# S-LEACH: A LEACH extension for Shared Sensor Networks

Gabriel Caldas, Claudio M. de Farias, Luci Pirmez, Flávia C. Delicato
PPGI
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil - 21941-901
{gcaldas08, cmicelifarias, luci.pirmez, fdelicato}@gmail.com
Corresponding author: Gabriel Caldas

***Abstract*— A new trend on Wireless Sensor Network field is the emergence of SSN (Shared Sensor Networks). SSN consists on multiple applications sharing the same sensoring and communication infrastructure. A major challenge for SSN is to extend the network's lifetime, since multiple applications potentially impose increased demand from the sensing and communication infrastructure. A promising solution to this challenge is the creation of algorithms that are capable to share the formation of clusters created by existing cluster-based routing algorithms in order to fully exploit the fact that a same sensing unit meeting the requirements of different applications is able to collect data only once and the collected result is shared by all of them. In this context, we propose an extension of LEACH algorithm [3], called S-LEACH (Shared LEACH). S-LEACH is able to handle the sensing requirements of several applications in SSN without executing redundant data collection. Our proposal is validated by experiments.**

***Keywords—LEACH, S-LEACH, Shared networks***

## I. Introduction

Wireless sensor networks (WSNs) are composed of: (i) Sensors that are low-cost, energy and resource constrained devices equipped with sensing interfaces and communication capabilities via wireless links; and (ii) one or more Sink node(s), a device that works as a gateway between the WSN and external networks. WSNs contain a large number of sensor nodes working collaboratively to monitor the target area and perform actions that may actively affect the physical environment [12]. In general, the sensor nodes are typically deployed in difficult-to-access remote locations. Most deployments of WSNs require unattended operation, thus, sensor nodes have to rely on batteries for communication and information gathering. A new trend on WSN field is the emergence of SSN (Shared Sensor Networks) [1] [9]. In SSN multiple applications share the same sensoring and communication infrastructure.

A major challenge for SSNs is to extend the network's lifetime, since multiple applications impose a a greater demand from the sensing and communication infrastructure. One of the techniques to extend the lifetime of a WSN consists in creating an hierarchy of clusters in order to route the collected data [2]. Those cluster-based routing algorithms are responsible for organizing the network in groups, called clusters, whose members are: cluster leader, called Cluster Head (CH), and sensor nodes, called cluster members (CM). The main role of a CH is to route the collected data from the sensors of its cluster towards the sink node through multihop communication. Since data communication is an energy-demanding operation and the overall distance among cluster members and its respective cluster-head is generally smaller than the distance among these cluster members and Sink node, cluster members save transmission energy and thus extend the network operational lifetime [3]. Some clustering techniques uses distance criteria [8] [9] such as received signal strength indicator (RSSI) and shortest communication distance among others to form clusters.

There are several techniques to extend the lifetime of a WSN. Most of them focus on the fact that communication between the sensor nodes and the base station is expensive, and so intends to reduce the number of transmissions. Such techniques search to organize the WSN in order to reduce the number of hops between the collected data and the Sink Node or to aggregate data to reduce transmissions. One of the most famous and widely adopted cluster-based routing algorithm for WSN is LEACH (Low-Energy Adaptive Clustering Hierarchy.

In the SSN scenario a cluster-based routing algorithm will have to deal with several applications simultaneously sharing the same infrastructure. Then, in a SSN, it is possible that a same sensing unit meets the requirements of different applications. This is potentially efficient since a same sensing unit could collect data only once and then share the results with several applications so as to further improve the use of limited node resources. So, a challenge for SSNs is related to the improvements and adaptation of existing cluster-based routing algorithms in order to fully exploit the fact that a same sensing unit that meet the requirements of different applications could collect data only once and the collected result be shared by all of them.

This paper proposes an application-aware extension of LEACH for SSN, called S-LEACH (Shared LEACH). S-LEACH is an application aware cluster-based routing algorithm for SSN because is designed to deal with several applications simultaneously sharing the same infrastructure of wireless sensor network. Therefore, in S-LEACH the clusters formation is created in order to route the data for multiple applications by transmitting these data once. Besides, by considering a context of shared applications in a common

This work is partly supported by the Brazilian Funding Agencies CNPq and FAPERJ under grants numbers FAPERJ - E-26/110.468/2012; CNPq - 307378/2014-4; CNPq 304941/2012-3; CNPq - 473851/2012-1; INMETRO - PRONAMETRO ; CNPq 477223/2012-5 (Universal ASGARD Project).

sensors infrastructure, the CH nodes of S-LEACH use data fusion algorithms designed for SSN [12] instead of traditional fusion techniques. The major problem with traditional fusion techniques when applied to SSNs is that they consider the universe (sensing units and data rates) of a single application. If the environment has a set of applications simultaneously running (as it is in a SSN scenario) and they use the same sensing unit (such as temperature), each application will apply fusion for each set of data instead of fusing these data for all applications. So, in order to deal with SSN scenarios, this work uses fusion techniques designed for SSN [12]. These fusion techniques designed for SSN reduce the number of collected data and the number of transmitted messages on SSNs, once that each sensor will send a single message independently of the number of applications, further extending the network lifetime [12].

The main benefits of using S-LEACH are: (i) S-LEACH extends the lifetime of SSN when compared to LEACH, since it reduces data traffic, by instead of transmitting the collected data of a same sensing unit that may meet the requirements of different applications several times, as LEACH would do, S-LEACH transmits this data only once for these applications; (ii), S-LEACH does not have a negative impact over the sensor resources (memory size). Our proposal is validated through simulations and tests on real nodes.

The rest of this paper is organized as follows. Section II reviews related works. Section III presents our designed clustering technique for SSN: S-LEACH. In Section IV, we describe the experiments to evaluate the proposal. And finally, at Section V, we conclude our paper and outline future works.

## II. Related work

Several works have proposed energy-efficient cluster-based routing algorithms for WSNs [3] [4] [5] [6] [7]. Our main related work is [3], presenting LEACH, an adaptive self-organizing cluster-based routing algorithm for WSN. The main idea presented in [3] is the creation of clusters in a distributed random way over the network. During each round a set of Cluster-Heads (CHs) is elected at random in a distribute way. Based on node proximity each non-CH node joins the nearest newly elected CH. After the CH election and the formation of clusters, LEACH starts to perform steady-state phase, when data related to the application are transferred to the base-station. LEACH [3] is one of the best known cluster-based routing algorithm for WSNs and is extended by the works T-LEACH [4], TL-LEACH [5] and Pegasis [6], which basically changes the criteria used to form clusters (the received signal strength indicator (RSSI) and shortest communication distance are example of criteria) in order to extend the network lifetime. Thus LEACH [3] as its extensions [4], [5] and [6] were designed to meet the requirements of a single-application. The difference among our proposal and LEACH [3] and its extensions [4], [5] and [6] is that our proposal extends LEACH to handle the requirements of multiple applications in a SSN context and use data fusion algorithms tailored for SSN scenario, while [3] [4] [5] [6] are cluster-based routing algorithms for WSNs, not tailored for a shared sensor network infrastructure (SSNs).

The work [8] proposes a reconfigurable semantic middleware, capable of handling data from multiple and heterogeneous applications. The proposed middleware [8] provides a precise and formal semantic value for each data; also allowing integration of different applications and the share of data that belongs to the same context. The proposed semantic middleware uses ontology to process information from each sensor node as well information related to the current network state. The proposed middleware promotes data enrichment and adds an automatic association between the meaning of the data, temporal and spatial aspects. The work suggests that the energy economy occurs through the decrease of communication, and pre-processing and collecting data only when are considered significant. Also, [8] adds semantic value to each data and associates it with the context of each application, keeping unrelated nodes in sleep mode, thus saving energy. The main difference from [8] to our proposal is that S-LEACH transmits this data only once for the several applications, saving energy. Also, we use data fusion algorithms suited for SSN.

To the best of our knowledge, there are some proposed works that deal with the clustering challenge for SSNs, but they present various restrictions that these clusterings imposed on applications. The main difference between S-LEACH and other clustering approaches found in literature is that our proposal relies on the creation of clusters for a shared infrastructure, while other clustering approaches are concerned with a single application. We also use data fusion algorithms designed for SSNs scenario in order to reduce data traffic, extending further network lifetime, since instead of transmitting the same data several times (each one of the applications), as LEACH would do, S-LEACH transmits this data only once for the several applications.

## III. S-LEACH

Our proposal, S-LEACH (Shared LEACH) extends LEACH to serve multiple applications efficiently. We have modeled the applications as a tuple, where an application $App_i = <Ident, Sts, Srs>$, where *Ident* represents the identification of the application, $Sts = <St_1, St_2, ..., St_n>$ represents the set of sensing units (for example, sensing temperature or humidity) required by the application and $Srs = <Sr_1, Sr_2, ..., Sr_n>$ represents the set of sensing rates that each sensing unit should perform to meet the requirements of this application.

This section is organized as follows: Section A presents an overview of the proposed algorithm and Sections B, C, D and E describe the phases of our algorithm.

### A. S-LEACH Overview

S-LEACH is performed by all the SSN nodes. Our algorithm is performed in a periodic basis (similarly to LEACH). Each period is a round. In S-LEACH each round encompasses a sequence of procedures performed in three mains phases: the *Set-up Phase*, the *Application Awareness Transient State Phase* and the *Steady-State Phase*.

***The Set-Up Phase*** (Section B) is responsible for setting the algorithm initial parameters, selecting the Role for each node, Cluster Head (CH) or Cluster Member (CM), and creating the clusters. The procedures of Role Selection and

Clusters Formation performed during this phase are the same as LEACH [3].

***The Application Awareness Transient State Phase*** (Section C) is responsible for the network nodes to be aware of several currently running applications in network. Additionally this phase of S-LEACH shares the clusters formation among several applications in order to the data collection of a same sensing unit be shared among the applications that demands the data generated by this sensing unit. Also, by performing this phase, S-LEACH can save energy of the sensor node that does not meet at least one of the requirements of the currently running applications or there is no currently running application on the network, because it keeps the nodes that do not attend an application on sleep state.

In ***the Steady-State Phase*** (Section D), each CM that meets at least one of the requirements of the currently running applications on the network should collect and send the collected data for the respective CH. Each CH executes data fusion algorithms especially designed to deal with multiple applications simultaneously in the SSNs context [12].

## *B. Set-up phase*

S-LEACH is a periodic algorithm that works in rounds. The first phase of each round is the Set-Up phase. All nodes should perform this phase at the same time, so the nodes must be synchronized. The clock synchronization problem is reported and addressed by several authors such as [3] and [8]. Considering the nodes synchronized, the *Set-Up phase* starts. This phase is divided in three procedures: (i) *Configurations*, (ii) *Role Selection* and (iii) *Clusters Creation*. The Role Selection and Clusters Creation procedures are the same as LEACH [3].

In the ***Configurations procedure*** all configurations are deployed to the nodes and the data structures are populated. Each CM node contains the following data structures: *SC, S, NV, L, CNI, A*, *FTU* and *APPs*.

The data structure *SC* is used by the sensor nodes in order to store its capacities. The *capacities* of the *sensor* node are the set of *Sts* (sensing units) and their respective *Srs* (sensing rates) that this node is able to perform and support the applications that demand these capacities. For each sensing unit, the quantity of data collected by the sensing unit (*St*) and its values is stored in the data structure *L*. The data structure *CNI* is is defined for each CM node in order to store the unique *network addresses* (NodeID) of the CH of its cluster. The data structure *APPs* is defined for all network nodes in order to contain all currently deployed applications in SSN. The sensor node uses the data structure *S* in order to store the identification of the applications that are supported by the sensor node. The data structure *A* stores the set of *St* and *Sr* for each application that is running on network and this node, when playing the CM node role, is able to attend its requirements. It is important to notice that differently from *A* data structure, *APPs* contains all currently running applications, not just the supported ones

Each CH node contains the following data structure: *NV* and *FTU*. The data structure *NV* stores the CM nodes that a certain CH has in its own cluster. This data structure has a field in order to store the size of the cluster, which is equal to its number of CM, and the rest of the structure is used to store the identifications (NodeID) of each CM node. The CH node stores in the *FTU* data structure the *identification* (id) of the *fusion technique designed for SSN* that should be performed given a set of currently running applications on network. As an example of a *fusion technique for SSN*, there is the *Enhanced Moving Average Filter* [13].

The ***Configurations procedure*** starts with the boot() procedure. This procedure represents the hardware initializations of each node, required for making the node operational and ready to start performing our algorithm. During the boot procedure the NodeID parameter is set. So, each node starts its operation knowing its own identification. The NodeID parameter stores a unique identification for each node in the network. After, the *init()* procedure is performed. Such procedure consists on setting the initial values of *S* (*supported applications*), *SC* (*capacities of the sensor node*) and *FTU* (*fusion technique given a set of running applications*). The other data structures are left empty and it will be filled during operation procedures of three phases of S-LEACH's, which are performed for all the nodes in the network. These nodes were already physically deployed over the structure

In the ***Role Selection procedure***, which is the same as LEACH, S-LEACH must select the roles of the nodes of the SSN as CH or CM. The role selection procedure in each node is made in the same way as in LEACH [3]. Each node chooses a random number α from the interval [0, 1] and calculates a threshold function $T(n)$, similarly to LEACH.

$$T(n) = \begin{cases} \frac{P}{1-P*(r \, mod \, \left(\frac{1}{p}\right))}, if \; n \in G \\ 0 \, , otherwise \end{cases} \quad (1)$$

Where P is the desired percentage of cluster heads present in the network, r is the current round and G is the set of CM nodes in the last $\frac{1}{p}$ rounds. At this point, since the ammount of nodes that are eligible to become CH decrease, the probability to become CH of the remaining nodes at G set must be increased. After $\frac{1}{p} - P$ rounds $T(n) = 1$ for any CM node. Once a node has become CH, it does not perform $T(n)$ again; in other words it is not eligible anymore. After $\frac{1}{p}$ rounds all nodes are eligible again.

In ***Clusters Creation*** procedure, which is the same as LEACH, each node verifies its role. If the role is CH (line 1, Fig 1), this node broadcasts CH_ROLE_BROADCAST message for its neighboring CMs to inform its role (line 2, Fig 1). After that, this node waits for the reception of the CM_JOIN messages disseminated by its neighboring CMs (line 3, Fig 1). After the reception of all the CM_JOIN messages disseminated by its neighboring CMs, this CH creates a TDMA schedule for each CM node that joined its cluster (line 4, Fig 1). Finally, this CH sends TDMA_SCHEDULE message for each one of its CMs informing TDMA schedule (line 5, Fig 1).

If the role is CM (line 7, Fig 1), this node waits for the reception of the CH_ROLE_BROADCAST messages disseminated by the CHs (line 8, Fig 1). Upon receiving these messages, this node chooses the CHs with the strongest RSSI (Received Signal Strength Indicator) (line 9, Fig 1). Then, this node sends a CM_JOIN message to this nearest CH in order to join its cluster (line 10, Fig 1). Next, this node waits for the reception of the TDMA_SCHEDULE message informing its TDMA schedule disseminated by its CH (line 11, Fig 1).

By the end of this procedure the network is self-organized in clusters, like the original LEACH [3]. Next the Application Awareness Transient State procedure starts in order to share this clusters formation among the deployed applications.

```
Input: T(n), α (a random number α from the interval [0, 1]).
Output: Node role and NV.

1.  If T(n) > α my role is CH:
2.    Send CH_ROLE_BROADCAST message for the neighboring CMs.
3.    Wait for CM_JOIN messages from each CM that has chosen this CH node. CH updates the NV.
4.    Create a TDMA schedule for each member node that joined my cluster.
5.    Send to each CM a TDMA_SCHEDULE message.
6.  End If.
7.  If T(n) < α my role is Member Node:
8.    Wait all CH_ROLE_BROADCAST messages.
9.    Choose the CH with the best RSSI among the received CH_ROLE_BROADCAST messages.
10.   Send a CM_JOIN message for the chosen CH.
11.   Wait for CH to inform my TDMA schedule, through TDMA_SCHEDULE_MESSAGE.
12. End If.
```

Fig 1. Pseudocode of Set-Up PhaseNext the Application awareness transient state phase starts

### C. The Application Awareness Transient State Phase

The pseudocode of the *Application Awareness Transient State* phase is shown in Fig 2.

```
Input: Applications to be deployed on the SSN.
Output: The set A, of the applications to be attended, SC the sensor capacities of each node.

1.  If my role is Cluster Head:
2.    Waits for the FUSION_DEPLOYMENT message from sink node informing the fusion technique to be used by the CH.
3.    Update the FTU structure, with the fusion technique to be used by a given set of currently running applications.
4.  End If.
5.  If my role is Member Node:
6.    Waits for the APP_DEPOYMENT message from sink node informing the ids of the applications to be deployed, and their respective Sts and Srs.
7.    Update the set APPS, to contain the identifications of all currently running applications on SSN, and their respective Sts and Srs.
8.    For each application identification i in set APPS:
9.        For Each sensing unit j in the set $Sts_{i,j}$.
10.           If $Sts_{i,j}$ from $App_i$ has the same $Sts_{,j}$ from SC, for giving application i:
11.               For the giving application i , in set A update $Srs_{i,j}$ with highest Sr between the values of $App_i$ and SC.
12.           End If.
13.       End For each.
14.   End For each.
15.   If $A = \varnothing$ :
16.       Sensor sleeps until next round.
17.   Else.
18.       Return set A, SC.
19.   End If.
20. End If.
```

Fig 2. Pseudocode of Application Awareness Transient State

First, each sensor node verifies its role (CH or CM) in network (lines 1 and 5, Fig. 2). If the sensor node is CH, it waits for the FUSION_DEPLOYMENT message informing the fusion technique to be used by it (line 2, Fig. 2). Next the CH updates the *FTU* structure with the fusion technique informed (line 3, Fig. 2).

If the role is CM, it waits for APP_DEPLOYMENT message (line 6, Fig. 2) from the *sink* node informing the currently running applications on SSN. The APP_DEPLOYMENT message contains the following fields: the number of applications to be created; for each application to be deployed, the message contains the Application 's ID (*id*), its sensing units (*Sts*), such as temperature, and its respective sensing rates (*Srs*) along with the quantity of sensing units that this application demands. Then, the node fills the set *APPS* to contain the identifications of all currently running applications (line 7, Fig. 2) in the SSN. Next, for each application in the *APPS* set (line 8, Fig. 2), the node has to verify if there is another application using the same sensing unit (line 10, Fig. 2). If the sensing unit is already being used by another application currently running in the SSN (in other words, the sensing unit already is in *Sc*), the node uses the most demanding rate for this sensing type (line 11, Fig. 2), and thus, serve all applications respecting the application that has the higher sensing rate. After performing these steps, the CM nodes that do not have any applications on set A (line 15, Fig. 2) remains in sleeping state in order to save energy (line 16, Fig. 2). Finishing the Application Awareness Transient State, the procedure has as result the sets *A* and the *SC*, containing the quantity of running applications, the identifications of the supported currently running applications on network, and for each application it stores the set of sensing tasks and sensing rates that the application needs and the *capacities* of the sensors (line 18, Fig. 2).

## D. The Steady-State phase

```
Input: The set A that contains all identifications of the currently running applications to be attended.
Output: Collected data for each concurrent application.

1.  While the CM is in Steady-State:
2.      For each application identification $i$ in APPS $\in A$, do:
3.          For each sensing units $Sts_{i,j}$ of giving application $i$, do:
4.              Collect data for the sensing unit $Sts_{i,j}$ in the respective sensing rate $Sr_{i,j}$ for giving application i.
5.              Store the collected data of sensing unit j $Sts_j$ on set L, for giving application i.
6.              End If.
7.          End For each.
8.      End For each.
9.      CM send the SENSED_DATA_SAMPLES message for its CH containing all collected Data (content of L) during the node's TDMA time slot.
10. End While.
```

Fig 3. Pseudocode of the Steady-State on CM view

The procedure shown in Fig 3 presents the pseudocode of this phase in CM point of view. First, during the Steady-State time (line 1, Fig. 3), the CMs verifies each identification of application (line 2, Fig. 3) in set *APPS* and each sensing unit of this application (line 3, Fig. 3). To, on following, collect the data (line 4, fig. 3). Next CM stores the collected data in the corresponding sensing unit at the set *L* (line 6, Fig. 3). And finally, the CMs send its collected data to the CH node during its TDMA time slot (line 9, Fig. 3).

It is important to notice that those collected data must be sent in a single message. It happens, because the data communication procedure demands much more energy than the processing procedure [11]. It is important to note that this is one of the most important step in order to extend the network lifetime. By sending all collected data in a single message the node prevents that the number of messages on network increases as the current number of applications increases.

The procedure shown in Fig 4 presents the pseudocode of this phase in CH point of view. At this point, the CH performs the data fusion techniques. In this procedure S-LEACH uses fusion algorithms designed for the SSN scenario [12] [13]. These techniques should be used because a fusion technique when applied in SSN context should be performed considering the different characteristics of each application (such as sensing units and sensing rates).

First, the CH receives SENSED_DATA_SAMPLES messages from its CM nodes and stores these collected data in the set *L* (line 1, Fig. 4). Based on the *FTU* structure, that contains the fusion technique to be used, CH performs the fusion technique on the data in the *L* set (line 2, Fig. 4). This is represented by the function FUSION (*L, FTU*). After that, the CH node send APPS_FUSED_DATA message (hop by hop using a routing protocol such as CTP) to the sink Node containing the data fusion results (line 3, Fig. 4).

```
Input: Collected data of CM, the set A, of applications identifications to be attended; FTU.
Output: The fused data for each application.

//Step 1: Receive the collected data
1. Store in the set L the data received through the message SENSED_DATA_SAMPLES.
//Step 2: Data fusion
2. Executes the function FUSION (L, FTU) in order to select fusion technique and to fusion the data
3. Send APPS_FUSED_DATA message to the Sink Node containing the data fusion results.
```

Fig 4. Pseudocode of the Steady-State on CH view

## IV. Experiments

This section describes the experiments conducted with S-LEACH in SSN scenarios for evaluating the impact of the algorithm in the WSN, compared to LEACH approach, in terms of the network lifetime and the required amount of memory.

### A. Experimental Settings

The experiments were conducted in the SUN SPOT platform [14], a sensor platform particularly suitable for rapid prototyping of WSNs applications. The SUN SPOT SDK environment includes Solarium, that is a tool to manage SPOTS that contains a SPOT emulator that is useful for experimenting SPOT software and/or to create scenarios with a large number of nodes whenever the real hardware is not available. The proposed algorithm was deployed on the SUN SPOT platform rev8 hardware [15] (1 Mb RAM memory sized, 8 Mb of Flash memory and AT91SAM9G20 with a master clock speed of 133.3248MHz).

In our experiments, we have used up to 10 applications (1, 2, 3, 5, and 10 applications). For each application, we assigned two randomly sensing units. Our implementation considered 1 – 5 different sensing units (accelerometers, temperature, light, humidity and presence) [10]. For each assigned sensing unit, we randomly assigned sensing rates varying from 1 to 5 seconds. The sensing units used in our applications represent the SUN SPOT embedded sensors.

All experiments were performed in a 100m x 100m field. The network nodes are in the Cartesian plane defined in the area {(0,0), (100,0), (0,100), (100,100)}. The sink is located far from any sensor node, at coordinates (200,100). All network nodes starts with 0.5 joules as initial energy within its batteries. We have randomly distributed 51 nodes in the network (50 nodes and 1 sink node). LEACH and S-LEACH are implemented using the message sizes in bits (Table I).

To simulate the message energy consumption in our experiments, the radio model used in the simulation is the same model as discussed in [3] which is the **first order radio**

**model**. Sending and Receiving messages are costly operations. Therefore, the usage of these operations should be minimal. Also it is assumed that the radio channel is symmetric so that the energy required to transmit a message from node *i* to node *j* is the same as energy required to transmit a message from node *j* to node *i*.

TABLE I. MESSAGES SIZES IN BITS

| | S-LEACH | LEACH |
|---|---|---|
| CH_ROLE_BROADCAST | 16 | 16 |
| CM_JOIN | 16 | 16 |
| TDMA_SCHEDULE | 80 | 80 |
| SENSED_DATA_SAMPLES | 176 | 80 |
| APPS_FUSED_DATA | 336 | 144 |
| APP_DEPLOYMENT | 26 | 26 |
| FUSION_DEPLOYMENT | 16 | 16 |

## B. Metrics

The metrics used for assessing the impact of S-LEACH in the WSN are: (i) the lifetime of the network and (ii) the memory consumption. In this paper, we adopted the same definition of network lifetime used in [11], which is the time elapsed until the first node in the WSN is completely depleted of its energy. The memory consumption is defined as the amount of memory used by the implementation of S-LEACH installed in the sensors nodes (RAM and ROM).

## C. Evaluating the impact of S-LEACH in the WSN

The main goal of the first set of experiments is to assess how long a SSN lasts using S-LEACH and LEACH algorithms by varying the number of applications (1, 2, 3, 5 and 10) simultaneously running in the network. Table II shows the network lifetime using S-LEACH and LEACH and the gains of S-LEACH, in terms of the lifetime, compared to LEACH for scenarios with 1,2,3,5 and 10 simultaneously running applications. The results of this experiment (TABLE II) shows that with the increment of the number of applications simultaneously running in the SSN, in both algorithms the network lifetime values are reduced. It is possible to observe in TABLE II that the network lifetime values achieved by S-LEACH were 28%, 101% and 340% higher than the network lifetime values achieved by LEACH for scenarios 3, 5, 10 applications, respectively.

TABLE II. USEFUL LIFETIME GAINED BY S-LEACH

| | LEACH | SLEACH | GAIN |
|---|---|---|---|
| 1 Application | 22.45 h | 9.5 h | -58% |
| 2 Applications | 10.45 h | 9.57 h | -8% |
| 3 Applications | 7.63 h | 9.75 h | +28% |
| 5 Applications | 4.85 h | 9.75 h | +101% |
| 10 Applications | 2.25 h | 9.90 h | +340% |

As more applications are simultaneously running in SSN, there is naturally an increase in the possibility of finding common sensing units among them. S-LEACH algorithms well utilizes this idea to reduce energy consumption of nodes. Beside that, our implementation of S-LEACH also uses the *Enhanced Moving Average Filter* [13] data fusion algorithm designed for SSNs scenario in order to reduce the number of transmission made by the CH to the BS, further extending network lifetime, since instead of transmitting the same data several times (one for each application), as LEACH would do, S-LEACH transmits data only once for the several applications.

On the other hand, we can observe in TABLE II that LEACH presents better result (58%) in terms of lifetime than S-LEACH for scenarios with a single application. This happens because as LEACH was designed to attend a single application and S-LEACH was designed to meet the requirements of multiple applications, S-LEACH computation procedures are more complex than LEACH's. For the scenario of two applications, LEACH also presents better result (8%) than S-LEACH. This happens due to the communication overhead imposed by S-LEACH. In short, S-LEACH presents better network lifetime than LEACH for scenarios with more than two applications running simultaneously in the SSN.

Considering ***the memory consumption*** in bytes for the sensor node, we noticed that the memory consumption of S-LEACH (29049 bytes (28.4 Kb)) was 37.8% higher than LEACH (21088 bytes (20.6 kb)). Although the memory consumption of S-LEACH presented an overhead in relation to LEACH, S-LEACH extends network lifetime of in relation to LEACH. It can be seen that the S-LEACH consumed a moderate amount of memory, because there are still 71.6, % of available RAM space. Additionally the external flash memory is completely available for the application.

## D. Comparison between simulated and real nodes

In this section, the same scenario simulated using Solarium was implemented on a real sensor node platform loacted in a controlled environment (our research laboratory at UFRJ). We aimed to validate the results obtained from simulations by comparing them with the results obtained from a real WSN platform. In this case, the nodes were kept stationary and disposed on the floor.

The experiment on simulated nodes consumed less energy than the real experiment, since there was no interference on the simulated experiments. On average, the obtained network lifetime value for real experiments with 3 applications was 8.9 hours, with standard deviation of 0.30h, while in the simulated environment the obtained network lifetime average value was 9.75 hours, with standard deviation of 0.20 hours.

## E. Discussion about of the accuracy of S-LEACH

In this section, we discuss that S-LEACH saves energy, preserves the data semantics as assures and enhances the data accuracy in SSNs making use of enhanced fusion techniques instead of traditional fusion techniques,. Traditionally fusion techniques were all designed for an application-specific network. This means that traditional fusion techniques process all the sensed data under the specific data semantics of a single target application. Recently, in works of [12] and [13] these traditional fusion techniques were adapted to the SSN scenarios in order to consider the distinct data semantics of each application. By data semantic we mean a pattern that describes an application and enables interoperability and integration between applications [13]. The issue of dealing with distinct data semantics of each application is particularly important and common in a distributed fusion system [13]; hereinafter we call this issue as application correlation in our work. If semantic differences were not taken into account, the fusion methods would produce unreliable results for different applications. Thus, by taking the semantics into account, it is possible to enhance the fusion techniques's accuracy. Besides the fact that the traditional fusion techniques process the sensed data under the specific data semantics of a single target application, they also assume that all the sensed data are weighted and handled equally and have the same data range. In SSN, fusion techniques need to consider that the same sensed data may have different degrees of importance for different applications and also different data ranges. Considering the aforementioned discussion, the authors in [12] and [13] argued that the existing traditional fusion techniques are not suitable to be used in SSN scenarios, since they were not conceived taking into account the SSN specific features. Therefore, there is a need for fusion techniques to deal with these features in order to achieve energy efficiency and reliable results in this emergent scenario. The authors in [12] and [13] proposed fusion techniques suited for SSNs (that we have used in the presented work) that are able to preserve the data semantics, to assure and enhance the data accuracy in SSNs since they combine information from multiple sensors, sources and applications to achieve inferences that are not feasible from a single sensor or source through probabilistic methods. The authors concluded that the data range and the weights assigned to applications (representing the relative priority assigned of each application) are extremely important in the fusion process. Since applications have different degrees of importance, it is intuitive to assume that their data have different degrees of importance. If an application less relevant but with a large data range has the same degree of importance of the other existing applications, it will lead to an error, i.e., a result which does not mirror the current situation of the environment. If we consider the data semantics, and weight it according to its importance, the fusion technique will return a result closer to reality. Therefore, we can say that S-LEACH making use the fusion techniques suited for SSN can guarantee and enhance the data accuracy.

## V. Conclusion

In this paper, we have presented an algorithm tailored for SSN, called S-LEACH (Shared LEACH). S-LEACH is an extension of LEACH algorithm that allows the formation of clusters to serve multiple applications efficiently. The results of our experiments shows that S-LEACH increased the network lifetime of the experimented scenarios and does not have high memory consumption. As future work we direct the research and development of different WSN cluster-based routing protocols in order to be able to cluster nodes based on the applications requirements instead of a geographical position. In S-LEACH we have extended LEACH, that performs the clustering scheme based on geographical position of the nodes. We advocate that base the clustering in the application requirements better suits cluster-based routing algorithms for SSNs. Another direction for future work consists on designing a mechanism capable to decide which fusion technique should be applied, given a set of application running in the network. Through some strategies, such as computational intelligence, finding the most appropriate fusion technique is useful in environments where applications change quickly, such as the SSN scenario.

## References

[1] Claudio M. de Farias, Luci Pirmez, Flavia C. Delicato, Wei Li, Albert Y. Zomaya, Jose N de Souza, "A scheduling algorithm for shared sensor and actuator networks," The International Conference on Information Networking 2013 (ICOIN2013), pp. 648-653, 2013 International Conference on Information Networking (ICOIN), 2013

[2] Yan Zhang, Laurence T. Yang, Jiming Chen; "RFID and Sensor Networks: Architectures, Protocols, Security, and Integrations"; Chapter 12 - Clustering in Wireless Sensor Networks; p334 ; ISBN 9781420077773; November 4, 2009 by CRC Press

[3] Heinzelman, W.R.; Chandrakasan, A.; Balakrishnan, H., "Energy-efficient communication protocol for wireless microsensor networks," System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on , vol., no., pp.10 pp. vol.2,, 4-7 Jan. 2000

[4] Hong, Jiman; Kook, Joongjin; Lee, Sangjun; Kwon, Dongseop; Yi, Sangho; "T-LEACH: The method of threshold-based cluster head replacement for wireless sensor networks"; Journal Article, Information Systems Frontiers, Springer US, 513-521, 1387-3326; 2009

[5] Loscri, V.; Morabito, G.; Marano, S., "A two-levels hierarchy for low-energy adaptive clustering hierarchy (TL-LEACH)," Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd , vol.3, no., pp.1809,1813, 25-28 Sept., 2005.

[6] Lindsey, S.; Raghavendra, C.S., "PEGASIS: Power-efficient gathering in sensor information systems," Aerospace Conference Proceedings, 2002. IEEE , vol.3, no., pp.3-1125,3-1130 vol.3, 2002

[7] AA Abbasi, M Younis; A survey on clustering algorithms for wireless sensor networks; Computer communications, 2007

[8] Bispo, K. a., Rosa, N. S., & Cunha, P. R. F. (2012). A semantic solution for saving energy in wireless sensor networks. 2012 IEEE Symposium on Computers and Communications(ISCC), 000492–000499. doi:10.1109/ISCC.2012.6249344

[9] Leontiadis, I., Efstratiou, C., Mascolo, C. and Crowcrof, Jt. 2012. SenShare: transforming sensor networks into multi-application sensing infrastructures. In Proceedings of the 9th European conference on Wireless Sensor Networks (EWSN'12), Gian Pietro Picco and Wendi Heinzelman (Eds.). Springer-Verlag, Berlin, Heidelberg, 65-81.

[10] S. Xiong, J. Li, M. Li, J. Wang, and Y. Liu, "Multiple Task Scheduling for Low-Duty-Cycled Wireless Sensor Networks," in INFOCOM '11.

[11] RAGHUNATHAN, V. et al. Energy-aware wireless microsensor networks. IEEE Signal Processing Magazine, v. 19, n. 2, p. 40–50, 2002.

[12] Farias, C.; Pirmez, L.; Delicato, F.; Carmo, L.; Wei Li; Zomaya, A.Y.; De Souza, J.N., "Multisensor data fusion in Shared Sensor and Actuator Networks," Information Fusion (FUSION), 2014 17th International Conference on , vol., no., pp.1,8, 7-10 July 2014

[13] de Farias, C. M., Pirmez, L., Delicato, F. C., Dos Santos, I. L. and Zomaya, A. Y.. 2012. Information fusion techniques applied to Shared Sensor and Actuator Networks. In Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012) (LCN '12). IEEE Computer Society, Washington, DC, USA, 188-191.

[14] Wilde, E.; Guinard, D.; e Trifa, V. Architecting a Mashable Open World Wide Web of Things, Institute for Pervasive Computing, ETH Zürich, Zürich, Switzerland, No. 663, February 2010

[15] Release 6.0 of the Sun SPOT SDK; Sun SPOT SDK Release Notes; Sun™ SPOT Programmer's Manual Release v6.0 (Yellow):New Rev 8 hardware : http://www.sunspotworld.com/docs/Yellow/ReleaseNotes.html;http://www.sunspotworld.com/docs/Yellow/SunSPOT-Programmers-Manual.pdf