VIRTUAL ARCHITECTURE AND ENERGY-EFFICIENT ROUTING PROTOCOLS FOR 3D WIRELESS SENSOR NETWORKS

Vianney Kengne Tchendji[1],Jean Frédéric Myoupo[2], Pauline Laure Fotso[3] and Ulrich Kenfack Zeukeng[1]

[1]Department of Mathematics and Computer Science, University of Dschang, Cameroon

[2]Lab-MIS, University of Picardie Jules Verne, 33 rue St Leu, 80039, Amiens, France

[3]Department of Computer Science, University of Yaounde 1, Cameroon

ABSTRACT

This paper proposes a virtual architecture for three-dimensional (3D) wireless sensor networks (WSNs), a dynamic coordinate system, and a scalable energy-efficient training protocol for collections of nodes deployed in the space that are initially anonymous, asynchronous, and unaware of their initial location. The 3D WSNs considered comprise massively deployed tiny energy-constrained commodity sensors and one or more sink nodes that provide an interface to the outside world. The proposed architecture is a generalization of a two-dimensional virtual architecture previously proposed in the literature, in which a flexible and intuitive coordinate system is imposed onto the deployment area and the anonymous nodes are partitioned into clusters where data can be gathered from the environment and synthesized under local control. The architecture solves the hidden sensors problem that occurs because of irregularities in rugged deployment areas or environments containing buildings by training the network of nodes arbitrarily dispersed in the 3D space. In addition, we derive two simple and energy-efficient routing protocols, respectively for dense and sparse networks, based on the proposed dynamic coordinate system. They are used to minimize the power expended in collecting and routing data to the sink node, thus increasing the lifetime of the network.

KEYWORDS

Wireless Sensor Network, Self-organization, Training protocol, Energy-efficient Routing Protocol.

1. INTRODUCTION

A wireless sensor network (WSN) is a distributed system comprising a number of tiny wireless sensing devices that integrate signal processing and wireless communications capabilities ([1], [2]). Each sensor is powered by a non-rechargeable and non-replaceable battery and has low memory, computation, and transmission range capacities ([1]). Each sensor is able to harvest a set of data in a certain environment, and transmit it in a multi-hop manner to a base station (BS) where it is utilized. WSNs have a significant impact on various fields including military, scientific, industrial, and healthcare. Further, the ubiquity of WSNs has resulted in them pervading society and redefining the way in which we live and work ([3], [4], [5], [6], [7], [8], [2]).

The fundamental goal of a sensor network is to produce, over an extended period of time, globally meaningful information from raw local data obtained by individual sensor nodes. Importantly, this goal must be achieved while prolonging as much as possible the useful lifetime of the network and ensuring that it remains highly available and continues to provide accurate information in the face of security attacks and hardware failure. The sheer number of sensor nodes in a sensor network and the unique characteristics of their operating environment (anonymity of individual sensors, limited power budget, and a possibly hostile environment) pose several challenges for the designers of protocols. In fact, the limited power budget at the individual sensor node level mandates the design of ultra-lightweight data gathering, fusion, and communication protocols. An important guideline in this direction is to perform as much local data processing at the sensor level as possible, avoiding the transmission of raw data through the sensor network. Recent advances in hardware technology have resulted in the biggest challenge of the sensor network community being the development of ultra-lightweight communication protocols for activities such as training, self-organization, network maintenance, security, data collection and fusion, and routing ([4], [9], [10]).

There are several possible models for WSNs. In this work, we consider WSNs in which all the sensor nodes are fixed, short-ranged, and homogeneous. We assume that the BS and all the sensor nodes have a local clock that keeps synchronous time, perhaps by interfacing with the BS. Further, all sensor nodes run the same protocol and can perform computations on the data being sensed. As is customary, time is assumed to be slotted and all transmissions take place at slotted boundaries ([11], [12]). This simple model involves using one or more special sink nodes deployed alongside the sensor nodes. Thus, the raw data collected by individual sensor nodes are fused in stages and forwarded to the sink nodes, which provide the interface to the outside world.
However, in some applications, it is impossible or impractical to deploy sink nodes within the sensor network. In such cases, the task of harvesting the information produced by the sensor network and that of providing an interface to the outside world may be performed by aircraft and/or helicopters overflying the sensor network, or by laser transmission to a satellite constellation. In such a network, security is a crucial point that requires serious consideration. In fact, WSNs have many constraints, including the communication medium, which is wireless: nowadays it is very easy to read, intercept, and even modify the data transmitted, and to compromise an entire network. In addition, the sensors' application context is another problem, as they are usually deployed in hostile environments. Thus, there is a need to secure the protocols in order to guarantee authentication, confidential exchanges ([13], [14]), data integrity, and network availability. Several security protocols have been proposed in the literature. They include TinySec ([15]) which ensure the authentication of the packets sent from a BS to all nodes (broadcast or multicast). Ultimately, a good security system should be able to prevent external attacks (coming from an attacker outside the network) as well as internal attacks (from an attacker inside the network, possibly by compromising a node).

1.1 RELATED WORK

Once deployed, a fundamental prerequisite for self-organization is that sensors need to acquire some form of location awareness, ([9], [16]). Almost all applications benefit that the sensed data be supplemented with location information, but not all of them. The problem for a sensor to know its location in the network is crucial in a anonymous network. One interesting paper on the self-organization of two dimensional WSN is due to Wadaa et al., in [17]. A sensor is trained to acquire its coordinates. A cluster is then the set of sensors having the same

coordinates and it results a 2D virtual WSN ([18], [19], [20]). Howard et al., ([20]) propose an incremental algorithm for self-deployment of in 2D. However there are few papers on the design of 3D WSN. A virtual 3D network architecture for Underwater Sensor Networks (UWSNs) is introduced by Tamoghna et al., ([21]). The main goal of the so called EDETA (Energy-efficient aDaptive hiErarchical and robusT Architecture) ([22]) which based on two-levels hieratical architecture, is to optimize the save node's power. EDETA is more suitable for the implementation of safety applications such as a wireless fire detection system. Boufares et al., ([16]) proposed a 3D architecture based on virtual force which is the generalization on the cellular network in 3D. In the 3D architecture in ([23]) the sensors need to be powerful to carry out the negotiation tactic that ensure network connectivity, and to manage a density control strategy that is used to balance the node distribution, hence we are far from the sensors defined as Micro Electro-Mechanical Systems (MEMS) – miniaturized low-power devices that integrate sensing, special-purpose computing and wireless communications capabilities. Chen and Qian ([24]) derive an algorithm based on ideal fluid dynamics for deploying sensors in 3D. Huang et al., ([25]) show how the coverage problem can be solved in 3D.

1.2 THE CONTRIBUTION OF THIS PAPER

In this paper, we focus on the concept underlying the 2D virtual architecture developed by Wadaa et al., ([17]). We extend their work in 3D and introduce new routing protocol in case of sparse network. The architecture is created and orchestrated by the BS, which is able to split the network into a set of clusters according to the strength and direction of the broadcast it can perform. Our proposed training protocol is a generalization to 3D sparse WSN of the one proposed by Wadaa et al., ([17]) for 2D dense WSN. Our major contributions are as follows:

- Firstly, we propose a virtual architecture comprising a 3D dynamic coordinate system for massively deployed collections of anonymous sensor nodes. This coordinate system yields, at no extra cost, a clustering scheme: two nodes are in the same cluster only if they have the same coordinates.

- We then show that while training the sensor nodes, the process through which nodes learn their coordinates can be performed by a secure binary protocol. This protocol only uses binary comparisons (that are less energy consuming compared to scalar operations) to perform training. It is then called lightweight protocol.

- Next, we propose two energy-efficient routing protocols, respectively for *dense* WSN and *low-density (or sparse)* WSN. The proposed protocols can be used to collect and forward data from the sensors to the sink node. They use the dynamic coordinate system to minimize the power expended in collecting and routing data.

The remainder of this paper is organized as follows: Section 2 presents the lightweight training protocol proposed for virtual clusterisation. Sections 3 and 4 describe the proposed routing protocols, respectively for *dense* WSN and *low-density* WSN, used to send the collected data to the BS. Section 5 concludes this paper.

2. TRAINING A 3D WSN: VIRTUAL ARCHITECTURE

2.1. THE MAIN LINES

The practical deployment of many sensor networks will result in sensors initially being unaware of their location: they must be trained with this vital information. Further, owing to limitations in form factor, cost per unit, and energy budget, individual sensor nodes are not expected to be GPS-enabled. Moreover, many probable application environments limit satellite access.

The localization problem is for individual sensor nodes to determine, as closely as possible, their geographic coordinates in the area of deployment. The most striking solutions to the localization problem are based on multilateration: sensor nodes receiving location messages from at least three sources can approximate their own locations. Langendoen and Reijers ([26]) provide a good survey of localization protocols for WSNs.) In some other applications, exact geographic location is not necessary: all that the individual sensor node needs is coarse-grain location awareness. There is an obvious tradeoff: coarse-grain location awareness is lightweight but the resulting accuracy is only a rough approximation of the exact geographic coordinates. It can be made by an overflying aircraft or helicopter. In this case, all that the individual sensor nodes need is to determine their approximate distance to three different positions of the training agent.

Our approach is different: we obtain this coarse-grain location awareness by the training protocol that imposes a coordinate system onto the sensor network. An interesting by-product of our training protocol is that it provides a partitioning into clusters and a structured topology with natural communication paths. The resulting topology makes it simple to avoid collisions between transmissions of nodes in different clusters, between different paths, and also between nodes on the same path. Our clustering protocol has the following desirable features:

- lightweight, as a by-product of training;
- organizes anonymous asynchronous nodes;
- a cluster is the locus of all nodes having the same coordinates; and
- individual nodes need not know the identity of other nodes in their cluster.

Hereafter we assume a WSN that consists of a sink and a set of sensors randomly deployed in its 3D broadcast range, as illustrated in Figure 1. For simplicity, we assume that the sink node is centrally placed, although this is not actually necessary. The task of training refers to the imposing of a coordinate system onto the sensor network in such a manner that each sensor belongs to exactly one sector.

We assume that the sink node can make *l* omnidirectional transmissions, *m* horizontal directional transmissions, and *n* vertical directional transmissions. The coordinate system divides the sensor network area into equiangular wedges (or sections). In turn, these wedges are divided into sectors by means of concentric spheres or coronas centered at the sink and whose radii are determined to optimize the transmission efficiency of sensors-to-sink transmission. Sensors in a given sector are mapped to a cluster; the mapping between clusters and sectors is one-to-one. With reference to Figure 1, the task of training a sensor network involves establishment of the following:

1. **Coronas**: The deployment area is covered by *l* coronas determined by *l* concentric *spheres* of radii $r_1 < r_2 < \cdots < r_l$ centered at the sink node;

2. **Horizontal wedges**: The deployment area is ruled into *m* horizontal angular wedges centered at the sink node;

3. **Vertical wedges**: The deployment area is ruled into *n* vertical angular wedges centered at the sink node.

As illustrated in Figure 1, at the end of the training period, each sensor node has acquired *three* coordinates: the identity of the corona in which it lies, and the identity of the horizontal and vertical wedges to which it belongs. Importantly, the locus of all the sensor nodes that have the same coordinates determines a cluster.

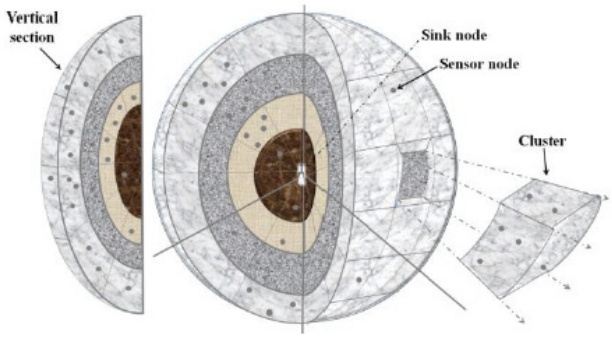

Figure 1: A trained 3D sensor network.

2.2. THE PROPOSED LIGHTWEIGHT TRAINING PROTOCOL

The main goal of this section is to present, in detail, our proposed lightweight scalable training protocol for WSNs. The key advantage of this protocol is that each node participating in the training, incurs an energy cost that is logarithmic in the number of clusters and wedges defined by the protocol.

After deployment, nodes sleep until they are awakened by their individual timers. Thus, each node sleeps for a random period of time, wakes up briefly and, if it hears no messages of interest, selects a random number x and returns to sleep x time units. Clocks are not synchronized but over any time interval $[t, t + \Delta t]$, a percentage directly proportional to t of the nodes are expected to wake up briefly. During this time interval, the sink continuously repeats a call to training, specifying the current time and a rendezvous time. Thus, in a probabilistic sense, a certain percentage of the nodes will be selected for training. The time interval t can be adjusted to control the percentage of nodes selected. Using the synchronization protocol proposed by Wadaa et al., ([17]), the selected sensor nodes reset their clocks and set their timer appropriately before returning to sleep. In fact, it is natural to assume that, just prior to deployment, the sensor nodes are synchronized. However, because of natural clock drift, resynchronization is necessary. This is performed with respect to the master clock running at the sink.

2.2.1. THE CORONA TRAINING PROTOCOL

The corona training protocol is similar to that developed by Wadaa et al., ([17]). However, the corona training protocol operates as follows: formally, consider an l-leaf binary tree T and refer to Figure 2 (borrowed from [17]). In the figure, the leaves are numbered left to right from one to l.

The edges of T are labeled by 0s and 1s in such a manner that an edge leading to a left subtree is labeled 0, and an edge leading to a right subtree is labeled 1.

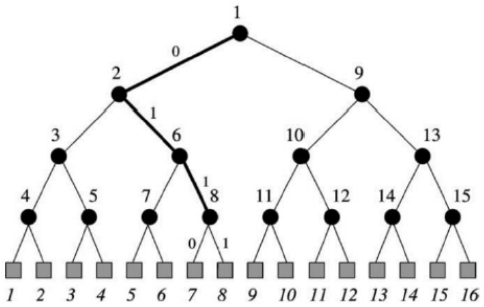

Figure 2: Illustration of corona training (Wadaa et al., ([17]).

In this corona training protocol, the preorder and inorder numbers of internal nodes in T correspond, respectively, to time slots in the training protocol and to the transmission ranges used by the sink.

2.2.2. THE WEDGES TRAINING PROTOCOL

In the wedges training protocol, the deployment area is divided into *m* horizontal angular wedges with angles α and *n* vertical angular wedges with angles β. This is shown respectively in Figures 3(a) and 3(b). As in the corona training protocol, sensors must read a string of length(or $\log_2 n$ bits for vertical angular wedges) they possess. The time is divided in slots, $s_1, s_2, \ldots, s_{m-1}$, and at each slot, the sink node makes a directional horizontal transmission of angle $\alpha_i, i \in \{1,2,\ldots,m-1\}$ (respectively, a directional vertical transmission of angle $\beta_j, j \in \{1,2,\ldots,n-1\}$).

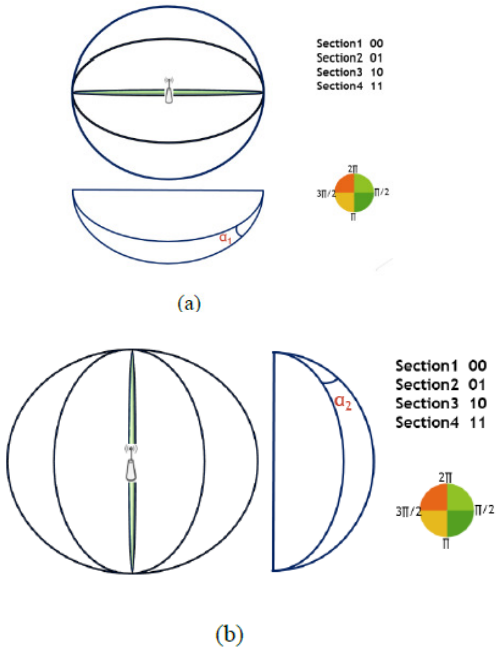

Figure 3: Horizontal and vertical wedges: (a) Horizontal wedges. (b) Vertical wedges

The vertical wedges are trained after the horizontal wedges. The wedge training protocol is virtually similar to the corona training protocol, but the omnidirectional transmissions of the sink node are replaced by the directional transmissions. The sink node uses the binary tree in Figure 2 to determine the values of the different angles to use in each slot. Figure 4 shows an example of vertical directional transmissions for $n = 8$:

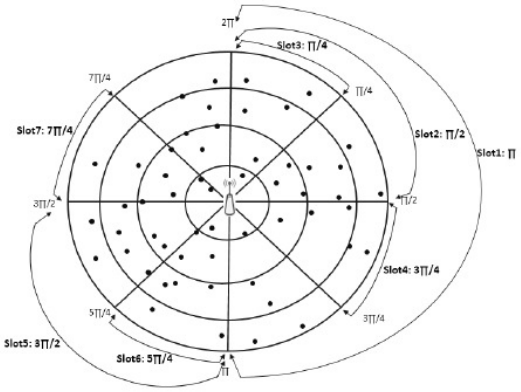

Figure 4: Directional transmissions of the sink node for $n = 8$.

In the next section, we describe how the data collected by the sensors arrive at the sink node.

3. ROUTING IN A DENSE TRAINED SENSOR NETWORK

In this section, we show how data collection is achieved in a network that we assume to be dense, i.e., after the clusterisation protocol, we have the following property:

$\forall (i, j, k) \leq (l, m, n)$, *cluster* (i, j, k) is not e*mpty* (2)

We assume that the sensor network is already clustered according to the technique presented in the previous section. This facilitates structuring of data routing at two levels: within a cluster and between clusters. To achieve this goal, we introduce two concepts:

- *Relay cluster*: the data collected by sensors in a cluster (i, j, k) can go through many clusters to reach the sink node. Cluster $(i\text{-}1, j, k)$ is designated as a *relay cluster* of cluster (i, j, k). It is the first cluster that transits the data collected by the sensors of cluster (i, j, k);
- *Cluster-head (or gateway node)*: This is a unique node in a given cluster that is responsible for forwarding data to the outside (i.e., to the *relay cluster*). It is used to conserve the energy of the sensors in a cluster and prevent unnecessary overloading of the network.

In our routing protocol, we make the following assumptions:

1. Each node has a unique identifier *ID* throughout the network;
2. A sensor is able to assess its residual energy (which we denote *E*r);
3. A message sent by a sensor is received after a finite time (a slot) by all its neighbors;
4. The clustering is set in such a way that all the sensors of a given cluster can directly communicate between them and with some sensors of their neighboring clusters.

We consider that all sensors communicate with the same frequency. Thus, there may be many collisions if communications are not well managed. To avoid collisions between packets transmitted, several communication channel management protocols exist, for example, frequency division multiple access (FDMA), time division multiple access (TDMA), and code division multiple access (CDMA). We chose the carrier sense multiple access/collision avoidance (CSMA/CA) protocol ([27]) for the reservation of the communication channel. We chose it for the following reasons:

1. The bandwidth is not subdivided, allowing fast transfer of data;
2. In most WSNs, the amount of data collected is not large, so they are unlikely to be fragmented; even if that were the case, the fragmentation-reassembly mechanism of CSMA/CA solves the problem;
3. It enables shared access to the channel, while solving interference problems, and concurrent access of sensors;
4. Its back-off algorithm gives the same probability to all the sensors to access the channel.

As the sensors have a limited source of energy, we must prevent redundant data releases. Indeed, it is important to reduce the network traffic to avoid collisions, conserve the energy of the sensors, and prevent unnecessary overloading of the network. To achieve these objectives, data collected by sensors must contain a minimum number of messages.

We use cluster-heads to satisfy these conditions. Cluster-heads are the collector nodes in a cluster; more specifically, they gather data in their cluster before sending them to the relay cluster. Therefore, the data held by cluster-heads can be compressed or aggregated before transfers. The data can also be filtered to prevent the transfer of identical messages. Thus, management of the routing within a cluster increases the energy consumption of the gateway node compared to the ordinary nodes. Consequently, the role of gateway is given only to nodes that have sufficient energy.

To start the routing protocol, the sink node repeats a call for routing, specifying the current time and a rendezvous time. On that date, the sensors must collect data and send them towards the sink node. Thus, our routing protocol is divided into two steps: (1) election of a cluster-head in each cluster, and (2) transmission of collected data to the sink node. The second step is itself subdivided into two phases: routing within a cluster, and routing between clusters.

3.1. DISTRIBUTED CLUSTER-HEAD ELECTION

Election of cluster-heads is conducted in a distributed manner, and at the same time in all the clusters. A sensor can act as a cluster-head only if its residual energy Er is greater than a threshold Es defined by a network administrator. For the gateway node (cluster-head) election process, sensors in clusters (i, j, k), $i > 1$, having a residual energy greater than the threshold Es, send a *Hello* message1 to their relay cluster $(i-1, j, k)$, and wait for an acknowledgment (a *Hi* message2). Then, all the sensors that receive a *Hi* message will be sure that they can access the sensors of their relay cluster. They postulate as gateway node by diffusing a *Head* message3 in their cluster (i, j, k). The sensor having the highest residual energy *Er* among those that postulate is elected gateway node. If several candidates have the same amount of residual energy, the one with the highest identifier (*ID*) is elected.

1. The *Hello* message contains two pieces of information, ((*i, j, k*), (*i*-1, *j, k*)): the coordinates of the current cluster and those of the relay cluster.
2. The *Hi* message contains two pieces of information, ((*i, j, k*), (*i*+1, *j, k*)): the coordinates of the relay cluster and those of the current cluster.
3. The *Head* message contains three pieces of information, (*ID*, *E*r, (*i*, *j, k*)): identifier of the current sensor, its residual energy, and the coordinates of its cluster.

The sensors in clusters (1*, j, k*)*,* do not send any *Hello* message, instead they respond to Hello messages sent from clusters (2*, j, k*) with *Hi* messages*.* Subsequently, the sensors with residual energy greater than the threshold *Es*, send a *Head* message in their own cluster. The criteria for selecting the gateway node are the same here as in clusters (*i, j, k*), *i* >1.

To conserve energy and prevent redundant transmission, each sensor reacts only once after the reception of *Hello* and *Hi* messages. This first step of our protocol (distributed cluster-head election) is performed by each sensor using pseudo-codes 1, 2, 3, 4, and 5 described in Annex 1. At the end of this step, a cluster-head is elected in each cluster, and the next task (the first phase of the second step) is to route the data collected towards the sink node through cluster-heads.

3.2. ROUTING WITHIN A CLUSTER

When the sensors in a given cluster wish to send messages to the sink, they send them to the gateway node (or cluster-head), which will in turn send them to their relay cluster, and so on, until the sink node is reached. When a sensor receives a message with a destination in its cluster, it forwards it to the gateway node; otherwise, it ignores the message.

To prevent redundant messages being sent, each gateway node keeps track of the last message received from each sensor within its cluster. Thus, if an identical message is received from another sensor, it will not send it again.

When the residual energy of the gateway is below the threshold *Es*, it initiates re-election of the gateway, but this time it does not postulate. If during this re-election, there is no sensor that has residual energy greater than the threshold *E*s, two solutions are possible with the last active cluster-head:

- It can decrease the threshold value (for example by half, i.e., $E_s/2$), and inform the sensors of its cluster. This allows the network to stay functional longer;
- It can also send a warning message to the sink node in order that, either new fixed sensors are deployed in the cluster or a mobile sensor is sent there. In the case of a rechargeable network, this alert message can initiate the charging mechanism of the sensors.

Remark: In our work, we assume that all the sensors in cluster (1,*j,k*) can directly communicate with the sink node. In the opposite case, sensors in cluster (1,*j,k*) that are candidates to become the gateway node must also verify that they can communicate with the sink node before sending the *Head* message. In this case, the sink node must participate in the election process as the only sensor in cluster (-1,-1,-1).

Once the data are forwarded to the cluster-head, it must route them wisely to sink node, possibly through other clusters. The next task (second phase of the second step) in our protocol describes how the messages collected in the different clusters are routed to the sink node.

3.3. ROUTING BETWEEN CLUSTERS

Property (2) is very interesting because cluster (*i, j, k*) can directly communicate with cluster ($i-1$, *j, k*) (its *relay cluster*). However, the sensors in cluster (1, *j, k*) can directly communicate with the sink node whose coordinates are (-1, -1, -1). The sink node is the final destination of all the messages traveling in the network. Thus, data can successfully move from cluster (*i, j, k*) to cluster (*i*-1, *j, k*), and then to (*i*-2, *j, k*),..., (1, *j, k*), until (-1, -1, -1).

In order to conserve energy, sensors are not awake all the time; they can automatically sleep for a random period of time, and wake up for a period of time (defined by the network administrator) to collect data and send them to the sink node. When sensors are awake, they can perform two possible actions:

- *An event occurs:* Sensors in cluster (*i, j, k*) make a message containing the code of the event and the coordinates of the cluster in which it lies, then send it to its cluster-head, for it to forward to the relay cluster, or send it directly to the sink node if $i = 1$;
- *Reception of a message:* The sensor checks if it is the receiver; in this case, it forwards it to its cluster-head, or to the sink node if its coordinates are (1,*j,k*). Otherwise, it simply ignores the message.

This process is illustrated by Figure 5, in which it can be seen how the data are collected within a cluster and between clusters. The dotted arrows show that ordinary sensors transfer the data they hold only to their cluster-head, and never out of their cluster. The continuous line arrows show that communication between clusters is made only by the cluster-heads, and always in the same angular wedge.

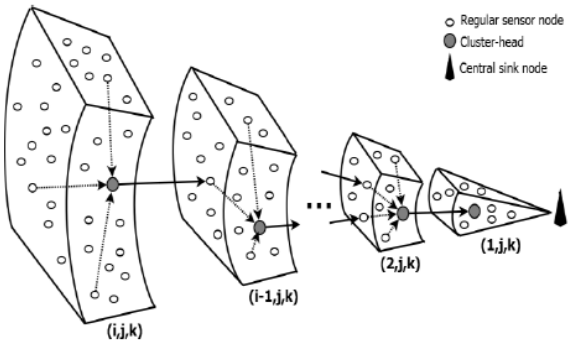

Figure 5: Routing in a dense sensor network.

This protocol has a number of advantages:

1. The CSMA/CA protocol minimizes collisions during transmissions;

2. The data collected in a cluster can be grouped, aggregated, or compressed before routing, which helps to reduce the amount of information flowing through the network, and the number of messages generated by the sensors in a cluster;

3. Routing is carried out by cluster-heads, allowing ordinary sensors to conserve energy;

4. Data collected in an angular wedge never go through another, this prevents overall network congestion;

5. The cluster-head re-election mechanism ensures the longevity of the network structure, and provides a first step in fault tolerance.

3.4. SIMULATION RESULTS

In the simulation conducted, we focused on the energy percentages of the sensors nodes and the cluster-head during the routing process in the 3D virtual architecture described above. Our simulations were performed using the WSNet software ([28]) based on a virtual architecture of 5 km in diameter, 10 coronas, eight horizontal wedges, and eight 45° vertical wedges, on which we randomly generated dense networks of 1000 sensors with a range of 500 meters. The ideal case, in which the BS is located at the center of the area, was used.

We performed tests repeatedly and averaged the results. The energy model used is one that has been utilized in many efficiency studies ([29]): $E = ET + ER + ES + EI$, where ET and ER are, respectively, the total energy used by the sensors for transmissions and receptions in the network; ES and EI are, respectively, the energy expended by the sensors while sleeping or simply awake. The curves were made with version 5.0 of the gnuplot software.

While $E\mathrm{T} \geq E\mathrm{R} \geq E\mathrm{S} \geq E\mathrm{I}$, the cluster-heads lose much more energy than the ordinary nodes. In fact, they make more transmissions in their cluster. This is illustrated in Figure 6, where the energy of the cluster-head decreases faster than that of ordinary nodes. On the other hand, the sharp drop in energy in the first slots corresponds to initialization of the communication phase, which includes the election of cluster-heads. Once the cluster-heads are elected, routing is performed with lower energy consumption.

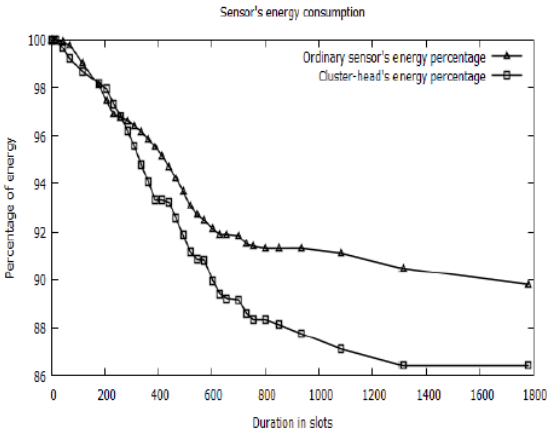

Figure 6: Energy consumption of a sensor during routing.

For a deployment of 500 sensors in a virtual architecture containing four coronas, and 90° vertical and horizontal wedges, there are 64 clusters. Thus, there are 64 cluster-heads, which have a larger workload than ordinary nodes; i.e., a ratio of 12.8% of sensors. However, with 1000 sensors, that ratio decreases to 6.4% of sensors. Therefore, this percentage decreases as the number of sensors increases. Considering the re-election of cluster-heads when the energy of the current cluster-head is less than the threshold, our protocol provides several cluster-head generations in each cluster, and thus increases the lifetime of the network.

Figure 7 shows that without cluster-heads, the energy of the sensors decreases very rapidly, whereas with cluster-heads, the overall energy of the sensors decreases very slowly. Our method of reducing transmission of identical messages further conserves this energy.

The protocol presented in this section is not suitable for sparse networks, in which the sensors partially occupy the deployment area. In this case, routing of data to the sink node can no longer follow the clear and optimal path (angular wedge) used for dense networks.

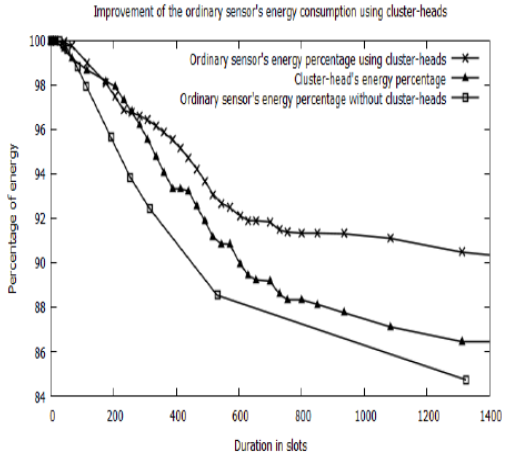

Figure 7: Impact of the use of cluster-heads on the conservation of the energy of ordinary sensors

4. ROUTING IN A SPARSE TRAINED SENSOR NETWORK

Training does not allow the sink node to distinguish empty clusters and non-empty clusters. In a sparse network, the following property holds:

$$\exists\,(i, j, k) \leq (l, m, n)\ ,\ cluster\ (i, j, k)\ is\ empty \qquad (3)$$

In contrast to dense networks, in which the data collected by cluster (i, j, k) are routed to the sink node through cluster $(i\text{-}1,j,k)$, in a low-density network, there is no guarantee that cluster $(i\text{-}1,j,k)$ will contain sensors that can forward the messages. Thus, an empty clusters detection phase must precede routing of data in order to define the best path for the data to follow. This empty clusters detection phase also makes it possible to determine the area actually covered by the sensors after the deployment, and therefore gives the possibility of reacting accordingly.

A cluster is considered empty if it contains no sensor, or if it contains a set of sensors disconnected from the rest of the network.

4.1. DETECTION OF EMPTY CLUSTERS AND DISTRIBUTED CLUSTER-HEAD ELECTION

Knowledge of the distribution of the sensors in the space of interest enables the sink node to determine the optimal path from a cluster to the sink (or to another cluster).

For l coronas, m horizontal wedges, and n vertical wedges, there are $(l \times m \times n)$ clusters in the virtual architecture; therefore, for each message received, the sink regularly updates two tables $h(l, m, n)$ and $relay(l, m, n)$. Each entry (i, j, k) of table $h(l, m, n)$ contains one if cluster (i, j, k) is not empty and zero otherwise, allowing the sink to obtain a global view of the sensor's distribution; and each entry (i, j, k) of the table $relay(l, m, n)$ contains the coordinates of the relay cluster of the cluster (i, j, k).

Because cluster-heads significantly affect conservation of the energy of the ordinary sensors, as well as the routing of data, they must be elected as soon as possible. We propose to realize distributed cluster-heads election during the empty clusters detection phase.

4.1.1. THE SINK'S ALGORITHM

At the beginning, the sink periodically broadcasts the date on which the detection algorithm will begin. All sensors are awake when the sink initiates the detection. It then transmits its neighboring clusters (those of the first corona) a *Detect* message containing its coordinates (-1, -1). Then, it waits for acknowledgments (ACK messages) that will allow it to update tables $h(l, m, n)$ and $relay(l, m, n)$. Owing to network connectivity, it is certain that at least one sensor will receive this message. During the process, each message transmitted by a sensor towards the sink contains the coordinates of its cluster and those of its relay cluster. Table $h(l, m, n)$ is also initialized to zero. At each reception of a message from a sensor in a cluster (i, j, k), the sink node puts one in $h(i, j, k)$ and, in the entry $relay(i, j, k)$, it assigns the coordinates of its relay cluster obtained from the variable *relay* of the received message. At the end of the algorithm, cluster (i, j, k) is considered empty when $h(i, j, k) = 0$, and the relay cluster of cluster (i, j, k) is determined by the value of the entry $relay(i, j, k)$.

4.1.2. THE SENSOR'S ALGORITHM

The network must be connected; therefore, for all clusters (i, j, k) considered non-empty, there is always a path from it to the sink node. Isolated clusters cannot reach the sink and are considered empty even if there are not. For the sensors, there are three main events in the detection of empty clusters: reception of a *Detect* message asking sensors to indicate their coordinates (here, this is equivalent to the reception of a *Hi* message in Section 4.1); reception of a *Head* message sent by a sensor of the same cluster to postulate as cluster-head (it starts the cluster-head election process); and reception of an *ACK* message sent by a cluster-head to the sink node to indicate its coordinates and those of its relay cluster.

Reception of a *Detect* message

In order to reach the sink by the shortest path, the sensors of each cluster must choose their relay cluster wisely. To accomplish this, each sensor has to route only the first *Detect* message that it receives. When a sensor receives a *Detect* message, it checks if it has not already had to route a *Detect* message. If it has not, it forwards it to enable the sensors in the other clusters to signal their coordinates. Otherwise, it ignores the message. Subsequently, each sensor that received its first *Detect* message, constructs a *Head* message and forwards it in its cluster to postulate as cluster-head.

Reception of a *Head* message

This election is carried out as in Section 4.1. At the end of the cluster-head election, the elected cluster-head sends an *ACK* message to the sink through their relay cluster. The cluster-heads of the first corona send their *ACK* message directly to the sink, while those of the other coronas send theirs on through the relay cluster.

Reception of an *ACK* message

A sensor that receives an *ACK* message from its neighbor node checks whether the message is for its cluster and, if it is, it sends it to its gateway node (cluster-head), which checks if it has already routed an *ACK* message from the same cluster. If it has not, it sends the message through its relay cluster. Otherwise, it simply ignores the message.

4.2. ROUTING

The sink node uses tables *h*(*l, m, n*) and *relay*(*l, m, n*) to build a message propagation tree that presents the path traveled by the messages from each cluster. This tree can be optimized (for example, by decreasing its height) to decrease the distance between the sink and the different clusters.

To send the messages to the sink node, the sensors in cluster (*i, j, k*) send them to their relay cluster (through their cluster-head), which in turn transmits them to their relay cluster, and so on, to the sink node. Figure 8 illustrates this communication process. Clusters (*i*-1*, j, k*), (2*, j*-1*, k*), and (1*, j*-1*, k*) are empty. It can be seen that cluster (*i, j*-1*, k*) is the relay of cluster (*i, j, k*), and the messages coming from cluster (*i, j, k*) pass through clusters (*i,j*-1,*k*), (*i*-1*, j*-1*, k*), (*i*-2*, j*-1*, k*), …, (2*, j*-1*, k*), and (1*, j*-1*, k*) to reach the sink node. These messages pass alternately through two angular wedges.

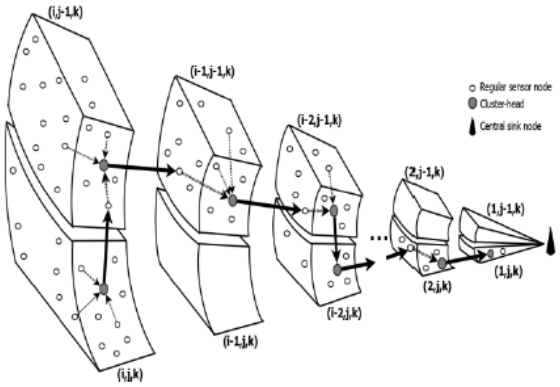

Figure 8: Routing in a sparse sensor network.

The methods used for channel access and communication within and outside of clusters are the same as those used for routing in a dense sensor network.

4.3. SIMULATION RESULTS

For a virtual architecture comprising eight coronas and 45° horizontal and vertical angular wedges, i.e., a total of 512 clusters, Figure 9 shows the average percentage of empty clusters obtained after 10 executions of our empty clusters detection algorithm. These tests were conducted with a random deployment of the sensors on the space of interest. The number of sensors varied between 100 and 10000. It can be seen that for these tests, more than half of the clusters are empty, which sufficiently illustrates the need for the empty cluster detection phase.

By depicting the deploying of sensors on approximately 50% of the space of interest, Figure 10 clearly shows that cluster-heads consume more energy than ordinary sensors. Because there are fewer sensors in the network (compared to a dense network), the energy difference between cluster-heads and ordinary sensors is lower than that in a dense network. The rapid decrease in energy in the first slots is due to the empty clusters detection phase and cluster-heads election.

It is still important to note that the simulations were performed under ideal circumstances: the area of the sensors is spherical, the BS is at the center, and environmental constraints were not considered. Evidently these ideals are rarely met (the sink node can be at the network edge, sensors have rectangular area, etc.).

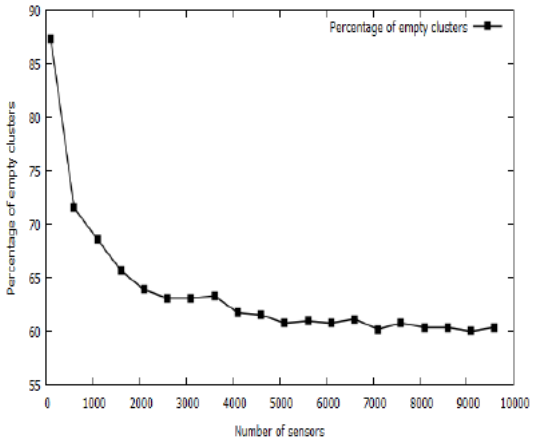

Figure 9: Percentage of empty clusters.

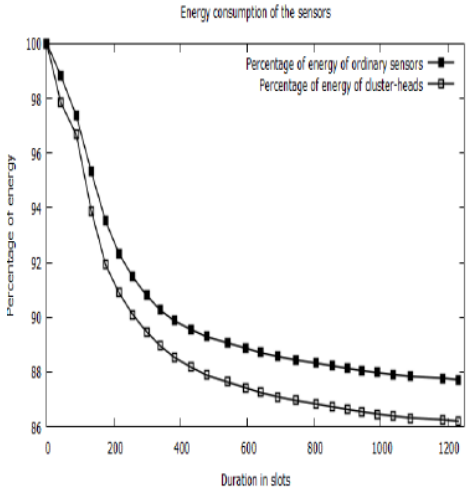

Figure 10: Energy consumption during routing in a sparse WSN.

5. CONCLUDING REMARKS

This paper proposed a virtual architecture for 3D WSNs and a dynamic coordinate system for collections of anonymous sensor nodes deployed in the space. The coordinate system provides, at no extra cost, an interesting clustering scheme in which two nodes are in the same cluster only if they have the same coordinates. It is important to note that this clustering scheme operates for anonymous sensor nodes. Thus, sensor nodes do not know the identity of the other nodes in the same cluster. A lightweight and secure protocol for use by the sensors, during training, to learn their coordinates was also proposed. As it is energy efficient, this training can be repeated on either a scheduled or an ad hoc basis to provide robustness and dynamic reorganization. We also showed that in a trained WSN the tasks of routing and data fusion can be performed by very simple and energy-efficient protocols.

Two simple and energy-efficient routing protocols for dense and sparse networks, based on the dynamic coordinate system, and which minimize the power expended in collecting and routing data to the sink node were also presented. Further, the concepts of *cluster-head* and *relays cluster*,

which facilitate centralized data compression or data aggregation in a cluster before transfers were also introduced for more efficiency. They are also used to avoid network congestion, and to preserve the overall energy of the sensors; thereby increasing the lifetime of the network.

However, despite these encouraging results, there is more work to be done. Firstly, in our model, synchronization of sensors with the global clock of the sink node after a period of sleep, has not yet been actualized. Further, the channel access method needs to be improved. Secondly, new challenges have appeared in sparse sensor networks. These challenges include management of network connectivity, routing optimization by seeking an optimal means of reaching the sink node, and management of mobile sensors in areas not covered by fixed sensors. Finally, because sensors are often deployed in unattended areas, they can be subject to external attack. Thus, security is also a major issue for routing in this architecture.

APPENDIX

Cluster-heads election pseudo-code

The pseudo-codes presented in this section enable sensors to choose a gateway node in each cluster. The notations presented in Table 1 are used.

Table 1: Parameters of the training and routing pseudo-code

Parameter	Description
l	Total number of coronas in the coordinate system.
m	Total number of horizontal wedges in the coordinate system.
n	Total number of vertical wedges in the coordinate system.
time()	Returns the local clock time in a sensor node or the sink node.
radio.transmit(r, m)	Transmit message m using a transmission radius of r.
radio.receive()	Put radio in receive mode for one message time; if a message is being transmitted it returns the message.
message(X)	Constructs a message with ordered tuple X as its contents.
destination(*Message*)	Indicates whether the Message was sent towards the sensors of a cluster.
r_i	Radius of concentric sphere i, $1 \leq i \leq l$
λ	Time slot size. For simplicity, we assume that it is also the time required to transmit (receive) a protocol message.

The *init*() procedure (Algorithm 1) facilitates initialization of the cluster-head election process in each cluster by broadcasting the message *Hello*. It helps the sensors that have sufficient energy to test the connectivity to their relay cluster. The *reception_Hello*() procedure (Algorithm 2) describes the behavior of the sensors while receiving the *Hello* message, it enables them to send an acknowledgment (*Hi* message) to the sensors in the

Algorithm 1: *init*() procedure

```
Data: E_r, E_s
Output: Hello message
Begin
    if (E_r > E_s) then
        if (i > 1) then
            Hello ← message((i, j, k), (i-1, j, k));
            radio.transmit(r, Hello) ;
        else already_receive_Hello ← true;
        end if
    end if
End
```

Algorithm 2: After reception of *Hello* message: *reception_Hello*()

```
Data: Hello message
Output: Hi message
Begin
    already_receive_Hi ← true;
    Hi ← message((i, j, k), (i+1, j, k));
    radio.transmit(r, Hi);
End
```

cluster who want to become cluster-heads.

The *reception_Hi*() procedure (Algorithm 3) describes the actions performed by a sensor on receiving a *Hi* message; it enables the sensor to initialize the coordinates of its relay cluster, as well as the information to offer as cluster-head. The *Head* message containing id and the residual energy of the sensor is diffused in the cluster to postulate the sensor as gateway node. The *reception_Head*() procedure (Algorithm 4) is used on receiving a *Head* message to choose the best cluster-head among the applicants.

Algorithm 3: After reception of *Hi* message: *reception_Hi*()

```
Data: Hi message
Output: Coordinates of the relay cluster.
Begin
    already_receive_Hi ← true;
    relay_cluster ← (i-1, j, k);
    gateway_node ← (ID, E_r);
End
```

Algorithm 4: After reception of *Head* message: *reception_Head*()

```
    Data: Head message
1   Output: ID of the gateway node.
2   Begin
3       if (gateway_node == NULL) or (Head.E_r > gateway_node.E_r)
4       or ((Head.E_r == gateway_node.E_r) and (Head.ID > gateway_node.ID)) then
5               gateway_node ← (Head.ID, Head.E_r) ;
6       end if
7   End
```

The *election_gateway*() procedure (Algorithm 5) makes calls to all the other procedures to complete the gateway node election process. It comprises two phases: In the first phase, sensors with residual energy greater than *E*s verify whether they can communicate with the relay cluster (procedures 1, 2, and 3). In the second phase, the choice for gateway node is made.

The variables *already_receive_Hello* and *already_receive_Hi* ensure that a sensor reacts only once after reception of *Hello* and *Hi* messages.

Algorithm 5: *election_gateway*()

```
    Data: Times T1 and T2.
    Output: ID of the gateway node.
1   Begin
2       already_receive_Hello ← false;
3       already_receive_Hi ← false;
4       step1 ← time() + T1;
5       init();
6       while (time() ≤ step1) do
7           if ((Hello ← radio.receive()) == true) and (already_receive_Hello == false)
            and (destination(Hello) == true) then
8               reception_Hello();
9           end if
10          if ((Hi ← radio.receive()) == true) and (already_receive_Hi == false)
            and (destination(Hi) == true) then
11              reception_Hi();
12          end if
13      end while
14      step2 ← time() + T2;
15      if (already_receive_Hi == true) then
16          Head ← message(ID, E_r, (i, j, k));
17          radio.transmit(r, Head);
18      end if
19      while (time() < step2) do
20          if ((Head ← radio.receive()) == true) and (destination(Head) == true) then
21              reception_Head();
22      end while
23  End
```

Each sensor makes only one call to the *init*() procedure and reacts only once on receiving *Hello* and *Hi* messages. As each sensor can be a candidate in the gateway node election, a sensor can respond several times when a *Head* message is received. As we assumed the network is dense, we cannot predict the number of sensors in each cluster. Consequently, we cannot predict the number of times Procedure 4 will be called. Therefore, it is up to the network administrator to parameterize the duration of each phase of Algorithm 5. Nevertheless, the duration of each phase should enable the execution of Procedures 2 and 3 once, and the execution of Procedure 4 several times.

***Corollary*:** The times $T1$ and $T2$ of each phase of Algorithm 5 should be determined according to the number of slots necessary for execution of Procedures 2, 3, and 4 and to the maximal number of sensors in a cluster.

REFERENCES

[1] Akyildiz, I. F., Su, W., Y. Sankarasubramaniam, Y. and E. Cayirci, E. (2002) 'Wireless sensor networks: A survey, *Comput. Networks'*, 38(4), 393-422.

[2] Zhirnov, V. V. and Herr, D. J. (2001). 'New frontiers: Self-assembly and nanoelectronics'., *Computer*, 34(1), pp. 34-43.

[3] Agre, J. and Clare L. (2000). 'An integrated architecture for cooperative sensing networks', *Computer,* 33(5), pp. 106-108.

[4] Dargie, W. W. and Poellabaurer, C. (2010) *'Fundamentals of Wireless Sensor Networks: Theory and Practice*,, Wiley.

[5] Intanagonwiwat, C., R. Govindan,R. and D. Estrin, D. (2000) 'Directed diffusion: A scalable and robust communication paradigm for sensor networks'. *MOBICOM,* pp. 56-67.

[6] Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V. and Culler D. E. (2001) 'Spins: Security protocols for sensor networks', In: *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 189-199.

[7] Shen, C. C., Srisathapornphat, C. and Jaikaeo, C. (2001) *'Sensor information networking architecture and applications'*, IEEE Pers. Commun., 8(4), 52-59.

[8] Tilak, S., Abu-Ghazaleh, N. B. and Heinzelman, W.(2002) 'A taxonomy of wireless micro-sensor network models'. *ACM SIGMOBILE Mobile Computer Communication Review*., 6(2), 28-36.

[9] El Emary,I. M. M. and S. Ramakrishnan, S. (2013) '*Wireless Sensor Networks: From Theory to Applications'*, CRC Press.

[10] Li, Y. and Thai, M. T. (2013). *Wireless Sensor Networks and Applications,* Springer.

[11] Bar-Yehuda, R., Goldreich, O. and Itai, A. (1991) 'Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection'. *Distributed Computing*, 5(2), pp. 67-71.

[12] Bertzekas,D. and Gallager R. (1992) *Data Networks*, Second Edition, Prentice-Hall.

[13] Li, C., Wang, Z. and Yang, C. (2011) 'Secure Routing for Wireless Mesh Networks', *International Journal of Network Security*, 13(2), 109-120.

[14] Saroit, I. A., El-Zoghdy, S. F. and Matar, M. (2011) 'A scalable and distributed security protocol for multicast communications', *International Journal of Network Security*, 12(2), 61-74.

[15] Karlof, C., Sastry, N. and Wagner, D. (2000) 'TinySec: A link layer security architecture for wireless sensor networks', *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 162-175, ACM.

[16] Boufares, N., Khoufi, I., Minet, P., Saidane, L. and Ben Saied, Y. (2015) 'Three dimensional mobile wireless sensor networks redeployment based on virtual forces', *IEEE Wireless Communications and Mobile Computing Conference (IWCMC),* pp.563-568.

[17] A. Wadaa, A., S. Olariu, S., L. Wilson, L., M. Eltoweissy, M. and K. Jones K..(2005) 'Training a wireless sensor network', *Mobile networks and Applications,* 10(1-2), 151-168, 2005.

[18] Bertossi, A. A., Olariu S., Pinotti M. C. (2008) 'Efficient corona training protocols for sensor networks', *Theoretical Computer Science*, 402(1): 2-15.

[19] Olariu, S., Wadaa, A., Wilson, L. and Eltoweissy, M. (2004) 'Wireless sensor networks: leveraging the virtual infrastructure'. *IEEE Network* 18(4): 51-56.

[20] Howard, A., Mataric, M.J. and Sukhatme, G. S. (2002) 'An Incremental Self-Deployment Algorithm for Mobile Sensor Networks', *Autonomous Robots, Special Issue on Intelligent Embedded Systems,* 13(2), pp. 113-126.

[21] Tamoghna, O., Manas, M. and Studip M..(2013) 'Tic-Tac-Toe-Arch: a self-organising virtual architecture for Underwater Sensor Networks'. *IET Wireless Sensor Systems,* vo. 3, Issue: 4, pp. 307-316.

[22] Capella, J. V., Bonastre, A., Serrano, J. J. and Ors, R. (2009) 'A New Robust, Energy-efficient and Scalable Wireless Sensor Networks Architecture Applied to a Wireless Fire Detection System', *International Conference on Wireless Networks and Information Systems*, WNIS '09. pp. 395 – 398.

[23] Miao, C., Dai, G., Zhao, X., Tang, Z. and Chen, Q (2014) '3D Self-Deployment Algorithm in Mobile Wireless Sensor Networks', In: Sun L., Ma H., Fang D., Niu J., Wang W. (eds) *Advances in Wireless Sensor Networks-CWSN 2014.* pp. 27-41.

[24] Chen, J., Qian, H. (2014) 'Three-dimensional deployment algorithm based on ideal fluid dynamics model for mobile sensor networks', *Computer Modelling & New Technologies* 18(9)12-18

[25] Huang, C-F., Tseng, Y-C. and Lo L-C. (2007) 'The coverage problem in three-dimensional wireless sensor networks', *Journal of Interconnection Networks*, 8, Issue 03, p. 209.

[26] Langendoen, K.. and Reijers, N. (2003). 'Distributed localization in wireless sensor networks: A quantitative comparison', *Computer Network*, 43(4), 499-518.

[27] Tanenbaum A. and Wetherall D. (2011) '*Computer Networks'*, Pearson Education, Inc/ Prentice Hall.

[28] WSNet, [online]. http://wsnet.gforge.inria.fr. (2016).

[29] A. Allirani, A. and M. Suganthi M.(2008) An energy sorting protocol (ESP) with low energy and low latency in wireless sensor networks. *International Journal of Computer Science and Network Security*, 8(11), pp. 208-214.

AUTHORS

Vianney Kengne Tchendji is a Senior Lecturer of Computer Science at the University of Dschang, Dschang, Cameroon. He received his PhD in Computer Science from the University of Picardie-Jules Verne, Amiens, France, in 2014. His current research interests include network virtualization, parallel algorithms and architectures, scheduling, wireless communication, ad hoc and sensor networking.

Laure Pauline Fotso is Professor of Computer Science at the University of Yaounde 1, Cameroon. She received her PhD in Computer Science from the Rensselaer Polytechnic Institute, Troy, New York in 1981. His current research interests include parallel algorithms, Multiple Objective Linear Programming, Optimization and Constraints Satisfaction Problems.

Jean Frédéric Myoupo is Professor of Computer Science at the University of Picardie-Jules Verne, Amiens, France. He was Dean of Faculty of Mathematics and Computer Science from 199 to 202. He received his PhD in Applied Mathematics from the Paul Sabatier University of Toulouse, France in 1983 and his Habilitation in Computer Science from the University of Paris 11, Orsay, France in 1994. He has been lecturer at the University of Sherbrooke, Canada, 1983-1985. Heis a past member of IEEE Computer, ACM and SIAM. His current research interests include parallel algorithms and architectures, wireless communication and mobile computing, ad hoc and sensor networking.

Ulrich Kenfack Zeukeng is a PhD student at the university of Dschang, Cameroon. He received his Master degree in Computer Science in 2015 from the university of Dschang. His current research interests include wireless communication and ad hoc networking.