Management of Low-density Sensor-Actuator Network in a Virtual Architecture

Vianney Kengne Tchendji*, Blaise Paho Nana*

*Department of Mathematics and Computer Science
Faculty of Science
University of Dschang
PO Box 67, Dschang-Cameroon
vianneykengne@yahoo.fr, blaisepaho@gmail.com

RÉSUMÉ. Les réseaux de capteurs sans fi (RCSF) font face à de nombreux problèmes dans leur mise en œuvre, notamment la connectivité des nœuds, la sécurité, l'économie d'énergie, la tolérence aux pannes, le routage [3]. Dans ce document, nous considérons un RCSF peu dense, caractérisé par une mauvaise couverture de la zone d'intérêt, et l'architecture virtuel introduite par Wadaa et al [1] qui permet de partitionner efficacemen ce type de réseau en clusters. Dans l'optique de router optimalement les informations collectés par chaque capteur jusqu'à une station de base (nœud sink, supposé au centre du réseau), nous proposons une stratégie de déplacement des capteurs mobiles (actuators) qui permet de: sauvegarder la connectivité du RCSF, optimiser le routage, économiser l'énergie des capteurs, améliorer la couverture de la zone d'intérêt, etc.

ABSTRACT. Wireless sensor networks (WSN) face many implementation's problems such as connectivity, security, energy saving, fault tolerance and routing problems [3]. In this paper, we consider a low-density WSN where the distribution of the sensors is poor, and the virtual architecture introduced by Wadaa et al [1] which provides a powerful and fast partitioning of the network into a set of clusters. In order to effectively route the information collected by each sensor node to the base station (sink node, located at the center of the network), we propose a strategy to allow mobile sensors (actuators) to move in order to: save connectivity of WSN, improve the routing of collected data, save energy of the sensors, improving the coverage of the interested area, etc.

MOTS-CLÉS : Réseau de capteurs sans fil architecture virtuelle, cluster vides, actuator, routage

KEYWORDS : Wireless sensor network, virtual architecture, empty cluster, actuator, routing

1. Introduction

For few years now, many improvements have been made in domains such as micro-electro-mechanical systems (MEMS) technology [9], wireless communications, and digital electronics. This enabled the development of micro components that easily combine data collection tools and wireless communication devices, and then opens a wide scope to wireless sensor networks (WSN) [3, 5, 8, 11].

Usually called microsensors or simply sensors, these devices with limited resources (bandwidth, computing power, available memory, embedded energy, etc.) have revolutionized traditional networks by bringing the idea to develop sensors networks based on the collaborative effort of a large number of sensors operating autonomously, and communicating with each other via short-range transmissions [6, 7]. These resource limitations added to the radio communication that have sensors, are factors that raise many problems (interference, intrusion, disconnection, data integrity, etc.).
In fact, it is common to see WSN composed of several thousand units [4]. In large networks, the sensors can be grouped into clusters based on their proximity in order to significantly increase the scalability, economy energy, routing, and consequently the lifetime of the network. The structure provided by this partitioning allows the use of various techniques to improve the quality of a WSN, such as data aggregation [10, 11].

In this paper, we consider a low-density WSN where the distribution of the sensors is poor, and the virtual architecture introduced by Wadaa et al [1] which provides a powerful and fast partitioning of the network into a set of clusters. In order to effectively route the information collected by each sensor node to the base station (sink node, located at the center of the network), we propose a strategy to allow mobile sensors (actuators) to move in order to : save connectivity of WSN, improve the routing of collected data, save energy of the sensors, improving the coverage of the interested area, etc.

The rest of this paper is organised as follows : we first present the virtual architecture in which we work, then we present a technique of detecting empty clusters, followed by our method of strengthening strategic points by the actuators, then the technique used to proper move the actuators is presented. A conclusion ends the paper.

2. Virtual architecture sensor network

2.1. Anatomy of a sensor

This is the basic equipment of any WSN. It has three main tasks : information collection from the deployment area, light treatment (optional) on the collected data and sharing these data with other sensors through multi-hop routing. Despite the great diversity (temperature sensors, humidity, pressure, etc.) existing on the market, they are all mounted on the same architectural diagram mainly made of a unit of : capture, processing, storage, communication, and energy. This material may be supplemented or reduced according to the developper [3]. One can for example add a locating system such as a GPS (Global Positioning System), a mobilizer (to get an actuator). The main and optional elements (represented by dashed lines) are shown in figure 1.

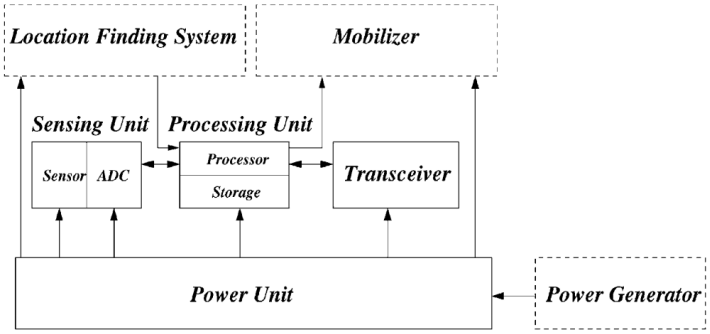
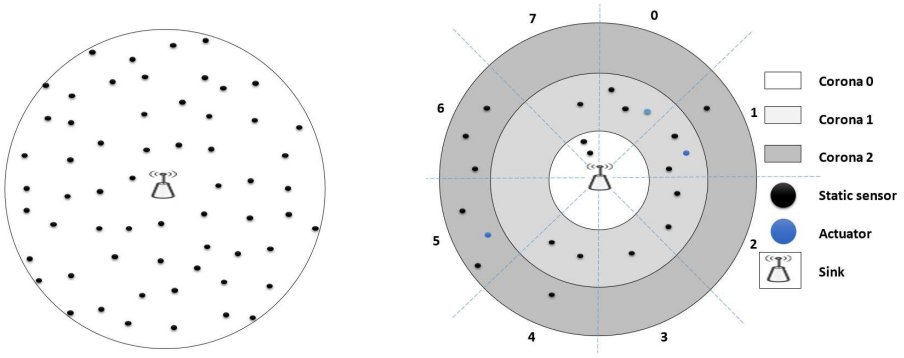


Figure 1 – Hardware architecture of a sensor.

2.2. Virtual network architecture

Let's consider a special sensor called the sink or base station (BS) unconstrained by common sensors's limits and capable of omnidirectional transmissions according to different radius and transmissions at various angles. Once deployed in the supervised area (figure 2a), the sensors can be grouped in clusters (as described in [1]) depending on the corona and the angular sector in which it is located (see figure 2b). Thus, the intersection of the corona i and the angular sector j forms the cluster (i, j). Since the network is sparse, it is important to identify the empty clusters. This allow to have an overview of the area covered by the sensors, and to achieve a better monitoring.



(a) Example of a basic WSN. (b) A sparse virtual architecture.

Figure 2 – An virtual architecture representation.

3. Detection of empty clusters and election of clusterheads

Knowing the distribution of sensors allows the sink to detect empty clusters and build the message propagation tree of figure 3b. For c coronas and s sectors, the sink counts $c \times s$ clusters in its virtual architecture ; therefore sink regularly updates to each message received two tables $h(c, s)$ and $relay(c, s)$. Each entry (i, j) of the table $h(c, s)$ contains 1 if cluster (i, j) is not empty and 0 otherwise, allowing the sink to get a global view of

the sensors's distribution ; and each entry (i, j) of the table $relay(c, s)$ contains the coordinates of the relay cluster of the cluster (i, j). For the rest we advance some hypotheses.

3.1. Hypotheses

The network is fully clustered as shown in figure 2b using the technique described in [1]. Each cluster has a gateway node or clusterhead (CH :node by which a message gets out of the cluster. This reduces redundancy.). Furthermore,

– Each sensor has a unique identifier ID in the network. The sensors are static and form a connected network. Adding or removing a sensor is a rare event ;

– A node is able to estimate its residual energy E_r and the sink has the ability to broadcast messages in the network at different radius, or at different angles ;

– The time is divided into slots of length r, parameters c and s are known to all sensors, and the local clock of each sensor is synchronized with the sink's ;

– A message sent by a sensor reaches all the sensors located in its transmission range after a slot ;

– Clusterisation is made such that all sensors of a given cluster can communicate with each other and some sensors of neighbor clusters.

Here is our protocol, its main lines are taken from S. Faye and J. F. Myoupo [2]. We are introducing the concept of clusterhead (CH).

3.2. Sink's algorithm

The sink periodically broadcasts the date on which the discovery algorithm will begin. All sensors are awake and sink initiates the detection by spreading in the first corona a discovery message $Detect(-1, -1)$. Due to network connectivity, it is certain that at least one sensor will receive this message. During the algorithm, each message transmitted by a sensor towards the sink contains the coordinates of its cluster and those of his relay cluster. Table $h(c, s)$ is initialized to 0. At each received messages from a sensor of the cluster (i, j), BS puts 1 in $h(i, j)$ and, in the entry $relay(i, j)$, it assigns the value contained in the variable *relay* of the received message. At the end of the algorithm, a cluster (i, j) is considered empty when $h(i, j) = 0$ and the relay cluster of the cluster (i, j) is indicated by $relay(i, j)$.

3.3. Sensors's algorithm

The network is supposed connected, so for all cluster (i, j) considered non-empty, there is always a path from it to the sink node. Isolated clusters can't reach the sink and are considered empty even if there are not. There are three main events in the detection of empty clusters : the reception of a message *Detect* asking sensors to indicate their coordinates ; the reception of a message *Head* sent by a node to propose itself as the gateway node and ; the reception of a message *ACK* sent by the sensors to the sink node to indicate their coordinates.

– **Reception of a message** $Detect$: On the first slot, after receiving a message $Detect(-1, -1)$ from BS, the sensors of the first corona build a message $Detect(1, j)$, and broadcast it to allow sensors in other clusters to reveal their presence. Then, those that received the message $Detect$ from a neighbor cluster and have an energy higher than the threshold E_s send a message $Head$ towards their own cluster to be elected as the CH.

– **Reception of a message** $Head$: At the reception of a message $Head$, the sensor saves the identifier of the CH in its variable $gatewayNode$ if this is its first reception, otherwise it compares the residual energy E_r of $gatewayNode$ with that of the received message and stores the one that has the biggest E_r. If the E_r are equal, the one with the highest ID is chosen. A sensor that has already received a message $Head$ can't send message $Head$, because it would have received its message $Detect$ from a more distant neighbor cluster to the sink than the relay sensor that sent him the first message $Head$. Finally, the sensor that is elected in the cluster builds a message $ACK((i,j),(-1,-1))$ and sends towards the sink node which is actually its relay cluster. This process is repeated for all the other sensors untill the most distant cluster sends its message ACK.

– **Reception of a message** ACK : A sensor that receives a message ACK from the neighbor node checks whether this message is for its cluster. In this case, it sends it to his gateway node which checks if it has not already routed a message from the same cluster. If not, it broadcasts it towards its relay cluster. Otherwise, it simply ignores it.

4. Searching and filling strategic empty clusters by actuators

This section is once again inspired from [2]. Here we introduce the actuators : sensors with a mobilizer, allowing them to move on the sink's order. They can be used for many purposes, depending on the user, for example :

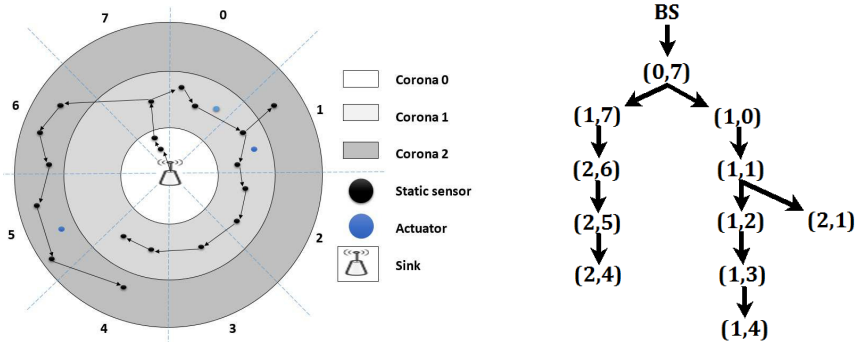
– Being the CH in a cluster where all the sensors have a small energy ;

– Collect and route information in isolated areas ;

– Connect a sub isolated connected network to the main network.

– Being sent in strategic empty clusters (purpose of this paper) to optimize the routing.

From the table $h(c,s)$ the sink knows the empty clusters. In order to know which ones it is going to fill first, it should reproduce the messages spreading tree like in figure 3b by using the two tables it has like this : Take BS as root and the first tree's leaf. As long as there's an unvisited leaf (i,j), search in the table $relay(c,s)$ the cluster that has the cluster (i,j) as relay cluster and add them as the sons of (i,j).



(a) Propagation of messages in the supervised area. (b) Messages spreading tree.

Figure 3 – The messages spreading tree obtained after the detection of the empty clusters.

Filling a strategic empty cluster has the effect of reducing the tree's height (figure 3b). The ideal one would be to reduce this size to the number of coronas of our virtual architecture. The routing will be optimal if for any cluster of corona k, the transmission of a message towards BS passes through k intermediate clusters.

4.1. Rule of detection of clusters which access can be improved

To optimize the routing to a cluster, it would be good to know whether the current access is optimizable. In figure 3, it is the case of cluster $(1, 4)$ which is at 5 intermediate clusters from BS instead of 1 if an actuator were placed in cluster $(0, 3)$.

Rule : Let $\mathbb{A}$ a message spreading tree similar to that of figure 3b, $prof(i, j)$ denotes the depth of the cluster (i, j) in the tree $\mathbb{A}$. The path from sink to (i, j) can be improved if there is another cluster (i', j') with depth $prof(i', j')$, such as $i' \geq i$ and $prof(i', j') < prof(i, j)$, i.e. (i', j') is in a corona greater or equal to (i, j)'s but in the tree $\mathbb{A}$, (i', j') appears at a depth less than (i, j)'s.

4.2. Detection of strategic empty cluster to fill in priority

To determine this priority cluster (PC), we establish for every corona a the list $L[a]$ of clusters of this corona which access can be improved (C coronas = C lists). Each list $L[a]$ contains the coordinates (a, j) of clusters of the corona a which access can be improved. It is in the form : $L[a] = [(a, j_1), (a, j_2), \ldots, (a, j_n)]$. From each list $L[a]$, *we extract the longest list* $L_s[a]$ *made of consecutive clusters of* $L[a]$. In the list $L_s[a]$, each (a, j) represents the coordinates of the clusters of corona a that follows in the message spreading tree. The coordinates (x, y) of PC are deduced from the longest sub list $L_{sp}[a]$ taken among the $L_s[a]$ extracted lists. $x = a - 1$ and y is equals to the default rounding average of j (the j are the second components items (a, j) of the list $L_{sp}[a]$).
As long as there's available actuators, it is necessary to move an actuator at the cluster (x, y), another at the cluster $(x - 1, y)$ if it is empty, ... another at the cluster $(0, y)$ if it is empty. The process can be repeated until the routing is optimal, i.e. up to $prof\mathbb{A} \leq C$.
For the example of figure 3, the determination of PC is presented in annex A.

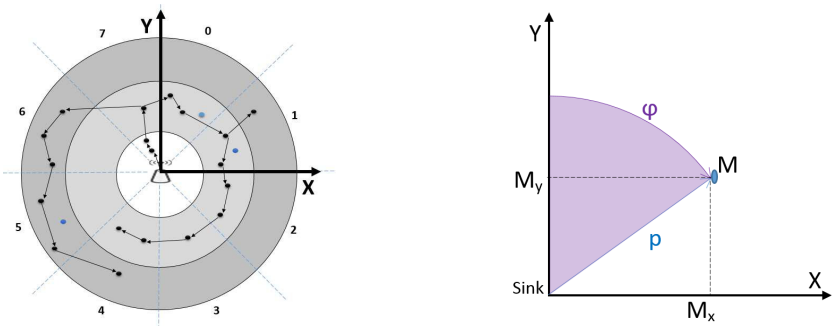
Before moving a mobile sensors, the sink node must calculate the distances from it to the target empty cluster, it is better to choose the most appropriate actuator, based on distance, residual energy, availability, etc.

5. Moving a mobile sensor

We propose to move an actuator from the cluster (x, y) to the empty cluster (j, k). The actuator will need the distance and direction to move. To simplify our calculations, we should be in a cartesian plane. For this we describe here how to transform our current coordinate system (Dynamic Coordinate System : DCS) in a Polar Coordinate System (PCS) and then in to a Cartesian Coordinate System (CCS).

5.1. Correspondence between DCS, PCS and CCS

To get the distance and the direction to follow, we must define a reference. Thus the origin of our reference will the sink. The Y axis is taken such that it coincides with the left edge of the first section (section 0). The X axis is at a quarter turn from the Y axis so that the angle $X, \widehat{sink}, Y$ is direct (figure 4a). Any point of the DCS is discoverable using p (its distance from the sink) and φ (angle measured from the Y axis) as in figure 4b.



(a) Definitio of the reference (b) PCS - CCS

Figure 4 – Correspondence between the DCS - PCS - CCS

Denote 'α' the angle of a sector and 'e' a thickness of a corona, we can state corollary 1.

Corollary 1 *Let M be a sensor of the cluster (c, s) assumed at it center. In the PCS, M has the coordinates (p, φ) where $p = c \times e + \frac{e}{2}$ and $\varphi = s \times \alpha + \frac{\alpha}{2}$;*
Let $M(p, \varphi)$ be a point in the PCS. In the CCS, M has coordinates (x, y) with $x = M_x = p\sin(\varphi)$ and $y = M_y = p\cos(\varphi)$.

5.2. Moving the actuator

Now we can start the necessary calculations (distance p and angles φ) to move the actuators.

5.2.1. Calculation of the distance (p)

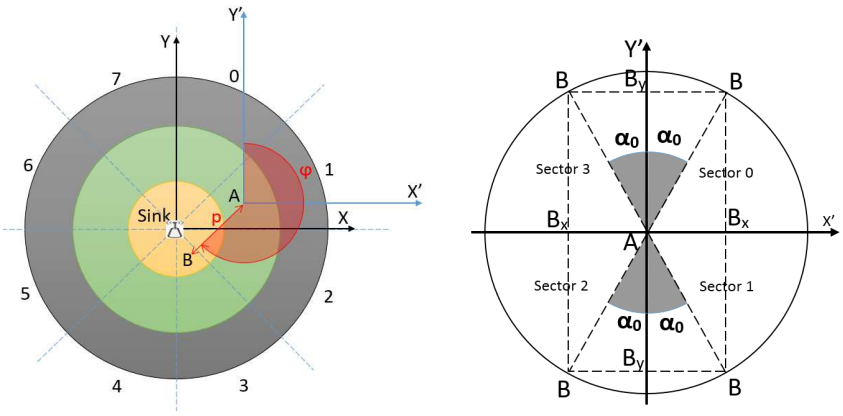
The distance between two points A and B of the plane is given by the norm of the vector $\overrightarrow{AB}$, denoted $||\overrightarrow{AB}||$ or just AB. According to figure 4b, an actuator that moves from the sink node (with cordinates $(0, 0)$) to the point M (with cordinates (x, y)) must cover the distance $p = ||\overrightarrow{sink\ M}|| = ||(x - 0, y - 0)|| = ||(x, y)||$. But $||\overrightarrow{sink\ M}||^2 = (sink\ M_x)^2 + (sink\ M_y)^2 = x^2 + y^2$. Thus $||\overrightarrow{sink\ M}|| = \sqrt{x^2 + y^2}$. So we have corollary 2.

Corollary 2 *Moving an actuator from the center of the cluster $A(c_1, s_1)$ of polar coordinates (p_1, φ_1) to the center of the cluster $B(c_2, s_2)$ of polar coordinates (p_2, φ_2) returns to move this actuator from the point $A(A_x, A_y)$ to the point $B(B_x, B_y)$ of the cartesian coordinates system on a distance $p = ||\overrightarrow{AB}||$ where $\overrightarrow{AB}(B_x - A_x, B_y - A_y)$, $p = ||\overrightarrow{AB}|| = \sqrt[2]{(B_x - A_x)^2 + (B_y - A_y)^2}$, $A_x = p_1\cos(\varphi_1)$, $A_y = p_1\sin(\varphi_1)$, $B_x = p_2\cos(\varphi_2)$ and $B_y = p_2\sin(\varphi_2)$.*

5.2.2. Calculation of the angle (φ)

To facilitate the calculation of the value of φ, let's make a change of reference.

Change of reference : We want to move a mobile sensor from a point A to a point B. For this, we define a new reference in which the base vectors are collinear with those of the previous (see figure 5a).



(a) Reference change

(b) Angle between the axis Y' and the displacement vector

Figure 5 – Angle displacement

The reference $R_1 = (sink, \vec{i}, \vec{j})$; $R_2 = (A, \vec{i}, \vec{j})$; $R_2 = t_{\overrightarrow{sink\ A}}(R_1)$ where t denotes the translation of vector $\overrightarrow{sink\ A}$. The coordinates of two points $M(x, y) \in R_1$, and $M'(x', y') \in R_2$ such that $M' = t_{\overrightarrow{sink\ A}}(M)$ are now linked by the following relations : $x' = x - x_A$ and $y' = y - y_A$.
Basis vectors of these two referenecs are pairwise collinear, that's why the angles found in one of the refeces will be equivalent in the second.

Calculation of the inclination α_0 ***formed by the displacement vector and the*** Y' ***axis :*** The displacement angle φ that we want to calculate is strongly related to the angle α_0 formed by the vector $\overrightarrow{AB}$ and the Y' axis. Figure 5b presents the different situations we may encounter. We deduced that $\sin(\alpha_0) = \frac{|B_x|}{AB} \Rightarrow \alpha_0 = \sin^{-1}\left(\frac{|B_x|}{p}\right)$ where $p = AB$.

Determination of the displacement angle φ ***:*** From figure 5b, point B can be found in one of the four sectors.

Corollary 3 *The displacement angle* φ *of an actuator from the point* $A(A_x, A_y)$ *to the point* $B(B_x, B_y)$ *of* $(A, \vec{i}, \vec{j})$ *reference is given by :*

1) if B is in the sector 0*, i.e.* $B_x > 0$ *and* $B_y \geq 0$ *then* $\varphi = \alpha_0$

2) if B is in the sector 1*, i.e.* $B_x \geq 0$ *and* $B_y < 0$ *then* $\varphi = \pi - \alpha_0$

3) if B is in the sector 2*, i.e.* $B_x < 0$ *and* $B_y \leq 0$ *then* $\varphi = \pi + \alpha_0$

4) if B is in the sector 3*, i.e.* $B_x \leq 0$ *and* $B_y > 0$ *then* $\varphi = 2\pi - \alpha_0$

For a practical example of moving a sensor, see annex B.

6. Simulation and analysis of our solution

6.1. Tools and simulation environment

Using an HP computer Intel (R) Core (TM) i7-2630QM CPU @ 2.00 GHz × 8, 8GB of RAM, running Windows 8 Professional ; a discrete event network simulator J-Sim ; and a sample of 1000 sensors randomly deployed within 10 km of the sink ; the virtual architecture has 10 coronas and 8 sectors of 45^o each. We performed repeatedly tests and averages the results. The energy model is the one adopted by many efficient contributions [13] : $E = E_{trans} + E_{recep}$. E_{trans} and E_{recep} are respectively the total energy used for transmissions in the network and receptions, knowing that each sensor has a range of 500 meters, and initial energy of 100 joules. He needs $35.28 \times 10^{-3} joule$ per transmission and $31.32 \times 10^{-3} joule$ per reception. The curves were made with version 5.0 of gnuplot software.

6.2. Analysis of the simulation results

Figure 6 compares the energy consumption of the cluster-heads and ordinary sensors when the routing is not optimized and when routing is optimized with actuators. An economy of energy is observed among both ordinary sensors and cluster-heads sensors. This increases the longevity of the network. The simulation is made for the detection of empty clusters, the election of cluster-head and routing. A slot is $78\mu s$. The great loss of energy observed at the beginning of the curve is due to the fact that the cluster-head are not elected at the beginning. It is clear that this energy loss is significantly reduced once the cluster-head are elected.

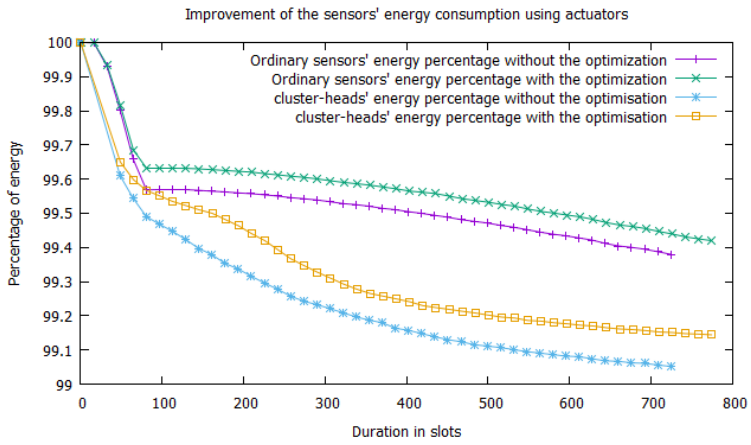


Figure 6 – Improvement of the power consumption using cluster-heads and actuators.

7. Conclusion

In this paper we have presented a virtual architecture that facilitates the management of WSN. We also introduced the gateways nodes, their election protocol and how to limit redundant messages through them. But our main aim was to optimize the routing of

collected data towards de sink. That's why we started by describing a method of detecting and filling strategic empty clusters in which we can send mobile sensors. We also present other possible utilities of the actuators, and show a new way of performing their movements to improve the routing of the collected data.

In a very soon future we intend to work on the mechanism of reelection of the clusterhead in a cluster ; the mechanism of changing relay cluster if the current relay cluster is no longer accessible and the mechanism of redirection of packets routing after the positioning of the mobile sensor.

8. Bibliographie

[1] A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, K. Jones, « Training a wireless sensor network », *Mobile Networks and Applications*, Vol. 10, N° 1-2, p. 151–168, 2005.

[2] Sébastien Faye, Jean-Frédéric Myoupo, « Deployment and Management of Sparse Sensor-Actuator Network in a Virtual Architecture », *International Journal of Advanced Computer Science*, Vol. 2, N° 12, December, 2012.

[3] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, E. l. Cayirci, « A survey on sensor networks », *IEEE Communications Magazine*, Vol. 40, N° 8, p. 102–114, 2002.

[4] B. Warneke, M. Last, B. Leibowitz and K. Pister, « Smart Dust : communicating with a cubic-millimeter computer », *IEEE Computer*, Vol. 34, N° 1, p. 44–51, 2001.

[5] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelman, « A taxonomy of wireless micro-sensor network models », *Mobile Computing and Communications Review*, Vol. 6, N° 2, p. 28–36, 2002.

[6] Mark A. Perillo, Wendi B. Heinzelman, « Wireless Sensor Network Protocols », *Algorithms and Protocols for Wireless and Mobile Networks, Eds. A. Boukerche et al., CRC Hall Publishers*, 2004.

[7] K. Sohrabi, J. Gao, V. Ailawadhi, G. J. Pottie, « Protocols for Self-Organization of a Wireless Sensor Network », *IEEE personal communications*, Vol. 7, N° 5, p. 16–27, 2000.

[8] C. Intanagonwiwat and R. Govindan and D. Estrin, « Directed Diffusion : a scalable and robust communication paradigm for sensor networks », *ACM Press*, p. 56–67, 2000.

[9] B. Warneke, K.S.J. Pister, « MEMS for Distributed Wireless Sensor Networks », *9th International Conference on Electronics, Circuits and Systems, Croatia*, Vol. 1, p. 291–294, 2002.

[10] S. Faye, J. F. Myoupo, « An Ultra Hierarchical Clustering-Based Secure Aggregation Protocol for Wireless Sensor Networks », *AISS : Advances in Information Sciences and Service Sciences*, Vol. 3, N° 9, p. 309 – 319, 2011.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, « SPINS : Security protocols for sensor networks », *Wireless networks*, Vol. 8, N° 5, p. 521–534, 2002.

[12] C. Karlof, N. Sastry and D. Wagner, « TinySec : A Link Layer Security Architecture for Wireless Sensor Networks », *in : Proc. of the 2nd international conference on Embedded networked sensor systems*, ACM, p. 162–175, 2004.

[13] D. Wei and S. Kaplan and H. A. Chan, « Energy Efficient Clustering Algorithms for Wireless », *in : Sensor Networks, Proceedings of IEEE Conference on Communications, Beijing*, IEEE, p. 236–240, 2008.

A. Practical example for determination of priority clusters

With the example of figure 3, the determination of priority clusters is performed as follow :
Lists construction :
$L[0] = \emptyset$; $L[1] = [(1,1),(1,2),(1,3),(1,4)]$; $L[2] = [(2,1),(2,4),(2,5)]$
Extraction of clusters sublists which follow :
$L_s[0] = \emptyset$; $L_s[1] = [(1,1),(1,2),(1,3),(1,4)]$; $L_s[2] = [(2,4),(2,5)]$
The longest sub-list : $L_{sp} = L_s[1] = [(1,1),(1,2),(1,3),(1,4)]$
Calculation of the coordinates (x, y) :
$x = 1 - 1 = 0$ and $y = floor(Average(1,2,3,4)) = floor(2.5) = 2$
Filling stategic clusters : The cluster $(0, 2)$ is free, an actuator should be sent into it. By repeating the process : We should put an actuator (if there are available) in the cluster $(0, 4)$ ...

B. Practical example of moving a mobile sensor

Let's move an actuator from the cluster $A(1, 1)$ to the cluster $B(0, 3)$. For our tests, let's suppose : the scope of the sink is $30.0m$ and the virtual architecture includes 3 coronas with $e = 10.0m$ each ; there are 8 angular sectors of $\alpha = \frac{\pi}{4}$ rad each.

Polar Coordinates : A(15.0, 1178 rad) and B(5.0, 2.749 rad)

Cartesian coordinates : $A(13858, 5740)$ and $B(1913, 4619)$

Distance : $p = ||\overrightarrow{AB}|| = 15.811$

Displacement angle : $B'_x = B_x - A_x = -11.945$ and $B'_y = B_y - A_y = -10.359$, then $\alpha_0 = 0.856$ rad. Since $B'_x < 0$ and $B'_y <= 0$ then B is in sector 2 and thus, $\varphi = \pi + \alpha_0 = 3.998$ rad or $\varphi = 229.065^{\circ}$.

Conclusion : To strengthen the area $(0, 3)$, the sink node asks the actuator of the cluster $(1, 1)$ to cover a distance $p = 15.811m$ with an angle of $\varphi = 229,065^{\circ}$.