a relation with the number of anchors: the more the number of anchors is increasing, the more the number of the virtual coordinates is increasing and vice versa, for example, if we use 10 anchors as reference points in the network, each sensor-node will be identified by 10 virtual coordinates. However, using VCSCClockwise, each sensor-node stores no more than three virtual coordinates whatever the number of sensor nodes deployed in the area of interest.

3- The major problem and difficulty when using anchors resides in how to select those anchor nodes. VCSCClockwise is free from anchors' constraints; we do not need to use them entirely. Each node builds a qualitative distance according to the physical number-hop neighborhood information (i.e. topological information).

4- The operations of assigning the two last virtual coordinates to the sensor-nodes starts in a parallel way in all levels as it is illustrated in the following figure (figure 9).

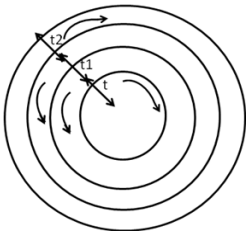
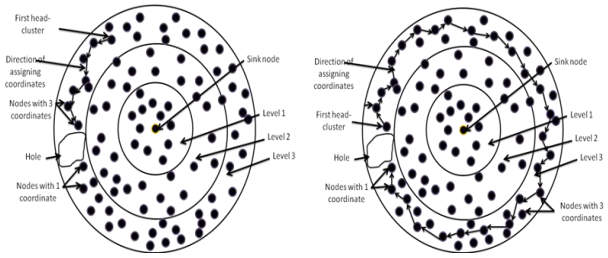


Fig. 9 Assigning virtual coordinates in a parallel way: t≈a.t1≈b.t2; a, b: constants, t, t1, t2: times of the beginning of assigning virtual coordinates in each level.

5- VCSCClocwise benefits from the denser network because it is based on connectivity to create clusters in levels. In a homogeneous and connected network (dense or not), VCSCClockwise succeeds 100%. But in a network with a hole in the same level its success is lower. In fact, its success depends on the selection of the first head-cluster and its direction to assign virtual coordinates in the same level. We must have at least one path from the head of the first cluster to the head of the last one constructing a circle (see figure 10(b)). Otherwise, this new virtual coordinate system stops working as it is illustrated in figure 10(a). In this case, some nodes can get only one coordinate (level's number).



(a) VCSCClockwise failed. (b) VCSCClockwise succeeds.

Fig. 10 Sensor network with a hole in the same level.

6- On the negative side of all VCS based anchors, mentioned previously, may be sensitive to collisions in the initialization phase that requires a flood from each

reference point (anchor) [26]. However, our contribution is a collision-free labeling algorithm; this leads to a non flooding operation.

7- VCSCClockwise is a position-based clustering algorithm for WSN where the clustering is determined after the deployment of the network, which is completely different from PANEL[33], the one which seems similar to VCSCClockwise . However, PANEL assumes that the sensor nodes are deployed in a bounded area partitioned into geographical clusters (unequal size clusters) from the smaller size to the bigger ones. The clustering is determined before the deployment of the network, and each sensor node is pre-loaded with the geographical information of the cluster to which it belongs. At the beginning of each epoch, a reference point is computed in each cluster by the nodes in a completely distributed manner depending on the epoch number. Once the reference point is computed, the nodes in the cluster elect the node that is the closest to the reference point as the aggregator (cluster-head) for the given epoch. The reference points of the clusters are re-computed and the aggregator election procedure is re-executed in each epoch. This ensures load balancing in the sense that each node of the cluster can become aggregator with nearly equal probability.

8- VCSCClockwise can be leveraged by a number of greedy routing protocols mentioned previously like the Greedy Perimeter Stateless Routing (GPSR) protocol [34].

5. Declivity Algorithm based on VCSCClockwise

Using VCSCClockwise, the sensor network can function properly due to the virtual position information provided to the sensor-nodes. Since routing protocols must be simple, easy to understand and implement, and have good average-case performance, we have proposed a very simple greedy routing algorithm named Declivity based on VCSCClockwise virtual system. The name of this latter comes from the selection of the direct path from the source to the sink-node destination by declining from the source's level to zero level (level of the sink node).

5.1 Assumptions and Notations

Let: - U the current node;
- L(U): level of the node U (its first virtual coordinate);
- N(U): the set of physical neighbors of the node U, (i.e. the set of nodes in communication range of the node U);
-A(U)ϵN(U): the set of physical neighbors of the node U which are in level just less than L(U); (L(A(U))<L(U)).
-B(U)ϵN(U): the set of physical neighbors of the node U which are in level equal to L(U); (L(B(U))=L(U)).
-C(U)ϵN(U): the set of physical neighbors of the node U which are in level just more than L(U); (L(C(U))>L(U)).
- DE(U)ϵN(U): the set of physical neighbors of the node U which are in dead-end case.

5.2 Algorithm Description

Two of the objectives that have been most commonly considered in the literature are minimizing the maximum transmission range at each node (assuming all nodes use a common transmission radio), and minimizing the total power incurred by all the nodes [17].
Using Declivity, each sensor-node, in the network, keeps its minimum transmission power (a short range radio for communication.). Moreover, Declivity selects the shortest path in terms of hops' count.
In a homogeneous network (dense or not), this proposed greedy forwarding works easily as follows: the current node with relative virtual coordinates (F,S,T) forwards a packet to its nearest neighbor toward the destination. This selected neighbor exists in the list A which contains all neighbors found in a level just less than the current level (Select A[1] with relative virtual coordinates (F-1,S',T'))
The following simple algorithm describes Declivity in a homogeneous network.

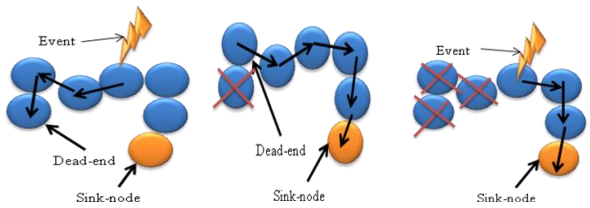
Declivity–in a homogeneous network

For each sensor node **Do**
1. **If** next-hop= (0,0,0) **Then**
2. **Exit** {/* Routing has succeeded*/}
3. **Else**
4. **Select** A[1] {/* Select the nearest neighbor in level just less than the level of the current node */}
5. **End If**

In case of holes, dead-ends or obstacles we follow these steps:

- The neighbors (**N**) of the current node **U** with level **L** are divided into three lists: The first one (**A)** is reserved for neighbors situated in a level just less than level **L (Level=L-1)**. The second one (**B**) will be attributed to those in the same level of **L**, and the third one (**C**) will be assigned to neighbors in level just more than **L (Level=L+1)**. All neighbor-nodes in each list are ordered from the nearest to the farthest in terms of distance.

- If A is not empty ($A \neq \phi$), the current node U will select the nearest neighbor from this list (select A [1]).
- In case of hole, if A is empty while B is not, U will select the nearest neighbor from the list B (select B [1]).
- If both A and B are empty, U, which is a dead-end node in this case, will select the nearest neighbor from the list C. Note that this selected neighbor must be different from the sender, but if the list C contains only one node, it is inevitable that U will select it. All nodes in communication range of the node U will put it in their lists of dead-end nodes (DE) to avoid selecting it in any future selection (see figure 11).



(a) (b) (c)

Fig. 11 The delete of dead-end nodes to choose a shortest path

With the use of Declivity, there will be some kind of flexibility to choose the next hop towards the sink-node, where we can go further until reaching the destination. In case of holes, we can go either left or right, or even, in the worst cases, we can return backward to choose another path.

In general, the following algorithm describes Declivity in three types of networks: homogeneous network, a network with holes (including case of dead-ends) and a network with obstacles.

Declivity -Algorithm

For each sensor node **Do**
1. **If** next-hop= (0,0,0) **Then**
2. **Exit** {/* Routing has succeeded*/}
3. **Else**
4. **If** ($A \neq \phi$) **Then**
5. **Select** A[1]
6. **Else** {/* case of Hole */}
7. **If** ($B \neq \phi$)Then
8. **Select** B[1] {/* B[1]≠ the sender of node U otherwise select B[i] ≠ the sender of node U */}
9. **If** |B|=1 then{/* case of Dead-End */}
10. **Send** to all neighbors of the current node (the set of nodes in communication range of U) the following message: ***U is a Dead-End*** {/*each neighbor will **add** U to its list **DE***/}
11. **End If**
12. **Else** {/* case of Dead-End */}
13. **Select** C[1] {/* C[1]≠ the sender of node U otherwise select C[i] ≠ the sender of node U */}
14. **Send** the following message to all neighbors of U: ***U is a Dead-End*** {/* each neighbor will **add** U to its list **DE** */}
15. **End If**
16. **End If**
17. **End If**

5.3 Discussion

1- Declivity routing protocol is described as being loop free. For example, let U the current node and W the selected neighbor's node for the next hop. The distance between U and the sink-node using hops count is equal to its level (d(U,Sink)=level(U)), and the distance between W and the sink-node using hops count is equal to its level (d(W,Sink)=level(W)). Thus W < U in terms of distance if and only if d (W, sink) < d (U, sink). Let us assume that node U0 is the source of the packet, the sink-node is its destination and the node U1 is the next hop chosen by the node U0. In a homogeneous network, it is impossible to get a list A(U0) empty, $A \neq \varnothing$, so, U1 will be an element from the list A and d (U1, sink) < d (U0, sink). Our routing process therefore strictly reduces distances to destination; this means that loops cannot be created. In case of network with holes U1 will be an element from a list A(U0) if this latter is not empty else U1either will be an element from a list B or a list C (if U1 ϵ C, so U0 is a dead-end). To avoid a loop in case of holes we must delete dead-ends nodes and avoid choosing a sender of U0, so: U1 must not be the sender of U0 unless if it is the unique neighbor of this latter. In this case, U0 is a dead-end which will be directly deleted from lists of its neighbors and will be avoided in future selections.

2- In Declivity, there always exists a next hop that is closer to the destination. If a network is homogeneous, automatically a node in the list A is chosen as the next hop, this ensures a progress on the A coordinates. Else, in the presence of holes or obstacles the next hop will be chosen in B or C lists, this ensures a progress on the B or C coordinates towards the destination.

3- Declivity guarantees delivery in a homogeneous network. Each node has a unique set of virtual coordinates. This ensures that the next hop of a packet is unique.

4- Using Declivity in a homogenous network, the shortest path has a length in hops count equal to the level's

number (ex: if level number=5 then shortest path=5hops) : the hops' number is equal to the source's level.

5- Two weaknesses can appear when using Declivity: 1- In a network with a hole, if randomly we have two paths one short and the other long, declivity can select the longer one depending on the selection of the nearest nodes to the source (see figure 18 in ssection 6.2.1). 2- If a sensor node is in case of dead-end node having all its neighbors in the same case. When this dead-end node detects an event, it can't root the packet towards the sink-node (as it is illustrated in figure 12) even if it exists a path from the dead-end node to the sink.

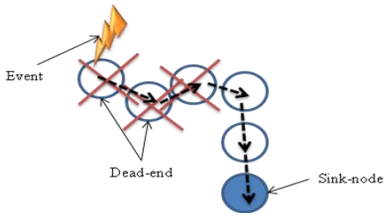


Fig. 12 Declivity fails to root the packet to the sink node in case of dead-end nodes

6. Simulation & results

We have simulated our work using VisualSense1.0.7 [18]: Visual Modeling for Wireless and Sensor Network Systems. This simulator is an open source, it is a project that is supported by the National Science Foundation (NSF award number CCR-00225610), and Chess (the Center for Hybrid and Embedded Software Systems).

6.1 Simulation setup norms

The simulated network can be described as follows:

- Sensor-nodes are static, deployed in a surface of 1250 × 1250 square, they are uniformly distributed over the area;
- All nodes have the same maximum transmission range Rmax= 200;
- The single fixed sink-node is placed in the center of the area with R=200;
- Applying the VCSCClockwise by simulation, The maximum level in our simulation is 4(5 levels from 0 to 4);
- Obstacles' characteristics: to test Declivity more effectively, VisualSense offers the possibility of using obstacles in the simulation as follows: if the attenuation depth of the obstacles = 0, it means that there is no communication between the neighbors that are separated by these obstacles. On the other hand, if the attenuation depth of the obstacles > 0, it means that there is a communication (connection) between the neighbors.

Remark : *in our simulation, the use of obstacles comes after applying VCSCClockwise (after assigning virtual coordinates to sensor-nodes).*

In our simulation, we have tested our protocol on three types of networks: in a homogeneous network (figure 13(a)), in a network with obstacles (figure 13(b)), in a network with holes (figure 13(c) with two holes, figure 13(d) with one hole).

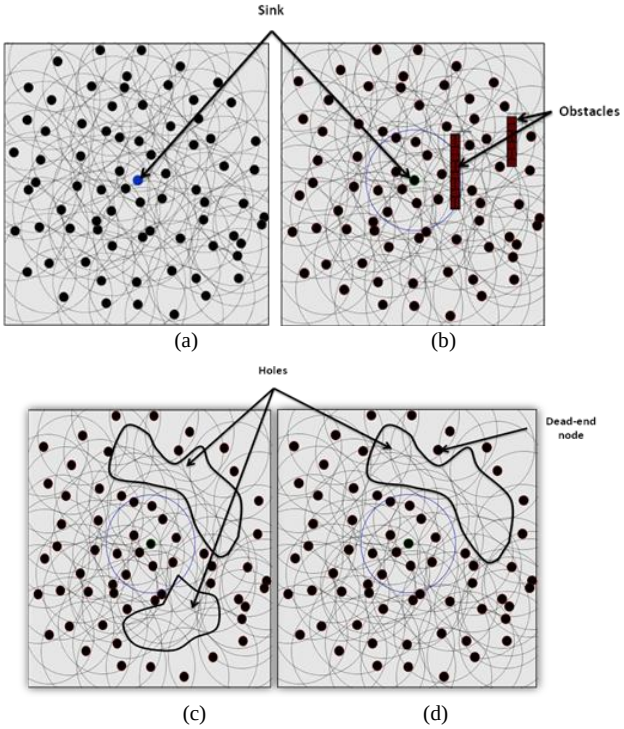


Fig. 13 The types of networks used in our simulation: (a) homogenous, (b) with obstacles, (c) with two holes and (d) with one hole.

6.2 Results

6.2.1 Evaluation of Declivity in three types of network

a. Homogeneous network

In a homogeneous network (either dense or not), Declivity always succeeds to choose the shortest path using the number of hops. Figure 14 illustrates the success of Declivity in routing where the source is situated in different levels. The paths found represent the unique shortest paths existing in the network in terms of hops count and energy consumption.

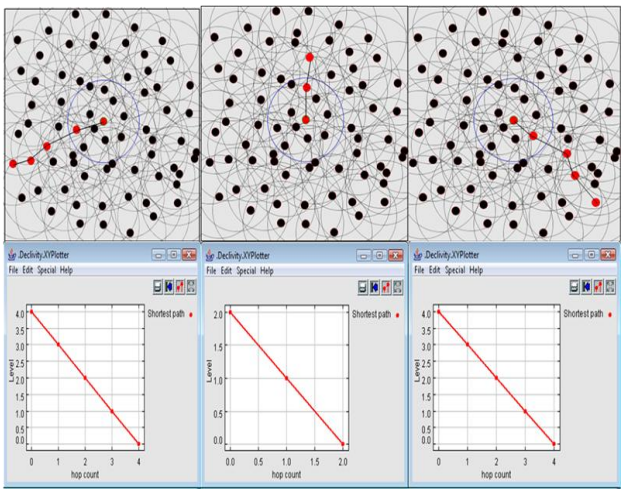


Fig.14. The routing in homogenous network with different sources' position

To compare Declivity to other protocols (see figure 15), we have chosen DIR from real coordinates system, Vcap and Vcost from virtual coordinate system.

We have implemented DIR protocol using VisualSense simulator. Each sensor node will have two coordinates (x,y) as real coordinates given by the landmark of VisualSense. The sink node will have as coordinates (0,0) representing the center of the network.

In VCap and VCost, the sink-node will have a set of virtual coordinates as any sensor-node, and because it is a final destination, it will have a distance equal to 0 applying Hamming distance as it is detailed in [3] and [5].

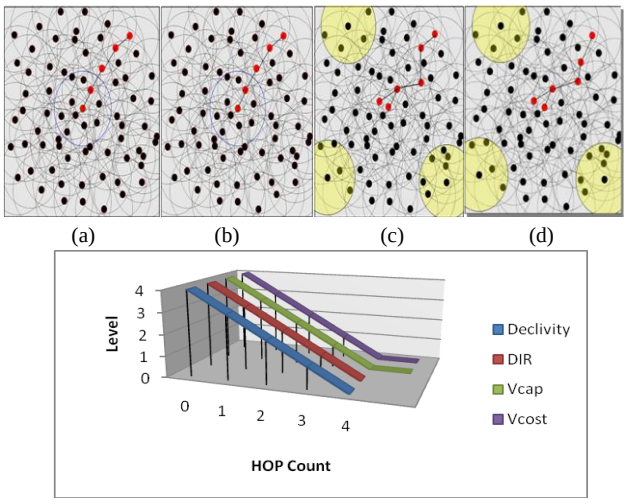


Fig. 15 A path to follow from the specified source to the sink using the following protocols: (a): DIR, (b) Declivity, (c) VCap and (d) VCost

In figure 15 (a) and (b), the length of the path followed by Dir and Declivity using hop count is the same. The protocol VCost is considered in [6] as the first energy efficient routing protocol that uses virtual coordinate system, but it lacks guaranteed delivery (as it is mentioned in section 3) where several sensor nodes can get the same virtual coordinates and this will lead to get several destinations instead of one as it is illustrated in figure 15 (c,d).

b. Network with holes

In figure 16, the sensor network contains one hole, with two different sources used in routing. In (a) the source will choose the nearest neighbor in the same level 4 because no node exists in the level 3, this latter will select one neighbor in the next level (level 3) until reaching the sink node (level 0) with a path length equal to 5 hops . In (b) we find a dead-end case, where Declivity succeeds to guarantee delivery with a path length equal to 7 hops.

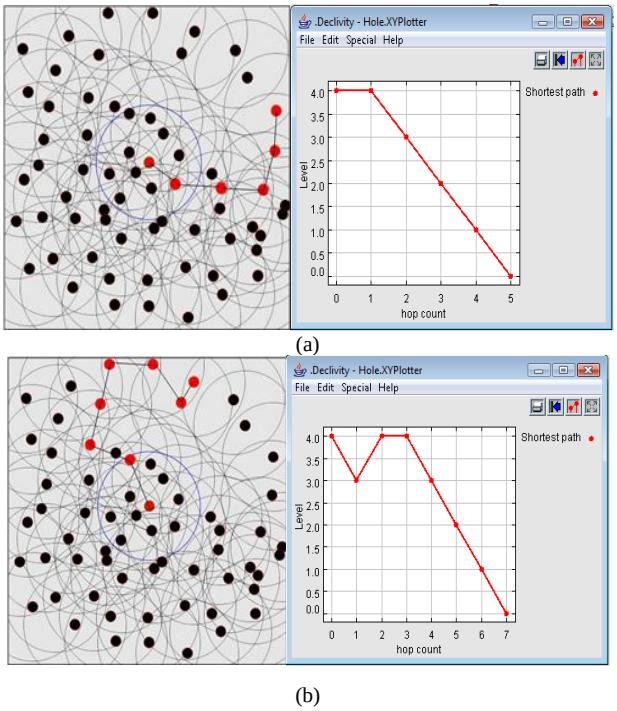
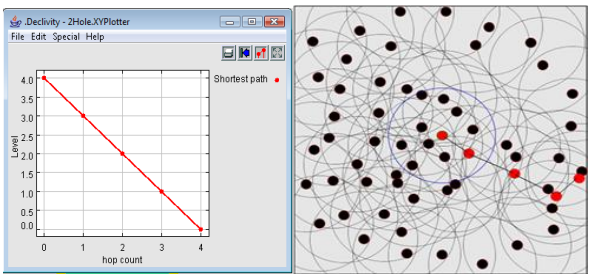
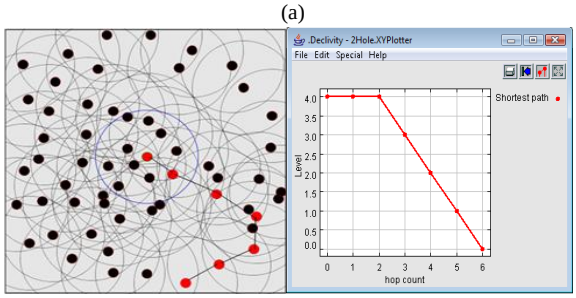


Fig. 16 The network with one hole :(a) shortest path has 5 hops, (b) presence of dead-end shortest path has 7 hops.

In figure 17, a hole is added to the precedent network to get two holes. In (a) and (b) the sources are different but the nodes selected for routing between the two holes are nearly the same.



(a)



(b)

Fig. 17 The network with two holes

In a network with a hole, if randomly we have two paths one short and the other long, declivity can select the longer one depending on the selection of the nearest nodes to the source (see figure 18).

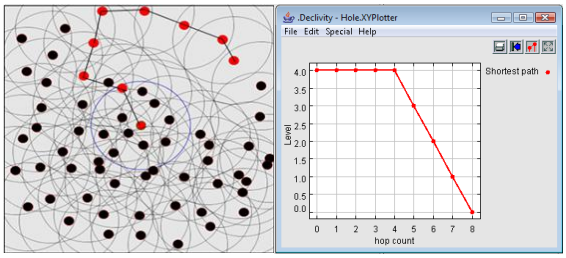
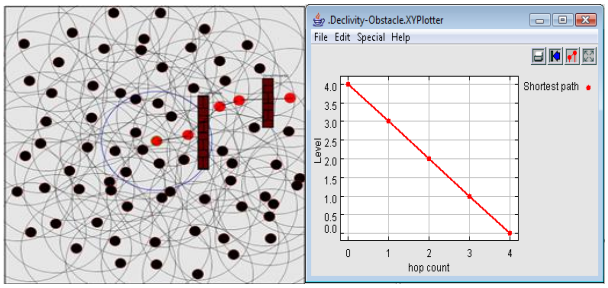


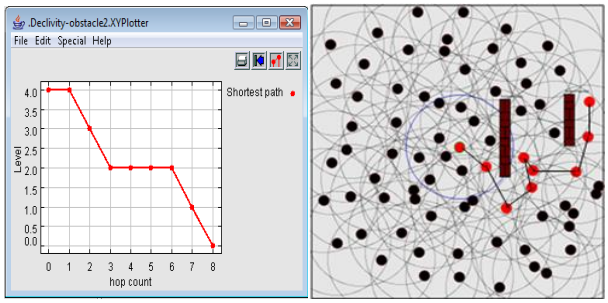
Fig. 18 Problem of a long path chosen by Declivity

c. Network with obstacles

In the following simulation, we have got two cases related to the attenuation depth. The figure 19(a) illustrates the case of an attenuation depth equal to 30 where the sensor-node can communicate with its neighbors that are situated behind the obstacle.



(a)



(b)

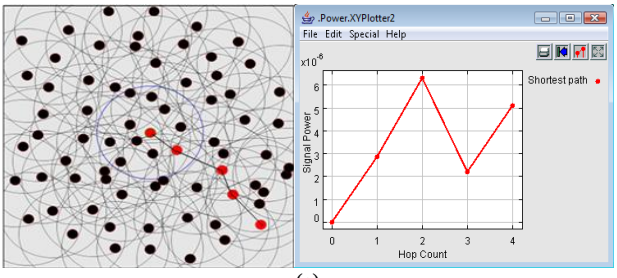
Fig. 19 Network with two obstacles: (a) attenuation depth is equal to 30 units, (b) attenuation depth is equal to zero

The figure 19(b) illustrates the case of attenuation depth equal to 0, where there is no communication between two neighbors separated by an obstacle. So, the node in front of the obstacle is obliged to select another neighbor free from obstacles until reaching the sink.

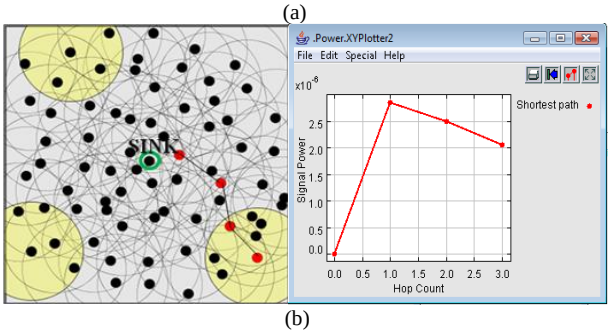
6.2.2 Declivity vs Vcost in power consumption

We have made a comparison concerning power consumption (transmission power) between Declivity and VCost in a homogeneous network and in a network with obstacles.

In figure 20(a), Declivity guarantees delivery with *four* hops, while, in (b), VCost failes to guarantee delivery and is stopped at *three* hops. However, the transmitting power of the path in Declivity is more than VCost (the energy spent using Declivity is less than VCost) which stops just in the third one.



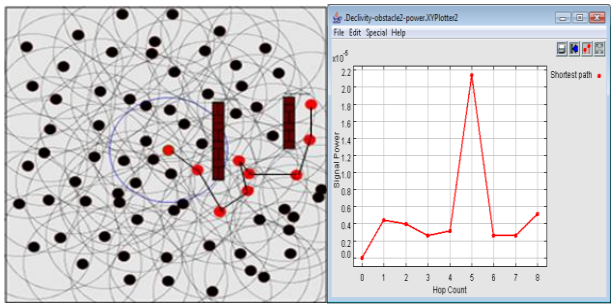
(a)



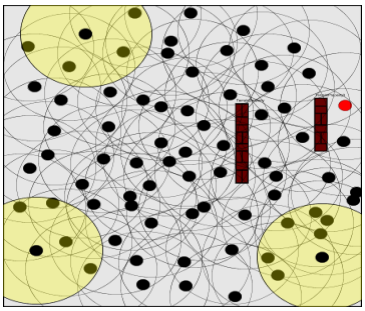
(b)

Fig. 20 Shortest path in terms of power consumption (transmission power), (a): Declivity, (b): VCost

In a network with obstacles, with attenuation depth equal to 0, we can't compare Declivity to VCost because this latter stops without making any hop (figure 21). And in order to make VCost working, the source node has to limit its choices to neighbors with positive progress and this is not available in this case of obstacles.



(a)



(b)

Fig. 21 The attenuation depth of obstacles equals zero (attenuation=0): (a) Declivity, (b)VCost

7. Conclusion

The main problem of geographic routing based on virtual coordinate system is the amount of reference points (anchors) needed to produce a unique reference framework. However, VCS constructed only by the network distance (number of hops) to anchors cannot provide naming uniqueness for general graphs.

Unlike the vast majority of existing protocols that rely heavily on the existence of a minimum of three anchors with or without known geographic position, our new virtual coordinate system is free-anchors, it works independently and with great success apart from this notion and its several shortcomings. To avoid duplicate virtual coordinates, the key idea of its success is to allow each sensor to get its position (its virtual coordinates) basing on the use of the cluster creation. Thus, it is impossible to get two sensor nodes identified with the same virtual coordinates. Therefore, this proposal helps perfectly to solve the problem of guaranteed delivery in geographic routing protocols. Moreover, the space needed to store the virtual coordinates is fixed to store no more than three virtual coordinates whatever the number of sensor nodes deployed in the area of interest.

The main goal of this paper is to propose a lightweight and robust virtual coordinate system for a wireless sensor network, consisting of tiny energy-constrained commodity sensors massively deployed in an area of interest.

The proposed virtual coordinate system can be leveraged by a number of greedy routing protocols mentioned previously like GPRS. Thus, we have tried to propose a simple greedy routing algorithm based on this new VCS. It is designed to be scalable, loop free, energy efficient and it guarantees delivery with the selection of the shortest path using hops' count.

As a perspective, we hope to solve the problem of holes in the network using either this new VCS (VCSCClockwise) or the proposed greedy routing algorithm (Declivity). Other aspects to be analyzed are the study of VCSCClockwise and Declivity towards node mobility, asymmetric links and extension to heterogeneous networks.

References

[1] H. Takagi and L. Kleinrock, Optimal transmission ranges for randomly distributed packet radio terminals. IEEE Transaction on Communications, com-22(3), 1984.

[2] R. Nelson and L. Kleinrock, The spatial capacity of a slotted ALOHA multihop packet radio network with capture, IEEE Transactions on Communications, 32, 6 (1984) 684-694.

[3] T.C. Hou and V.O.K. Li, Transmission range control in multihop packet radio networks, IEEE Transactions on Communications, 34, 1 (1986) 38-44.

[4] G.G. Finn, Routing and addressing problems in large metropolitan-scale internetworks, ISI Research Report ISU/RR-87-180, March 1987.

[5] S. Basagni, I. Chlamtac, V.R. Syrotiuk, B.A. Woodward, A distance routing effect algorithm for mobility (DREAM), Proc. MOBICOM (1998) 76-84.

[6] Y.B. Ko and N.H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, MOBICOM, 1998, 66-75; Wireless Networks, 6, 4 (July 2000) 307-321.

[7] E. Kranakis, H. Singh and J. Urrutia, Compass routing on geometric networks, Proc. 11th Canadian Conference on Computational Geometry, Vancouver, August (1999).

[8] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, 3rd int. Workshop on Discrete Algorithms and methods for mobile computing and communications, Seattle, August 20 (1999) 48-55.

[9] S. Olariua, Q. Xu, A. Wadaa, I. Stojmenovic, Virtual Infrastructure for Wireless Sensor Networks, Book style, 2005, chapter 9 pp. 277-308.

[10] A. Caruso, S. Chessa, S. De, and A. Urpi, Glider: gradient landmark-based distributed routing for sensor networks. INFOCOM 2005. 24th Annual Joint Conf. of the IEEE Computer and Communications Societies. Proc. IEEE., vol. 1, pp. 339–350, 2005.

[11] J. Bruck, J. Gao, and A. Jiang, MAP: medial axis based geometric routing in sensor networks, in IEEE/ACM MOBICOM, 2005, pp. 88–102.
[12] E. Ch´avez, N. Mitton, and H. Tejeda, Routing in wireless networks with position trees. In Ad Hoc Now'07, Mexico, 2007.
[13] A. Caruso, S. Chessa, S. De, and A. Urpi, GPS-free coordinate assignment and routing in wireless sensor networks. In INFOCOM'05, pages 150–160, FL, USA, 2005.
[14] E.H. Elhafsi, N. Mitton, and D. Simplot-Ryl, Cost over progress based energy efficient routing over virtual coordinates in wireless sensor networks. In t2pWSN'07, Helsinki, Finland, 2007.
[15] N. Mitton, T. Razafindralambo, D. Simplot-Ryl, I. Stojmenovic, Hector is an Energy effiCient Tree-based Optimized Routing protocol for wireless networks. In Mobile Ad Hoc and Sensor Networks MSN'08, Wuhan, China, 2008.
[16] I. Stojmenovic, Localized network layer protocols in sensor networks based on optimizing cost over progress ratio. IEEE Networks, 20:21–27, 2006.
[17] S. Olariua, Q. Xu, A. Wadaa, I. Stojmenovic, Virtual Infrastructure for Wireless Sensor Networks, Book style, 2005, pp.107-137.
[18] http://ptolemy.eecs.brekeley.edu
[19] S. Capkun, M. Hamdi, and J. Hubaux, GPS-free positioning in mobile ad-hoc networks. In HICSS, 2001.
[20] E. Ermel, A. Fladenmuller, G. Pujolle, and A. Cotton, On selecting nodes to improve estimated positions. In MWCM, 2004.
[21] D. Niculescu and B. Nath, Ad hoc positioning system. In GLOBECOM, 2001.
[22] D. Niculescu and B. Nath, Ad hoc positioning system (APS) using AOA. In INFOCOM, 2003.
[23] N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller, The cricket compass for context-aware mobile applications. In MOBICOM, pages 1–14, 2001.
[24] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer, Virtual coordinates for ad hoc and sensor networks. In Proceedings of ACM DIALM-OMC'04, October 2004.
[25] Ke Liu and Nael Abu-Ghazaleh. Aligned virtual coordinates for greedy geometric routing in wireless sensor networks. In Proc. of 3rd IEEE international Conference on Mobile Ad-hoc and Sensor Networks (MASS) 2006, Oct. 2006.
[26] Ke Liu and Nael Abu-Ghazaleh, Virtual coordinate backtracking for void traversal in geographic routing. In Procl. of 5th International Conference on AD-HOC Networks & Wireless (Ad hoc Now), August 2006.
[27] D. M. Nicol, M. E. Goldsby, and M. M. Johnson, Simulation analysis of virtual geographic routing. In Proceedings of the 2004 Winter Simulation Conference, 2004.
[28] Ben Leong, Barbara Liskov, and Robert Morris, Greedy virtual coordinates for geographic routing. In Proceedings of ICNP'07, 2007.
[29] Qing Cao and Tarek F. Abdelzaher, LCR: A scalable logical coordinates framework for routing in wireless sensor networks. In RTSS, pages 349–358, 2004.
[30] R. Fonseca, S. Ratnasamy, J. Zhao, C. Tien Ee, D. Culler, S. Shenker, and I. Stoica, Beacon vector routing: Scalable pointto-point routing in wireless sensornets. In Proc. of the 2nd Symposium on Networked Systems Design and Implementation(NSDI 2005), May 2005.
[31] Jonathan Ledlie, Michael Mitzenmacher, Margo Seltzer, and Peter Pietzuch, Wired geometric routing. In Proceedings of IPTPS 2007, 2007.
[32] J. Newsome and D. Song, Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In Proceedings of SenSys'03, November 2003.
[33] L .Buttyan, P .Schaffer, Panel: Position-based aggregator node election in wireless sensor networks, Proceedings of the IEEE International Conference on Mobile Ad hoc and Sensor Systems, MASS, pp. 1–9, 2007
[34] B .Karp, H. T. Kung, Greedy perimeter stateless routing for wireless networks, ACM Mobicom, Boston, 2000.

Karima AKSA received engineer diploma in industrial computer science from Batna University (Algeria) in 2003. She received her M. Sc degree in Computer Science from Batna University in 2006. Currently, she is a lecturer and a PhD student in Batna University under the supervision of Pr Mohamed Ben mohammed on the field of wireless sensor network. She has 6 years of teaching experience. She is interested to the following topics: wireless sensor network (routing, security, scalability, QoS, energy consumption, trucking), bio-inspired systems, NP-complete problems.

Mohamed BENMOHAMMED was born in Constantine, Algeria on December 26, 1959. He received his B.Sc. degree from the High School of computer Science (CERI, Algiers, Algeria), in 1983, and the Ph.D degree in Computer Science from the University of Sidi Belabbes, Algeria, in 1997. He is currently a Professor at Constantine University. His current research interests are: Parallel Architectures, High Level Synthesis, Network, Embedded System, CAD-VLSI, DSP, ASIC, ASIP.

Azeddine BILAMI received the B. Sc degree in computer science from the High School of Computer Science, C.E.R.I. Algiers, in 1983. He obtained his M. Sc. and Ph.D. degrees in computer science from the University of Batna, Algeria, respectively in 1996 and 2005. He is currently a professor at the same university. His research Interests are high-performance interconnects for parallel architectures and multiprocessors, system-on-chip architectures, wireless and mobile networks, TCP/IP and Internet.