Deployment and Management of Sparse Sensor-Actuator Network in a Virtual Architecture

Sébastien Faye & Jean-Frédéric Myoupo

Manuscript
Received:
9,Jun., 2012
Revised:
22,Jul., 2012
Accepted:
1,Oct., 2012
Published:
20,Dec., 2012

Keywords
Wireless Sensor-Actuator Networks, Virtual Architecture.

***Abstract*— The use of a virtual architecture in wireless sensor networks provides a powerful and fast partitioning into a set of clusters, each being locatable in space by a base station, conductor of the network. In this extented paper of previous works, we study the case of a low-density network, where the distribution of the sensors is poor. We first propose a protocol for the detection of empty clusters. Then, we propose a strategy to allow mobile sensors (actuators) to move to empty clusters, in order to improve the routing of collected data, by detecting priority empty clusters and by providing the possibility to physically move sensors. Finally, we show by simulations the necessity of considering empty clusters.**

1. Introduction

The wireless sensor networks (WSNs) are from the family of mobile ad-hoc (MANET), but have additional features and constraints: typically, they consist of a wide range of sensors with limited energy capacity. Each sensor is powered from a non-rechargeable and non-replaceable battery ([1]) and has a low capacity in terms of memory, computation (CPU) and wireless transmission range. Each sensor is able to harvest a set of data in a certain environment, and transmit it in multi-hop way to a base station (denoted BS, also called sink), which may act as instructor of the network. The use of such networks is widespread in many applications: for example the monitoring of forests or critical infrastructure, the detection of biochemical agents or the road traffic control. Some examples of works can be found through [1-5].

Technology related to sensors advancing day by day, it is common to see WSNs composed of several thousand units ([6-7]). 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 ([8-9]).

In this paper, we focus on the concept of virtual architecture developed by Wadaa et al. in [10]. The latter is created and orchestrated by BS, which is able to split the network into a set of clusters, depending on the strength and direction of the broadcast it can perform. We focus on the case where the network is sparse, leaving many empty clusters formed in the virtual architecture, contrary in [10] where the WSN is assumed to be dense. Once detection has occurred, we introduce the notion of actuators (sensors that have the ability to move) to consider a positioning strategy onto empty clusters. It is important to note that this paper is an extension of previous works [3]: especially, it included additional simulations, pseudo-code algorithms and more informations about our approach.

The rest of the paper is organized as follows: section 2 presents the model of our network and specifically the virtual architecture; the protocol for the detection of empty clusters is described in section 3; the strategy of feeding the empty clusters by the actuators is presented in section 4; section 5 describes how to move the actuators onto the empty clusters; finally, the simulation results are presented in section 6 and show the necessity of considering empty clusters.

2. Model

Consider a set of anonymous sensors distributed around a master node BS. Figure 1 illustrates an example of such a network. In this WSN, each sensor has the ability to collect data, perform calculations, store data temporarily, and to transmit and receive informations within a certain range R that does not cover the entire network because of energy constraints and costs. The transmission of information from one sensor to BS is therefore usually done in multi-hop. BS has the opportunity to transmit information to some powers (to the sensor farther away).

The virtual architecture proposed in [10] consists of a partition of the WSN into different zones by BS. It has the opportunity to disseminate informations from the lower to the higher range in order to create coronas. It has also the opportunity to disseminate informations in certain directions as [10] to create different angular sectors. The area (i, j) is the intersection of the corona i with the angular sector j. The sensors of the same area are therefore in the same geographical location and form a cluster. This is illustrated in figure 2. Other works on this type of architecture can be found in [11-13].

On this basis, the authors in [10] define some mechanisms, such as data aggregation, routing and

Université de Picardie-Jules Verne, UFR Sciences, 33 rue Saint Leu, 80039 Amiens France (fayee@telecom-paristech.fr, jean.frederic.myoupo@u-picardie.fr).

generally training coordinates. We will not detail these operations here. Interested readers can find them in the original article [10]. Here we focus only on sparse WSN in which some clusters can be empty. In such networks the routing is optimal, and BS is not able to detect innately these empty clusters.

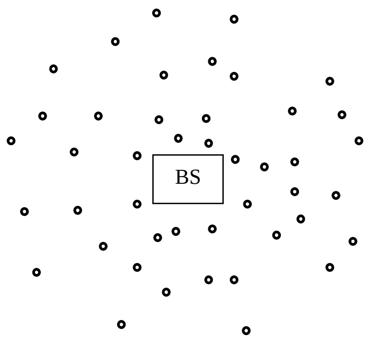


Fig 1. Example of a basic WSN

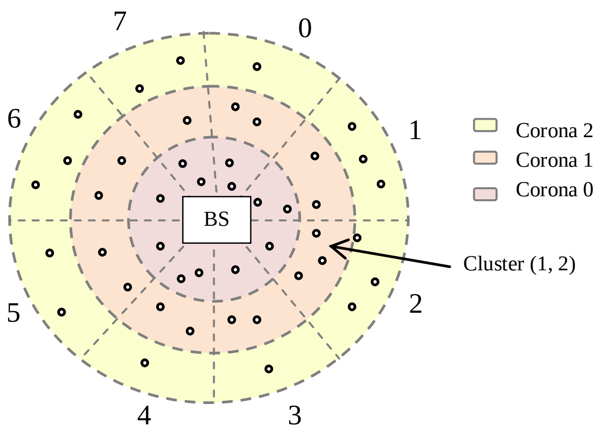


Fig. 2. An virtual architecture representation.

Here, we propose to equip BS with a protocol to detect these empty clusters. Note that the operations described here and below can be done in a secure way with, for example, µTESLA [14]. As an example, in previous works [8], we propose a secure aggregation protocol based on virtual architecture.

3. Detection of empty clusters

A. Assumptions

In addition to classical static sensors, we introduce in our network some actuators (i.e. mobile nodes), denoted ACT. The set of sensors-actuators is denoted WSN*. In this paper, we assume that the WSN is clustered as in [10] as in figure 3 below. Initially, BS affects different numbers of coronas and angular sectors to WSN* (based on a variable diffusion range and unidirectional broadcast, see 2.). This partitioning, as our contribution, requires slot times s_x ($x \geq 0$) to allow all nodes to be synchronised. Finally, the network is partitioned into a set of clusters: each cluster of corona i and angular sector j is called a zone (i,j).

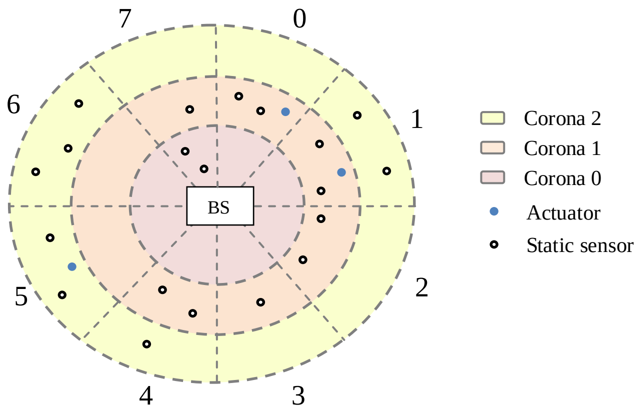


Fig 3. A sparse virtual architecture.

Synthetically, we consider the following assumptions:

1) We assume that the network of WSN*\{ACT} is static, composed of anonymous immobile nodes once deployed. Optionally, adding and deleting nodes (for fault tolerance) is allowed, but considered rare. In contrast, ACT represents mobile nodes, which have an identity, and whose main objective is to improve the connectivity of the network.
2) The network topology is known to any sensor, but the density of nodes allows us to assume that WSN*\{ACT} is connected, as well as all WSN*.
3) BS has the opportunity to disseminate informations across the network more or less powerfully (i.e. at different ranges) and at different angular sectors (i.e. one-way broadcast).
4) Time is divided into a set of slots s1, s2, ... sn and the local clock of each sensor is synchronized with the one of BS. In this way, each sensor is awake and listening to the network for each of these slots, but is put to sleep the remaining time to save the overall network energy.
5) We consider that BS and the actuators have a notion similar direction: it is necessary to allow BS to manage the movements of actuators.

B. Description of the detection protocol

1) ***Phase 1:*** *Initialization of the detection by BS.* In the first slot s0 all sensors-actuators are awake. BS initiates the construction of the breadth search tree. For this, it broadcasts at range R_{bs} (corresponding to the first corona of the virtual architecture) a discovery message noted Build(zone) to all the sensors WSN*. WSN being connected, we are assured that at least one sensor receives the information. Here, Build(zone) = Build((-1, -1)), where (-1, -1) represents the area consisting solely of BS.

2) ***Phase 2 (recursive):*** *construction the breadth search* *tree*. This phase is repeated as s0 is not exceeded, and has 4 steps:

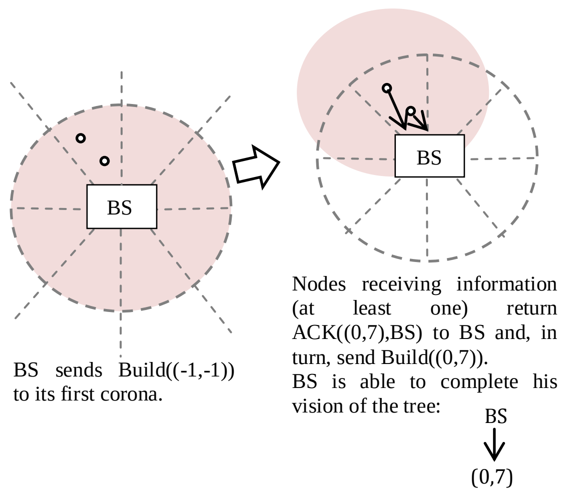
Step 1. All actuators that had no heard any message of construction during s0, when receiving the message Build(zone) they simply return a receipt to BS, stating their position for future movements.

Step 2. When receiving the message Build(zone), all static sensors that have never received a message of construction during s0 are able to temporarily store Build.zone, corresponding to the cluster where the sender sensor is located (father's sensor).

Step 3. When receiving this information, static sensors then perform two actions:

a. If their cluster is different from the one of the message of construction received, then send an acknowledgment backwards to BS, directly or in multi-hop through the cluster Build.zone. This message looks like ACK(zone, parent). zone representing the cluster from which the acknowledgment is sent, and parent the information Build.zone registered by the sensor. In receiving this message, BS is able to complete his vision of the tree width.

b. Continue to spread the message of tree construction by sending the message Build(zone) at a transmission range R (where R is the transmission range of a sensor, which we consider to be homogeneous).



BS sends Build((-1,-1)) to its first corona.

Nodes receiving information (at least one) return ACK((0,7),BS) to BS and, in turn, send Build((0,7)).
BS is able to complete his vision of the tree:

Fig. 4. Illustration of the first two phases of the detection protocol.

These first two phases can be represented through the figure 4. When these are completed and slot s0 is expired, BS has an overview of the network and the tree width of the zones of virtual architecture installed on it, like the example shown in figure 5 and figure 6. We present a simplified distributed algorithm of these two phases in algorithm 1 and summarize the notations.

C: number of virtual architecture coronas.
A: number of virtual architecture angular sectors.
zone: pair of integers (i, j) – refers to the area formed by the intersection of the corona i (zone.corona) and the angular sector j (zone.sector). (-1,-1) corresponds to the initial zone, BS.
R_{BS}: BS transmission range, at least egual to the scope of the first corona.
R: sensor transmission range.
slot: one slot time, in seconds.
Tree[]: A*C sized array of integer lists.
ACT[]: |ACT| sized array which contains actuators positions.
exists(zone): returns true if the custer « zone » exists in Tree[].
send(R, M): sends a message M of range R.
number_of(zone): returns a unique integer, corresponds to the lexicographic order of one zone (i, j) among all zones. For exemple, number_of((-1,-1)) = 0, number_of((0,0)) = 1, etc.
zone_of(number): inverse function of number_of(). For exemple, zone_of(2) = (0,1).
ACK(zone, parent): message with a zone and its parent node zone.
Build(zone): message with a zone.
time(): local time, synchronised with all network members.
consecutive_integers_of(L): the largest suite of consecutive integers in the list of integers L.
initial_time: integer. **parent**: zone.

<u>Algorithm of BS, without take into account actuators.</u>

```
Initially:
send(R_BS, (-1,-1))
initial_time = time()

At the reception of ACK(zone, parent) in cluster « zone »:
if(time() < initial_time + slot) then
tree[number_of(parent)] = number_of(zone) U tree[parent]
```

<u>Algorithms of all sensors (є *\{ACT}).</u>

```
Initially:
parent = NULL
initial_time = time()

At the reception of Build(zone) in cluster « zone »:
if(parent == NULL AND time() < initial_time + slot) then
    parent = Build.zone
    if(parent != zone) then send(R, ACK(zone, parent))
        send(R, Build.zone)

At the reception of ack(zone, parent):
if(time() < initial_time + slot) then
send(R, ACK(ACK.zone, ACK.parent))
```

Algorithm 1.

3) ***Phase 3:*** *Detection of empty clusters.* BS being informed of all clusters of the virtual architecture of the network (he is the designer), he is able to detect empty clusters i simply browsing the generated tree.

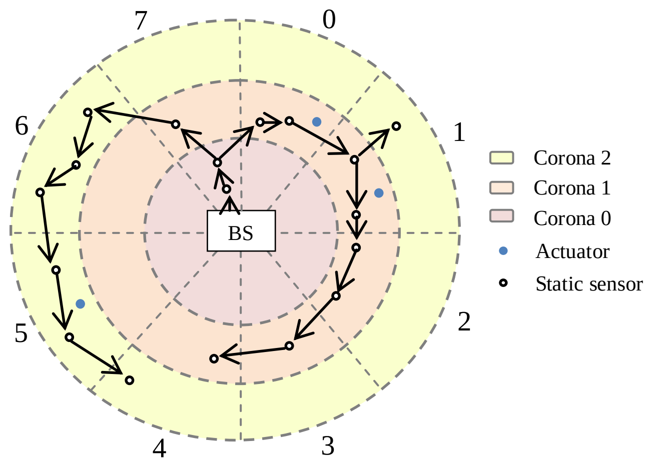


Fig. 5. Construction of the breadth search tree.

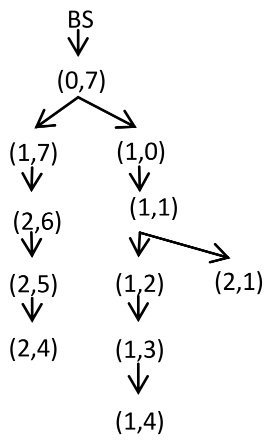


Fig. 6. The breadth search tree.

4. Strategy of feeding the empty clusters by the actuators

The empty clusters are now known to BS. We detail a strategy that could allow BS to place actuators on these empty clusters in order to gather and route eventual data collected on these clusters. Of course, this strategy is just a sample application: the user can choose to apply a priority policy, by choosing for example a part of the actuators for the collection of information in critical areas that are not equipped with stationary sensors.

Our approach consists of two main phases: the first improves the tree width obtained above, in order to reduce the number of hops required for certain nodes to communicate with BS. Second, we may use the remaining actuators to improve communication in some areas with small populations.

1) ***Phase 1:*** *Improvement of the breadth search tree.* Let H be the height of the graph obtained in section 3. Consider C the number of coronas used by the virtual architecture, plus one (example in figure 6: we have H = 7 and C = 4). The purpose of the first phase is to improve the value of H, reducing it until it reaches, if possible, H = C (requires a sufficient number of actuators). Here, we apply a strategy of feeding the empty cluster in terms of the number of actuators available and known by BS. At this point, two cases arise:
Step 1. If at the start we have $H \leq C$, then the application of this phase is not necessary because the routing is optimal: we can directly proceed to Phase 2.
Step 2. Otherwise, we have to determine the priority empty clusters among all the empty clusters of our virtual architecture. The priority empty clusters are those that we must first feed (by an actuator) in order to improve the height H of the tree. To do this, BS calculates from the tree the C - 1 lists: each list number i denoted L[i] contains the list of angular sectors $j \in L[i]$ corresponding to clusters (i, j) whose access can be improved. A cluster (i, j) to which access can be improved (by positioning correctly an actuator) is such that i+1 is strictly below the level of (i, j) in the tree. This is illustrated through figure 7.

2) ***Phase 2:*** *Network consolidation.* This phase starts when the first phase is complete, and only if there are available actuators. This optional phase is left to the will of the final user, and may depend on multiple strategies. BS have a global view of the network, it can implement several solutions. For example:
1. Ability to use the remaining actuators to collect informations in some non-accessible zones.
2. Ability to use actuators to solidify zones with a low density of actuators, in order to improve the connectivity and reactivity of the WSN.

From this list, we can easily deduce the priority empty clusters: consider the cluster (i, j) such that $j \in L[i]$, *the priority empty clusters are the empty clusters (m, j) with m<i*. This is illustrated through figure 8. Algorithm 2 present the algorithm of this first phase.

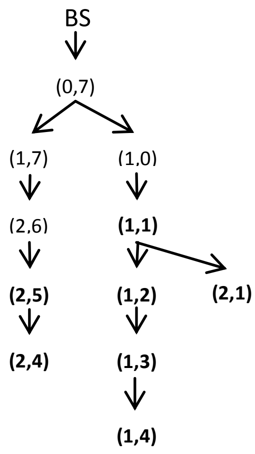
Here, we propose to place actuators in priority empty clusters in order to improve the routing and connectivity between sensors. We assume that one actuator can cover an entire cluster. Of course, this is only a proposition: final user can choose to place |ACT| available actuators at their convenience. We identified two cases:

Case 1. If the number of priority empty clusters is lower than the number of actuators, then each zone can have at least one actuator.

Case 2. Otherwise, BS have to find the best locations for each actuator. We have to compute the greatest set of consecutive priority empty clusters owned by one corona, and divide this set by placing an actuator in the center, then repeat operation until there is no actuators. If there are several equal sets, we select the set with smaller corona, next smaller angular sectors. We note the list L[i] the biggest set, and $u_1 \ldots u_n$ its largest suite of consecutive integers. We can compute u_{avg} = ROUND(AVG($u_1 \ldots u_n$)): one actuator have to go in zone (i-1, u_{avg}), one other to (i-2, u_{avg}), until (i-T, u_{avg}), with T =

i, or when there are no more actuator. Algorithm 3 show an example of implementation.

Example with figure 8: the largest set of consecutive zones is represented by L[1]: (1,1), (1,2), (1,3), (1,4). We have ROUND(AVG(1,2,3,4)) = 2: we need an actuator to go in zone (1-1,2) = (0,2). By repeating this operation, the largest group is (1,3), (1,4): we must assign an actuator to zone (0,3). For the last actuator available, we assign it to (1,4).



L[0] = Ø
L[1] = 1,2,3,4
L[2] = 1,4,5

Fig. 7. Detection of clusters whose access can be improved (in bold on the tree).

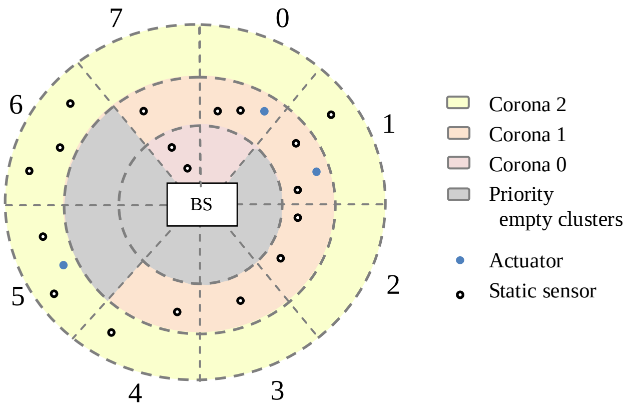


Fig 8. Detection of priority empty clusters.

Main variables and functions: see algorithm 1.

Determination of L[corona] lists, the zones where access can be improved.

```
for (w = 1 ; w ≤ A*C ; w++) do
    for (L in Tree[w]) do
        if (get_zone_of(w).corona = L.corona) then
        L[corona] = L[corona] U L.sector
```

Determination of priority empty clusters in a « Prior » list.

```
for (w = 1 ; w ≤ C ; w++) do
    for (U in L[w]) do
        for (k = 0; k ≤ X – 1; k++) do
            if( !exists(k, U) ) then Prior = Prior U (k, U)
```

Main variables **Algorithm 2.** **and functions:** see algorithm 1.

Case 2 of our solution:

```
cpt = 0
if(|Prior| ≤ |ACT|) then
    for (zone in Prior) do
        ACT[cpt] = zone
        cpt = cpt + 1
    else
        for all L[X] rand by (|L[X]| DESC, X ASC) do
            L = consecutive_integers_of(L[X])
            Position <- ROUND(AVG(L))
            L[X] <- L[X] \ {Position}
            for (k = 0 ; k ≤ X – 1; k++) do
                if( !exists(k, Position) ) then
                    ACT[cpt] = (k, Position)
                    cpt = cpt + 1
```

Algorithm 3.

5. Moving the actuators to the empty clusters

It is necessary to provide to an actuator in cluster (i, j), all informations that will help it to move correctly to the empty cluster (i2, j2). BS has an overview of the network and knows the parameters used by the virtual architecture. One of its roles is then to calculate and transmit the movements to be performed by each actuator, using simple formulas from trigonometry. Below, we detail the various calculations that BS must do to move an actuator from (i, j) to (i2, j2).

We assume that BS knows the angle α of an angular sector and the scope β of a corona. Let A be a virtual point that we consider to be the center of the cluster (i, j). Let B be a second virtual point considered the center of (i2, j2).

A. Computation of the distance Df between A and B

According to equation 1 and 2, distances from BS to A and B are compute. Let [BS,0) be the half straight line starting from BS and following the left end of the angular 0. The angle formed by [BS,0) and [BS,A] is described in equation 3. In the similar way, [BS,B] is described in equation 4. Therefore, the angle between [BS,A] and [BS,B] is shown in equation 5. Finally, the distance A to B is given by the equation 6.

$$A.D = b \times i + \frac{b}{2} \qquad \text{(Equ. 1)}$$

$$B.D = b \times i2 + \frac{b}{2} \qquad \text{(Equ. 2)}$$

$$A.R = \alpha \times j + \frac{\alpha}{2} \qquad \text{(Equ. 3)}$$

$$B.R = \alpha \times j2 + \frac{\alpha}{2} \qquad \text{(Equ. 4)}$$

$$R1 = MAX(A.R, B.R) - MIN(A.R, B.R) \quad \text{(Equ. 5)}$$

$$Df = \sqrt{A.D^2 + B.D^2 - 2A.D \times B.D \times \cos(R1)} \quad \text{(Equ. 6)}$$

B. Computation of the angle to send A in the direction of B

Let [A, 0) the half-line parallel to [BS, 0) but starting from A (assumption (5)). Here, we look for the angle between [A, 0) and [A, B], denoted by Rf. Here, consider only the horizontal axis: let [0, A] be the segment of length DA from A to the line [BS, 0), perpendicular to [BS, 0). Let [0, B] be a similar segment of length DB, in the case of point B:

$$DA = A.D \times \sin(\Delta A) \quad \text{(Equ. 7)}$$

$$DB = B.D \times \sin(\Delta B) \quad \text{(Equ. 8)}$$

Dx is the angle at x of the rectangular triangle thus formed (by BS, x, and the point to the segment [BS, 0]), taking the following values (Algorithm 4):

```
If x.R < 90 Then Δx = x.R
Else If x.R < 180 Then Δx = 180 - x.R
Else If x.R < 270 Then Δx = x.R - 180
Else Δx = 360 - x.R
```

Algorithm 4.

Let us consider now the rectangular triangle of hypotenuse [A, B]. Set Dtmp the length of its adjacent side having the end point A as (Algorithm 5):

```
# If an point is on the left side of BS and another on its right side
If (A.R > 180 and B.R < 180) or (A.R < 180 and B.R > 180)
then Dtmp = DA + DB
# If both A and B are on the same side de BS
Else Dtmp = MAX(DA,DB) – MIN(DA,DB)
```

Algorithm 5.

We can then easily calculate the angle between the adjacent side of [A, B], say R2:

$$R2 = \cos\left(\frac{Dtmp}{Df}\right) \quad \text{(Equ. 9)}$$

Rf is then determined by adjusting R2 depending on the position of A relative to B. Several cases are possible. These cases are summarized algorithmically as follows (Algorithm 6):

```
# A on the left BS and B on the right
If (A.R < 180 and B.R > 180) Then
    # A below and on the right of the line of B
    If (A.R > 360 – B.R) Then Rf = 270 + R2
    Else Rf = 180 + R2
# A on the right of the line BS, B on its left
Else If (B.R < 180 and A.R > 180) Then
    # A above and on the left of B
    If (B.R > 360 – A.R) Then Rf = 90 + R2
    Else Rf = 90 - R2
# A and B on the left side of BS
Else If (B.R < 180 and A.R < 180) Then
    # A on the left of B
    If (A.D > B.D) Then
        # A is above B
        If (A.R > B.R) Then Rf = 90 + R2
        Else Rf = 90 – R2
    # A on the right of B
    Else
        # A is below B
        If (B.R > A.R) Then Rf = 270 + R2
        Else Rf = 180 + R2
# A and B are on the right side of BS
Else
    # A on the left of B
    If (A.D < B.D) Then
        # A is above B
        If (A.R < B.R) Then Rf = 90 + R2
        Else Rf = 90 – R2
    # A is on the right of B
    Else
        # A is below B
        If (B.R < A.R) Then Rf = 270 + R2
        Else Rf = 180 + R2
```

Algorithm 6.

These calculations can be illustrated by the example in figure 9a and 9b, where an actuator placed in (1,1) must go to (0.3). BS sends an instruction to the actuator located at (i, j) to move on a distance Df with an angle Rf° to the north common with BS (assumption (5)).

Example: for α = 45° and β = 30m: A.R = 67.5, A.D = 45, B.R = 157.5, B.D = 15, DA ~= 41.5, DB ~= 5.74, Dtmp ~= 35.8, R2 ~=40°. This yields Df ~=47m and Rf ~=230°.

The actuator then moves on 47m following a path of 230° to the direction north that it shares with BS (assume that the start is the angular sector 0). Once the actuator placed, it sends an acknowledgment to BS containing the area where it is located. BS checks that everything coincides and regularly asks for the actuator's status and position, in order to guide it or replace it if something goes wrong. Conversely, the actuator can send alert to BS messages in case of problems.

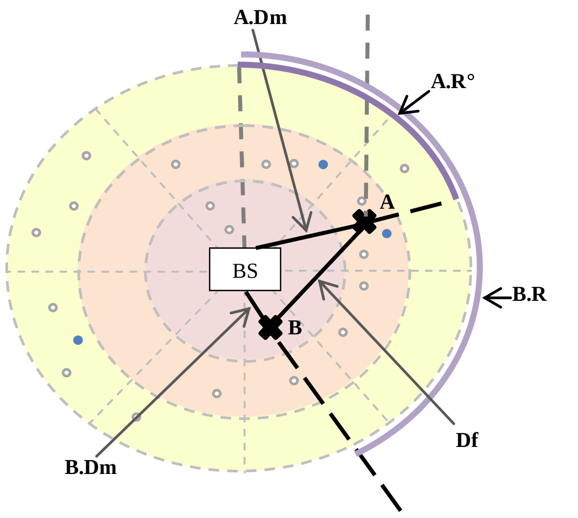


Fig 9.a. Illustration of variables and calculations used for the placement of actuators: calculation the distance between A and B.

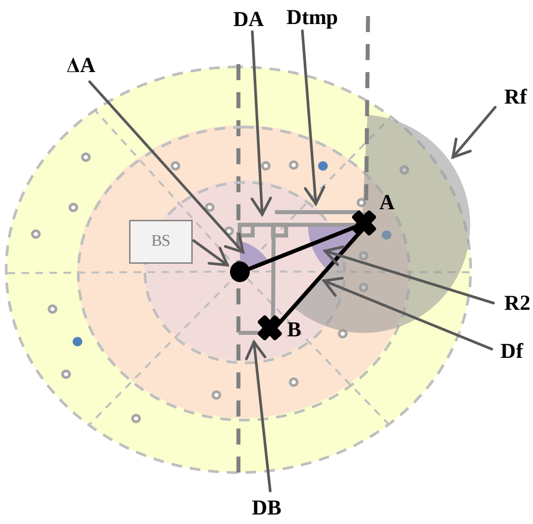


Figure 9.b. Illustration of variables and calculations used for the placement of actuators: calculation of the angle.

6. Simulation results

Here we focus on percentages of empty clusters that may be present in the virtual architecture described above. Our simulations were performed using the software WSNet [15] and are based on circular area of 1km, 4km and 9km in diameter, on which we randomly generate connected networks of 500, 1000, 1500, 2000 and 2500 sensors. We take here an ideal case, where BS is located at the center of this circular area. Figure 10 illustrates the simulations obtained from a virtual architecture with 6 coronas and an angular sector of 30°. Figure 11 illustrates the simulations results obtained from a virtual architecture with 4 coronas and an angular sector of 45°. Each point of the various curves is the average result of 10 simulations. By curiosity, we provide to interested readers the number of clusters, based on coronas and angular sectors capacities (figure 12).

As we can see on the curves below there are empty some clusters on virtual architecture considered especially in low-density network. If we take the example in figure 10 where the angular sectors are 30° and with 6 coronas, we see almost 80% of empty clusters in the case where 500 sensors are spread over a circular area of 9km in diameter. However, this percentage slows down to 60% in figure 10. So there is a need to manage the existence of empty clusters: the detection and the implementation of strategies with potential sensors and actuators. Furthermore, note that the simulations were performed under ideal circumstances: the area of sensors is circular, BS is at the center, and we do not take into account environmental constraints. It appears evident that these points are rarely met (BS can be at the network edge, rectangular area of sensors, ...).

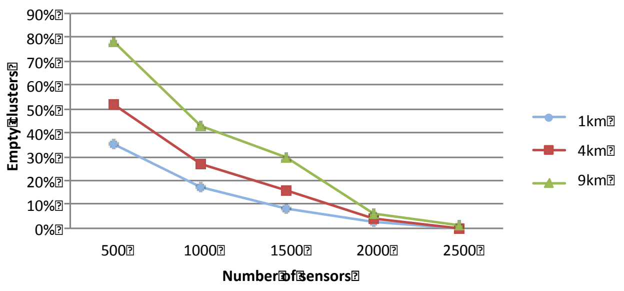


Fig. 10. Average percentage of the number of empty clusters based on the number of sensors, for 3 zones of different sizes. Here, we have 6 coronas and angular sectors of 30° each.

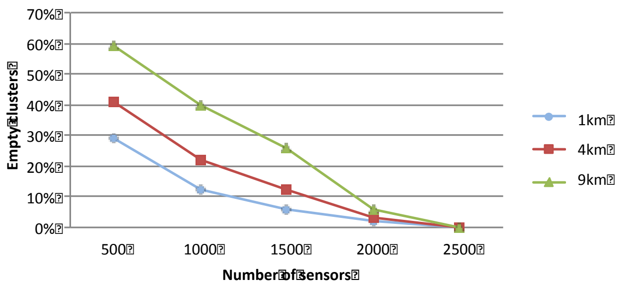


Fig. 11. Average percentage of the number of empty clusters based on the number of sensors, for 3 zones of different sizes. Here, we have 4 coronas and angular sectors of 45° each.

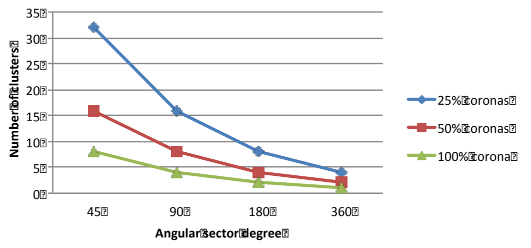


Fig. 12. Number of clusters based on angular sectors and coronas.

7. Conclusion

In this paper we have presented a protocol to identify the empty clusters in a virtual architecture of a WSN. Next, we have studied few strategies and methods to use actuators and move on these empty clusters. We have shown how actuators can be guided by BS to position themselves on desired empty clusters. However, we could have also assumed that each actuator is equipped with GPS that should facilitate the management of the movements. Similarly, we could have assumed that each actuation is equipped with APS [16-17] protocol.

In future work, it would be interesting to study such problems with more strong conditions: for example by removing the assumption of connectedness and reducing the density advantage of the network, we could explore the possibility of using the actuator nodes to allow some connected components to reach BS.

References

[1] I. F. Akyildiz, W. Su and Y. Sankarasubramaniam. Wireless sensor networks: a survey, Computer Networks (38), pp. 393-422, 2002.

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

[3] S. Faye, J. F. Myoupo, Deployment of Sparse Sensor-Actuator Network in a Virtual Architecture. The 2011 International Conference on Wireless Networks, ICWN'11: July 18-21, 2011, USA.

[4] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: Proc. MOBICOM'00, Boston, MA (August 2000).

[5] C.-C. Shen, C. Srisathapornphat and C. Jaikaeo, Sensor information networking architecture and applications, IEEE Personal Communications (October 2000) 16-27.

[6] J.M Kahn, R.H Katz and K.S.J. Pister, Mobile networking for Smart Dust, in: Proc. MOBICOM'99, Seattle, WA, August 17-19 (1999).

[7] B. Warneke, M. Last, B. Leibowitz and K. Pister, SmartDust: communicating with a cubic-millimeter computer, IEEE Computer 34(1) (2001) 44-51.

[8] 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, No. 9, pp. 309 ~ 319, 2011.

[9] C. Karlof, N. Sastry and D. Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. SenSys04, November 35, 2004.

[10] A. Wadaa , S. Olariu , L. Wilson , M. Eltoweissy, K. Jones, Training a wireless sensor network, Mobile Networks and Applications, v.10 n.1-2, p.151-168, 2005.

[11] F. Barsi, A.A. Bertossi, F. Betti Sorbelli, R. Ciotti, S. Olariu and M. C. Pinotti, Asynchronous Training in Wireless Sensor Networks. Proceedings of the 3rd international conference on Algorithmic aspects of wireless sensor networks, pp. 46-57, 2007

[12] A.A. Bertossi, S. Olariu, and M.C. Pinotti. Efficient Training of Sensor Networks. ALGOSENSORS, pp. 1-12, 2006

[13] S. Olariu, A. Wada, L. Wilson, and M. Eltoweissy, Wireless Sensor Networks: Leveraging the Virtual Infrastructure. IEEE Network, vol. 18,pp. 51-56, 2004.

[14] A. Perrig, R. Szewczyk, V. Wen, D. Cullar and J. D. Tygar. Spins: Security protocols for sensor networks, In Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 189-199, 2001.

[15] http://wsnet.gforge.inria.fr

[16] D. Niculescu and B. Nath, Ad hoc positioning system (APS), Proceedings of IEEE GLOBECOM, pp. 2926-2931, San Antonio,TX, November 2001.

[17] S. Faye, C. Chaudet, I. Demeure, "A Distributed Algorithm for Adaptive Traffic Lights Control", 15th International IEEE Annual Conference on Intelligent Transportation Systems, Anchorage, USA, September 2012.