Copyright ⓒ2017 KSII

A Software Defined Networking Approach to Improve the Energy Efficiency of Mobile Wireless Sensor Networks

Joaquin Aparicio[12], Juan Jose Echevarria[12] and Jon Legarda[12]
[1] DeustoTech - Deusto Foundation, Avda Universidades, 24, 48007, Bilbao, Spain
[2] Faculty of Engineering, University of Deusto, Avda. Universidades, 24, 48007, Bilbao, Spain.
[e-mails: {joaquin.aparicio, juanjose.echevarria, jlegarda}@deusto.es]

Received December 5, 2016; revised March 16, 2017; accepted April 5, 2017; published June 30, 2017

Abstract

Mobile Wireless Sensor Networks (MWSN) are usually constrained in energy supply, which makes energy efficiency a key factor to extend the network lifetime. The management of the network topology has been widely used as a mechanism to enhance the lifetime of wireless sensor networks (WSN), and this work presents an alternative to this.
Software Defined Networking (SDN) is a well-known technology in data center applications that separates the data and control planes during the network management. This paper proposes a solution based on SDN that optimizes the energy use in MWSN. The network intelligence is placed in a controller that can be accessed through different controller gateways within a MWSN. This network intelligence runs a Topology Control (TC) mechanism to build a backbone of coordinator nodes. Therefore, nodes only need to perform forwarding tasks, they reduce message retransmissions and CPU usage. This results in an improvement of the network lifetime.
The performance of the proposed solution is evaluated and compared with a distributed approach using the OMNeT++ simulation framework. Results show that the network lifetime increases when 2 or more controller gateways are used.

Keywords: Energy efficiency; IoT; MWSN; SDN; Topology control

1. Introduction

The Internet of Things (IoT) is a term used today to describe the network of physical devices which can sense and react to their environments and communicate with users or with other devices [1], Thus, Wireless Sensor Networks (WSNs) are one of the main building blocks of this paradigm [2] [3] [4] [5]. As they represent the interface with the physical environment [6]. As the number of connected devices increase, so do the data transmissions. This would overload the network, and consequently increase the energy consumption.

WSNs [7] operate considering a distributed approach, where each node takes routing and topology decisions based on node-based computation of the gathered information. It allows a better network fragmentation management but it lacks flexibility in terms of topology management and routing rules, and so it is more difficult to use the network lifetime as the main network management criteria [8].

Mobility in WSNs is known as Mobile Wireless Sensor Networks (MWSNs). These networks add new challenges as they increase the amount of transmitted messages in order to get their network services updated, which implies a higher energy consumption, and consequently a shorter network lifetime. At the same time, there are several silos or isolated domains of WSNs that rely on static infrastructures. This fact makes difficult to resolve the routing overhead due to message collisions and retransmissions (energy waste) derived from the mobility.

In this context, several architectures of data collection in MWSNs have been proposed in order to address the energy conservation problem [9], and there are also energy-efficient methods focused on the routing perspective [10]. However, they have to deal with other challenges besides energy consumption, like computation capabilities, hardware constraints or network reorganization and hybrid technologies orchestration. Current proposals do not consider an integral vision that could address those challenges, specially the energy usage. They deal with those problems from an isolated layer perspective (i.e., MAC, Physical, Routing, Application), increasing the probability of network fragmentation and decreasing the flexibility.

This work proposes increasing the network lifetime in MWSNs by means of a Software Defined Networking (SDN) perspective. Our proposal solves the contention and the isolated operations issues that occur in current distributed network models. A Topology Control (TC) technique is deployed above the MAC layer by abstracting the routing control in order to optimize its availability in terms of energy lifetime. This approach enables plug and play extensions with other network layers, and it provides an architecture that can extend its connectivity to the hierarchical and cluster routing based protocols that consider a reliable energy usage [11].

The solution proposed in this work takes advantage of the separation of the control plane and the data plane defined in SDNs. Specifically, this paper proposes an algorithm that deploys a TC mechanism [12] in order to build a backbone of coordinator nodes that maximize the network lifetime. This energy saving is achieved by taking more accurate routing decisions based on a centralized knowledge of the complete network that reduces the amount of data transmissions. In addition, the control and the data plane separation results in an abstraction of the network nodes from the underlying physical layer, and so the integration of the nodes in any IoT domain can easily be implemented. For example, the orchestration of new routing metrics that consider route stability or channel contention and interference [13] can be easily deployed at the control plane at runtime operation of the

network services. Furthermore, distributed monitoring systems can be deployed in intelligent transportation systems (ITSs) [14] where their architectures are conceived from a tier perspective, and the control plane can take advantage of this by managing the upper tiers through the knowledge of the entire network.

2. SDN and MWSN - Related Work

Several approaches offer a limited separation of concerns between network tasks; as a result, these approaches deal with a low capacity to manage heterogeneous environments from an energy conservation perspective. Furthermore, there are difficulties to build network layer orchestration in order to reach cooperative communications. Thus, it is beneficial to give network frameworks the capacity to combine energy aware with efficient programmable network [15] and simple flow control like SDN paradigm. **Table 1** shows a comparison between different energy conservation approaches that can be applied to MWSN.

Table 1. Comparison of energy conservation approaches in MWSN

Approach	Energy aware topologies	Separation of concerns	Simple and efficient programmable	Cluster organization	Flow control	Control plane simplicity	Ref.
Energy consumption reduction	✓			✓		✓	[9]
Hierarchical topology				✓	✓	✓	[11]
SDN-TC	✓	✓	✓		✓		[16]

SDN provides a reliable solution for a unified management of computers, networks, and clusters [16]. It has been used to get a flexible routing control and real time monitoring in many types of networks [17] (e.g., Radio Access Networks (RAN) [18], data center networks and Wireless Adhoc Networks (WANETS) [19].). It has also been considered to build a virtual IoT framework capable to orchestrate multiple communication technologies (e.g., WiFi, ZigBee, Bluetooth, infrastructure networks, and other communications standards [20], [21].).

As the IoT infrastructure continues growing [22], there will be more WSN/MWSN deployments [23]. As mentioned before, this integration brings many benefits [24], but since these kind of networks are usually energy constrained devices, their low power operation becomes essential . In this sense, SDN paradigm has already been used in order to address this issue in the literature.

But SDN has also been analyzed as an option to optimize the network energy consumption. In [25], sleep scheduling mechanisms with a SDN perspective have been used in order to extend the network lifetime. Despite the energy saving is important, one of the main drawbacks of this approach is that the network availability is penalized.

An important advantage of SDN is that it offers the capability of switching between different topology algorithms in a centralized manner so that each node does not need to be programmed separately every time the algorithms are modified. That is why Topology Control (TC) mechanisms will be used in our approach in order to achieve the desired energy saving in the network.

2.1 Topology Control

In Topology Control, nodes are capable of changing their configuration according to some criteria that benefits the network lifetime without penalizing the network connectivity and throughput [26]. They can change the transmission range and the power modes, or even they can change their role in the network according to their eligibility of joining to a network backbone. These are known as "clustering approaches" and the *eligibility rules* can be part of the central intelligence deployed in the abstraction layer of the control plane.

A well known TC mechanism that uses clustering approach is Span [27]. In Span all network nodes take decisions using only locally gathered information from their neighbour nodes through broadcast messages. Span adaptively selects a coordinator backbone according to the available energy of the nodes and their *usefullness* to connect with two of their neighbours. Moreover, Span incorporates randomized backoff delays in the *usefullness* announcements in order to reduce the amount of collisions that occur when multiple nodes decide to be coordinators at the same time. A coordinator node is defined like a node that is part of the active communication backbone of the network; this connected backbone stays awake in order to forward packets between any source and destination nodes. The nodes that are not part of the active communication backbone remain in power-saving mode, with minimal communication tasks and periodic health checks. Our approach introduces the idea of a SDN-assisted TC management using the *separation of concerns* and central coordination of the SDN paradigm in order to reduce the energy consumption.

2.2 Openflow

Some research have been carried out in order to deploy the SDN technology in WSNs and MWSNs [28]. Sensor Openflow [29] tackles most of the challenges using the Openflow protocol [30], which is one of the main technologies for SDN. As Openflow requires IP connectivity, nodes in a MWSN can use TCP/IP implementations designed specifically for resource constrained devices (e.g., uIP and lwIP). Our solution uses the Sensor Openflow approach in order to adopt the SDN characteristics. As a result, we propose an algorithm, called **Cross-over-net**, that operates between the routing and transport layers, and can be considered as a complementary modification of the routing logic.

3. Cross-over-net Algorithm

Our solution proposes the use of the SDN paradigm in MWSN scenarios in order to maximize the network lifetime without affecting the availability. It lies in between a clearly centralized approach and a totally distributed approach. This intermediate state grants the capability of managing the network intelligence through controller gateways at the physical layer, and control abstraction at logical level. As a result, the network nodes avoid complex tasks and only execute forwarding tasks, decreasing the number of maintenance messages between nodes and hence prolonging the network lifetime.

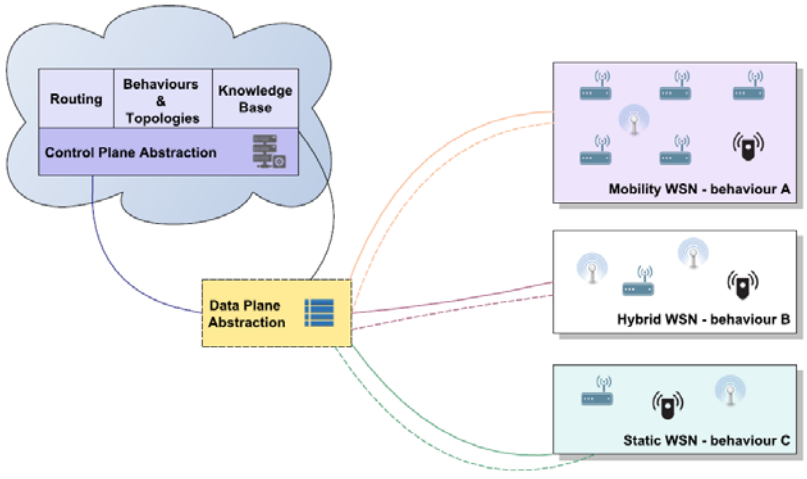
We assume the transmission range is at least twice the sensing range. Thus, we focus in the connectivity range among neighbours, because it is proved that if the transmission range is at least twice the sensing range, coverage implies connectivity in the network [31]. As a result, the energy savings are achieved by minimizing the transmission costs among nodes within a neighbourhood. The remainder of this section describes the general architecture of Cross-over-net. We present the operational logic that describes each Cross-over-net phase, and we highlight the main principles and definitions that rule our algorithm.

3.1 General Architecture

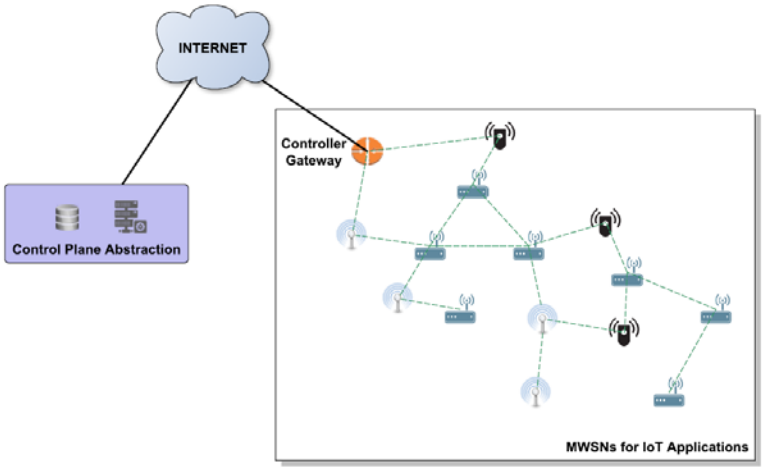
As shown in **Fig. 1**, our approach places the routing intelligence in the Internet cloud using SDN characteristics. This **control plane abstraction** (routing intelligence) manages MWSNs for IoT applications [32] through one or more **controller gateways**. The controller gateway is an extension of the controller placed in the Internet cloud (**control plane abstraction**); in fact, it could be considered to be the connection point of the central network controller (**control plane**) within the MWSN. Our experiments have considered several scenarios with different number of controller gateways, since the paths to reach the central control plane abstraction depend on the position of the controller gateways.

From the logical perspective, the proposed solution provides a central control layer that orchestrates multiple logical MWSNs layers. This allows to change at runtime the size and the number of networks at any time. Furthermore, the network gets an unified and automated management of several devices.

- **Control Plane Abstraction**. The central intelligence. This entity takes the critical decisions regarding the network behaviours, routing policies, and general network knowledge. It controls the frequency of the *checknet* process that will be described more in detail in section 3.2.
- **Data Plane Abstraction**. It is in charge of simplifying the packet forwarding tasks. The data plane in the nodes is limited to match inbound packets with flow entries contained in flow tables. Each flow entry consists of match fields and a routing action set. If the packet matches a flow entry in a flow table (provided by the controller), the corresponding instruction set is executed; otherwise, the packet returns to the controller or the data plane drops the packet.
- **Software-Defined Network Behaviours**. At any time, the network is capable to control different behaviours. Additionally, we can change the coordination backbone in each logical network; also, the network is capable to configure one single behaviour for all nodes.



(a) Physical representation



(b) Logical representation

Fig. 1. Cross-over-net general architecture

The key features of the general architecture of **Cross-over-net** are summarized below.

1) Knowledge base. As shown in **Fig. 2**, the control plane deployed in the network controller (or controllers) is able to maintain a knowledge base that receives feedback from the implemented processes in the production network. With each checknet iteration, the controller receives the key information for each node, like the computation of its usefulness, extra pairs of nodes that could be connected to the communication backbone, the shortest routing paths from a source to a destination, node's centrality within the network graph, most used nodes, energy expenditure in neighbourhoods, and decisions made over the time. All this information produces and refines a knowledge base that can be managed by the control plane in order to provide a more intelligent behaviour to the network [33]. Therefore, the controller can predict mobility trends based on historical information or probabilistic models. Furthermore, routing schemes according to energy consumption patterns in specific neighbourhoods could be established.

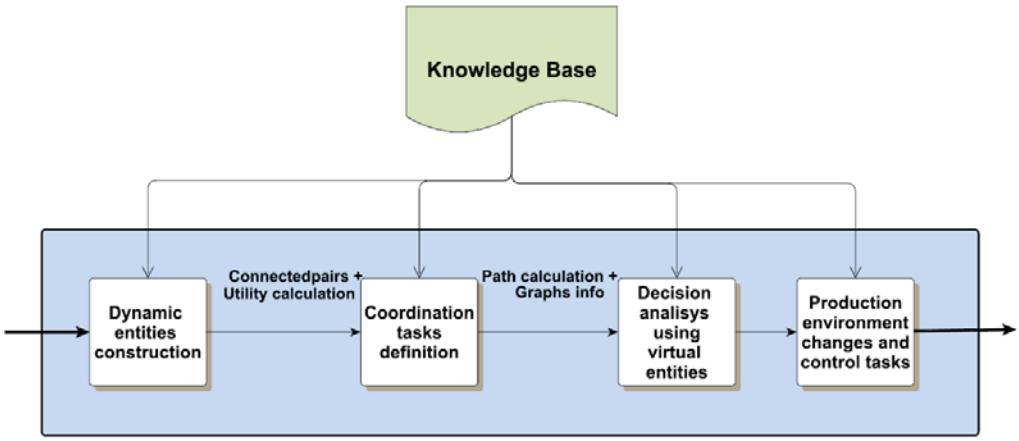


Fig. 2. Knowledge base creation process

2) Internal structures. The proposed controller deploys four abstract entities as shown in **Fig. 3**. The first entity is the production network. This structure monitors the real network environment, like the network status, network coordinators, and the state of the neighbourhoods of each node.

The second entity is the internal virtual network. This entity saves all the production network information and uses it as a test and development environment to analyse the results of the control updates before we change the network. The idea is to achieve additional energy savings when potential failures are detected. The internal virtual network entity manages 2 structures that contain the production network state and the list of coordinators. The network status contains a structure that saves the *node*, *neighbours*, and *neighbour of neighbours* fields. Finally, the coordinators structure is a list that indicates which nodes are coordinators at a specific time.

A third abstract entity is the network graph library. This entity contains algorithms, APIs, and databases that store a knowledge of the network graph in order to apply this knowledge in the checknet process. This graph library can apply power law considerations, centrality analysis, routing schemes and *cliques* identification. These techniques add more intelligence to the controller knowledge base and help in the execution of better-informed decisions [34].

Finally, the fourth entity represents the dynamic modules, which are a set of structures and methods that run every time a checknet process is triggered. The goal of these modules is to acquire customized and unique information of each network component.

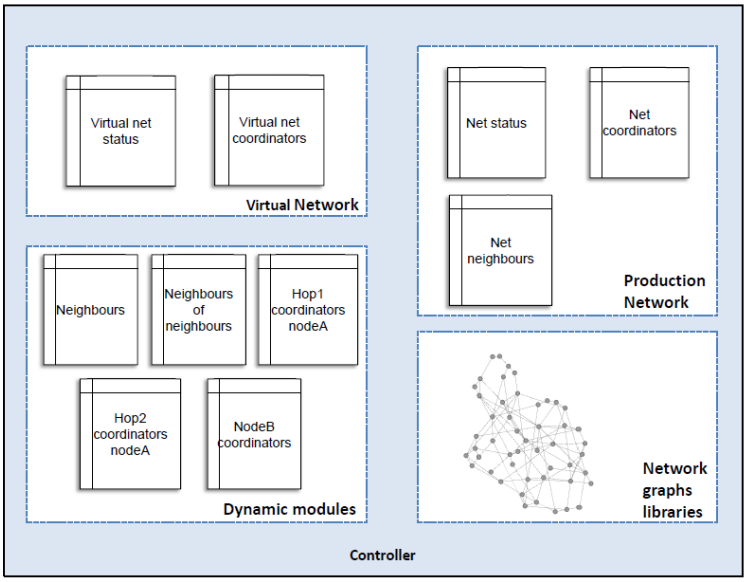


Fig. 3. Controller abstraction

Fig. 4 Shows the internal structures proposed for the network nodes that manage the data plane.

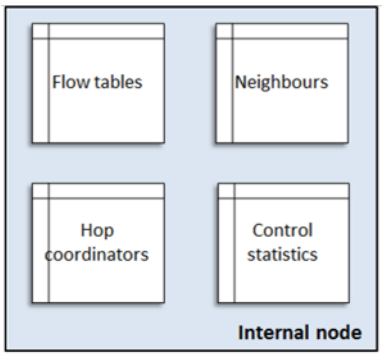


Fig. 4. Node abstraction

The flow tables are governed by the controller, information from its neighbours and coordinators is used for forwarding purposes, and finally control statistics are used in maintenance and error handling.

3.2 Operational Logic

The operational logic of the proposed algorithm can be divided into four phases: *initialization, checknet, transmissions, and controlcheck* as shown in **Fig. 5**. Specific tasks connect each phase. Moreover, the *initialization*, *checknet* and *controlcheck* phases execute their own processes in order to maintain the virtual structures described in the previous section 3.1. The controller gets the full graph of the network topology using the control channel, as well as the general details of the network state and customized information of each node. The four phases are described below:

Initialization. First, the initialization process begins when the nodes and the controller build their initial internal structures using the available MAC layer information and message broadcast. Afterwards, the discovery analysis is executed and moves to a coordinated cycle formed by the *checknet*, *transmission* and *controlcheck* phases.

CheckNet. Thus, in a time basis, the controller executes the *checknet* process; that is, a scan into its control tables (called cross-over) to determine if changes in the network graph and cluster subgraph (backbone) are needed.

Transmission. After that, the controller evaluates potential energy savings in an internal virtual network, and uses this information in order to decide over the routing flows.

ControlCheck. If there are no changes in the network, the controller is limited to the control tasks of the nodes. This operation reduces the messages collisions occurring in distributed operations, when multiple nodes want to get coordination roles at the same time.

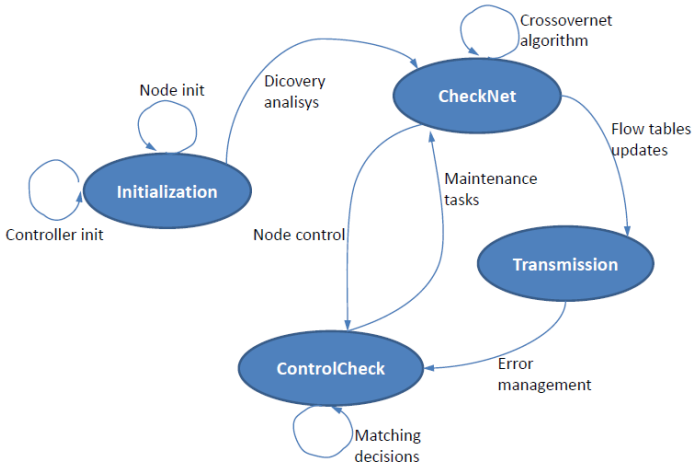


Fig. 5. Functionality context

3.3 Principles and States Definitions

The main behaviours of our algorithm are ruled by a set of principles and states definitions. Therefore, our algorithm knows how to deal with the energy consumption problem in each phase of the operational logic. These principles and states definitions are described below:

Let the graph *G = (V, E)* describes a given MWSN, where the Euclidean distance between each pair of nodes represents an edge in the graph *G*. Then, we select a backbone *G1 = (V1, E1)*, similar to a cluster domain set (CDS) builded with a subset of nodes of the graph *G*. This backbone is able to preserve the network capacity, and its construction is governed by the eligibility of the nodes to perform coordination tasks. A node becomes a coordinator considering its energy level and its utility to connect additional pair of nodes among its neighbours.

The controller determines changes in the graph *G* and subgraph *G1*. Also, the controller evaluates the changes in a test environment (internal virtual network), and decides in which network nodes applies the corresponding changes.

After the network initialization process, the algorithm selects the coordinator nodes based on the usefulness of each node, which is given by:

$$utility = \left(1 - \frac{E_r}{E_t}\right) + \left(1 - \frac{C_i}{\binom{N_i}{2}}\right) \quad (1)$$

where E_t denotes the maximum energy capacity of the node, E_r represents the remaining energy of the node, N_i represents the number of neighbours related to node i, and C_i is the number of additional pairs of neighbour nodes that would be linked if node i becomes a coordinator node.

State definitions. For simplicity reasons, we consider an independent control module that is in charge of the network discovery services of each node. The main definitions of **Cross-over-net** are described below:

Definition I. First, the network controller builds the neighbour's graphs of each node by using the information in their respective communication ranges. The pair of direct neighbours *i* and *j* is denoted by $NE_{(i,j)}$:

$$NE_{(i,j)} = DNE_{(i,j)} < R_i \quad (2)$$

where $DNE_{(i,j)}$ is the distance between node *i* and node *j*; R_i is the transmission range of node *i*. This process is calculated for each node *i* in range (0, *n*). The *n* is the total number of nodes in the network.

Definition II. A discovery module registers each Euclidean distance in an internal array for statistics and control purposes. The controller calculates the distance for each pair of nodes by using:

$$DNE_{(i,j)} = \varepsilon_{(i,j)} \quad (3)$$

The controller computes the registration of the distance from the node *i* to the node *j*, which is denoted by $DNE_{(i,j)}$. The controller builds this registry for each *i* in range (0, *n*) and for each *j* in range (0, *n*). The *n* is the total number of nodes in the network. If the nodes know their *x* position and their *y* position respecting a reference point, the controller also registers this information.

Definition III. The controller builds the entire network state and the list of coordinators. First, the length of the *neighbours net* structure is defined as:

$$Le_{NE} = \sum_{i=0}^{n_{ne}} NE_{(i)} \quad (4)$$

We use this boundary to calculate the global network state, which is a table that includes the node *i*, neighbour *j* and neighbour of neighbour *k* as its fields. The following equations are used:

$$Tn_a = NE_{(j)} \quad (5)$$

$$\forall\, Tn_a: GN_{(i,j)} = NE_{(i,j)} \quad (6)$$

$$\forall\, Tn_a: GN_{(i,k)} = NE_{(k)} \quad (7)$$

where Tn_a is the actual tested node;$GN_{(i,j)}$ denotes a direct neighbour pair of a node in *state net* structure; $NE_{(i,j)}$ is a direct neighbour pair of a nodes in the *neighbours net* structure; $GN_{(i,k)}$ represents the neighbour of neighbour *k* for node *i* in *state net* structure and $NE_{(k)}$ is the neighbour *k* in *neighbours net*. We get *k* by evaluating each neighbour *j* within the same *neighbours net* $NE_{(i,j)}$ in a previous nested iteration.

Definition IV. Periodically, the controller (*control plane*) builds a new network graph *G* and a new CDS *G1* according to the capacity of the nodes. We use the control channel required in the Openflow protocol operation to perform this network check process.

The length of the *state net* structure and the length of *local neighbour of neighbour* structure are defined according to equations (8) and (9) respectively:

$$Le_{GN} = \sum_{i=0}^{n_{gs}} GN_{(i)} \quad (8)$$

$$Le_{RNN} = \sum_{i=0}^{n_{rnn}} RNN_{(i)} \quad (9)$$

where n_{gs} denotes the total nodes in *state net* structure; $GN_{(i)}$ indicates a node in *state net* structure; n_{rnn} is the total nodes in *local neighbour of neighbour* structure and $RNN_{(i)}$ indicates a node in *local neighbour of neighbour* structure.

Let $GN_{(i,k)}$ be a pair formed by neighbour of neighbour node *k* for node *i* in the *state net* structure. Thus, the *local neighbour of neighbour* $RNN_{(i,j)}$ structure is designed as follow:

$$RNN_{(i,j)} = GN_{(i,k)} \quad (10)$$

considering the iteration of node *i* in range (0, Le_{GN}).

So, as a shown in **Fig. 6**, we get a single structure that includes the neighbours of the node *n* represented by $RNN_{(i)}$ and; at the same time, the neighbours of neighbours of the node *n* represented by $RNN_{(i,j)}$.

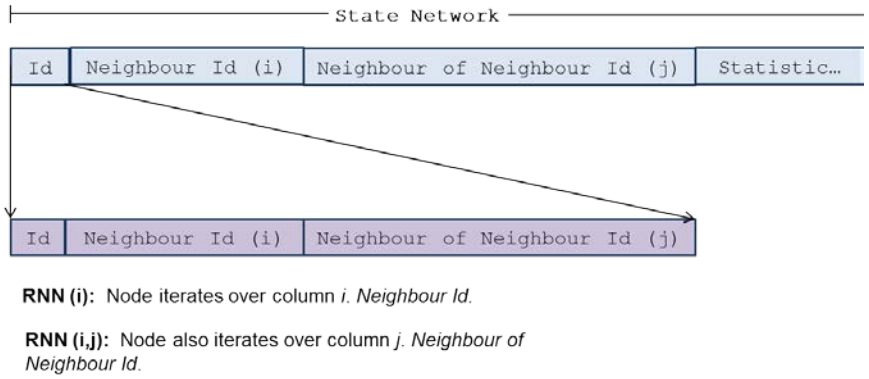


Fig. 6. Neighbour of neighbour structure

Then, the *local neighbours* structure can be calculated by:

$$RN_{(i)} = RNN_{(i)} \quad (11)$$

fulfilling that $RNN_{(i)} \neq RNN_{(i-1)}$ for each node *i* in range (0, Le_{RNN}). On the other hand, the controller calculates the potential *connected pairs CP* of each node *i* by using:

$$Le_{RN} = \sum_{i=0}^{n_{rt}} RN_{(i)} \quad (12)$$

$$CP = CC_{(RN_{(i)},RN_{(i+1)})} \quad (13)$$

the controller computes the length of *local neighbours*, which is denoted by Le_{RN} in order to use it like the bound for the iteration of node *i* within the *shared coordinators* expresion $CC_{(RN_{(i)},RN_{(i+1)})}$. The n_{rt} represents the total nodes of *local neighbours* structure. Also, the controller determines if the node is useful US_N with:

$$US_N = UTN > LNS_{(i)} \quad (14)$$

where *UTN* is the utility of the node (eq. (1)) and $LNS_{(i)}$ indicates the utility of its set of local neighbours. This comparison is made against each node *i* in range (0, Le_{RN}).

Finally, all the nodes follow a simple rule in order to transmit a message according to their *data plane*: if the destination is within its neighbourhood, then the node sends the packet directly; otherwise, it sends the packet to a coordinator node.

4. Experimental Results

Cross-over-net abstracts layers from different network models regardless of the technologies and protocols that govern them. Thus, the new abstraction decouples the control of the network models and manages them under a common strategy. Consequently, the unique mechanisms that are used in a specific model to organize the network are enabled to be applied ubiquitously to all other network models that are under SDN management. In our

approach, the control plane is deployed on one or more network devices with the required capacity; thereby, the risk of a critical failure point is minimized. The control is centralized at the abstract level but is distributed at the physical level. We use simulations in order to study the performance of our proposed solution *(Cross-over-net*) and compare it with *Span*, a well-known TC algorithm. We compare with *Span* because this type of TC algorithm performs more node coordinator-specific tasks than other TC approaches. Also, *Span* spends less messages transmissions among distributed communications when occurs the network fragmentation. We use the OMNeT++ simulation framework and we analyse energy consumption, network lifetime, and hopcount. We use a linear energy consumption model [35] that mimics the ZigBee communication technology in an ideal case where the transmission and reception costs are equal [36]. We test multiple scenarios with different number of controller gateways and we repeat experiments with different node densities under the same initial settings. Our algorithm operates between the MAC layer and the routing layer. The algorithm can be considered as a complementary modification to the routing logic. The controller dynamically builds the local graphs that contain the information of neighbours, and neighbours of neighbours structures for each node in order to analyze its utility to become a coordinator. In our scope, the controller handles a virtual internal network before sending the updates to the nodes; thereby, the only nodes that will receive an update message are those that are likely to change. The idea is to flood the network with less update messages and therefore minimize redundancy for energy efficiency.

4.1 Network model

We use a network model *N* of a MWSN that is represented by:

$$N = \{(S_i, T_i)\}, i \in [0, n] \quad (15)$$

where S_i is the current node and T_i represents the energy of S_i. We iterate *i* in the range (0, *n*), where *n* is the total number of nodes in the network *N*.

Neighbour nodes of *N* are denoted by:

$$Neighbour_i = \{(S_{(i,j)}, T_{(i,j)})\}, j \in [0, t_i] \quad (16)$$

where t_i denotes the number of nodes that are in range to be a neighbour of node S_i. We iterate *j* in the range (0, t_i).

4.2 Controller gateways – Route selection

We implement a procedure which can directly detect the best path to an existing *controller gateway* within the network. The procedure is described as follows:

First, we evaluate the possible access paths to a *controller gateway* using the selected routing protocol. We get a set of paths CP_i described by:

$$CP_i = g(n) + p(n), i \in [0, nc] \quad (17)$$

where *n* denotes the current evaluated node and *nc* denotes the number of *controller gateways* within the network. In each case, the formed path is *p(n)*, while *g(n)* represents the possibility to use the path. The values of *g(n)* only can be true or false.

Then, we check all the available paths and the cost related to their use. The general cost that a node has in order to reach an available *controller gateway* is defined as follows:

$$CPL_i = p_i(n) + c_i, i \in [0, nc] \quad (18)$$

where $p_i(n)$ is the available path and c_i is the associated cost of using that path in terms of energy consumption. The current evaluated node is *n* and the number of *controller gateways* within the network is *nc*.

Finally, we compare all available paths and their related cost in order to define the lowest-cost-path. This structure is described by:

$$lp = CPL_i + i, i \in [0, nc] \quad (19)$$

where$CPL_i$was defined in eq. (18) and *nc* denotes the number of *controller gateways* in the network.

In addition, we apply a SDN based architecture to the behaviour of each node, which has the following definitions:

1. All the routing decisions are made by the central controller.
2. There are no local decisions between nodes to communicate each other. They only follow the routing rules of their internal structures and wait *ControlCheck* messages to update instructions.
3. The controller transmits *ControlCheck* messages in a random time basis. So, the time interval changes in each executed simulation.
4. The global mobility of each node works with a random approach; therefore, the mobility model of the nodes change in each executed simulation.
5. We do not use a sleep scheduling and we take into account the idling costs between the nodes.

Finally, the controller backbone is maintained and updated (*checknet* process) by using the *ControlCheck* messages; so, the frequency configuration of the *ControlCheck* messages determines the variation time of the information collected at the controller.

4.3 Parameter settings - mobility scenarios

We simulated multiple networks scenarios of 10, 20, 30 and 50 wireless sensor nodes in a 475m x 475m square area. The nodes in our simulations have a transmission range of 75 meters. We considered up to 4 *controller gateways* for each scenario. For simplicity, network discovery services are considered on a separate control module. Furthermore, there are *n* sensor nodes randomly distributed during a simulation with a time limit of 40000s in each scenario. Routing algorithms have a deep relationship with the network topology. There are approaches that build a topology in a first stage and then deploy a routing algorithm. Other approaches build on-demand-topology while they run the routing algorithm. The SDN paradigm extends control over topology and routing since both entities can be managed and analyzed individually or in a hybrid perspective. As a result, all routing and topology management mechanisms become customizable on demand. This knowledge is available in the central control plane; so, the network is capable to get valuable information like: isolated nodes in the neighbourhoods, nodes with more connections, potential disconnections that cause network fragmentation, and paths related to a node. Then, this information is a key component in order to derive the energy consumption of a specific routing algorithm. Consequently, Cross-over-net enables a routing scheme that is capable of selecting its type according to the energy consumption needs. The method to set the routing considers dynamic aspects such as: structure graphs that define the neighbourhoods, the paths to each controller gateway, and the actions to execute in case of detecting a node outside the general network graph. In addition, this capability can be configured according to the type of technology and standards used by network devices even in heterogeneous networks. In all simulation experiments the *controller gateways* are capable of moving like the rest of the nodes. On a random time basis, among the total number of nodes, some nodes are randomly selected to move and the others remain static. We mimic the system movement following the random walk mobility model [37] with a distance traveled set to 25m. We select this

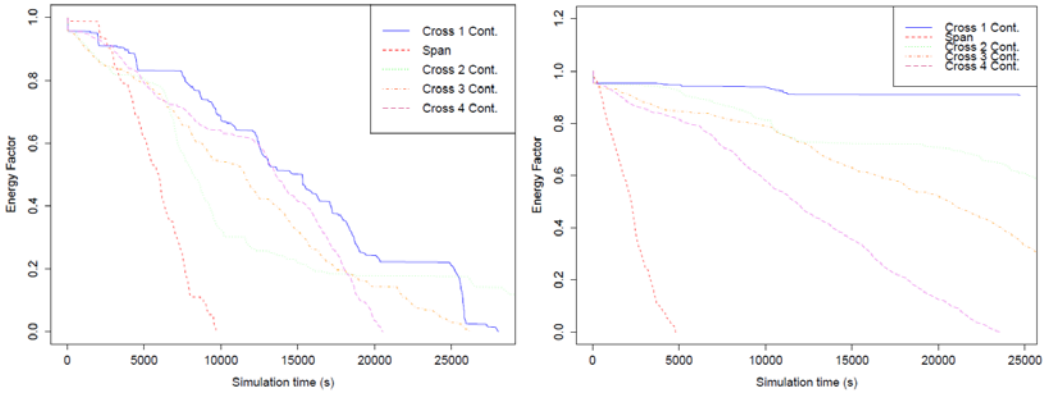
mobility model because the model applies to several MWSN scenarios. Each node chooses a random direction between 0 and *2Π;* then, the node is allowed to travel the configured distance for a random time interval. Periodically, the *control plane* builds the new network graph and a new backbone using the Cross-over-net algorithm explained in section 3. In order to get a more exhaustive comparison, we executed each simulation test 30 times and then their average values are computed. **Table 2** shows the main parameters of our simulations.

Table 2. Simulation parameter settings

Parameter	Value
Area	475mx475m
Number of controller gateways	1, 2, 3, 4
Radio range(m)	75
Total node energy capacity	2500 mAH
Transmission costs(Tx)	30 mA
Reception costs(Rx)	30 mA
Number of mobile nodes	10, 20, 30, 50
Simulation time limit	40000s
Number of iterations	30

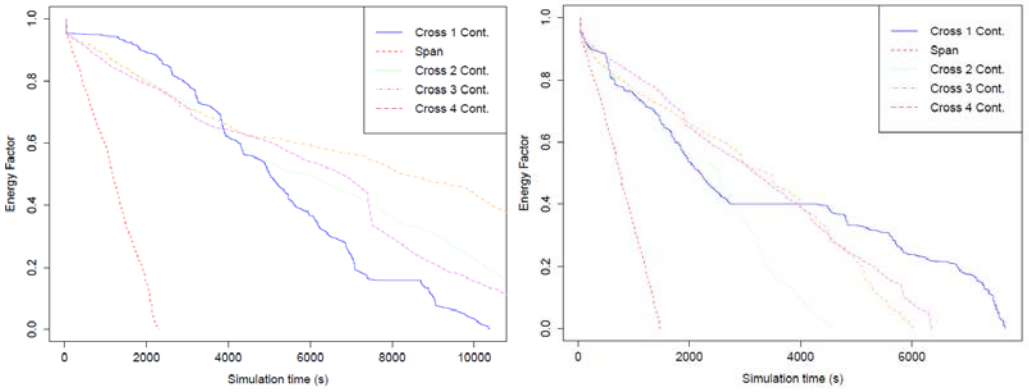
4.4 Energy consumption

The main objective of our approach is to save energy. The energy values define the usefulness of a particular node. The node utility defines the network graph and the coordinator subgraph. The algorithm uses the network graph and the coordinator subgraph entities to set the routing scheme. Therefore, the routing approach is capable of building routing decisions from an isolated network layer perspective or a combined network layers perspective. The central control abstraction manages all the processes without the reduction of the network services. One notion of the energy consumption is the *energy factor*, as it is discussed in [38]. Let E_r denotes the remaining energy in node u, and E_t the total energy of the node u at its maximum capacity. The *energy factor* is defined as the ratioE_r/E_t. **Fig. 7** describes the total *energy factors* evolution in one random iteration of our experiments. As expected, our solution performs better than the one with the decentralized control plane. However, we noticed that when we increase the number of nodes, the impact of the initialization costs are higher; so, the energy saving gaps between the algorithms get closer. This behaviour is due to the density of nodes and the position of the controller gateways in each case. The communication costs increase for the controller when it has to handle a large network graph because the controller needs to update much more nodes. As shown in **Fig. 7,** we observed that Cross-over-net reduces the energy consumption in all scenarios compared to Span. When the number of controller gateways increases, more energy is used. This behaviour is produced by the amplified coverage of nodes that we get with more controller gateways; thus, the network fragmentation decreases but the energy consumption increases. Nevertheless, our solution gets better energy savings even with 4 controller gateways. These results demonstrate that Cross-over-net handles TC better in networks with number of nodes between 10-30.



(a) n=10

(b) n=20



(c) n=30

(d) n=50

Fig. 7. Residual energy factor with different node density

We note that our solution tries to save energy by reducing the "request messages" required in order to announce the coordination/no-coordination tasks. Also, we get an orchestrated and coordinated manner to communicate neighbours that reduces the use of transmission messages. Furthermore, our algorithm proves that the individual neighbour and statistics tables can also be handled with less message overhead.

Finally, **Fig. 8** describes the mean value in simulation time when each network scenario reaches its complete depletion of energy. Our algorithm gets better values in all tested node density scenarios. We observed that in the 10, 20, 30 nodes scenarios, the most reliable results are with Cross-over-net with 3 and 4 controller gateways, because we get a feasible balance between coverage and energy use. Nevertheless, if we consider only the energy consumption metric, the Cross-over-net with 2 controller gateways is a better solution. We also noticed that Cross-over-net outperforms Span with remarkable differences in the 20 node scenarios. Finally, in the 50 node scenarios, despite Cross-over-net performs better than Span, results are closer for all the controller gateway configurations. Thus, the most effective solution is when we use 2 controller gateways. The **Table 3** shows a summary of the simulation time when the network uses the complete energy of all its nodes. We noticed that

when we increase the number of nodes, our algorithm is penalized with more communication costs. Nevertheless, these results show that even in 50 node scenario our solution requires less energy consumption.

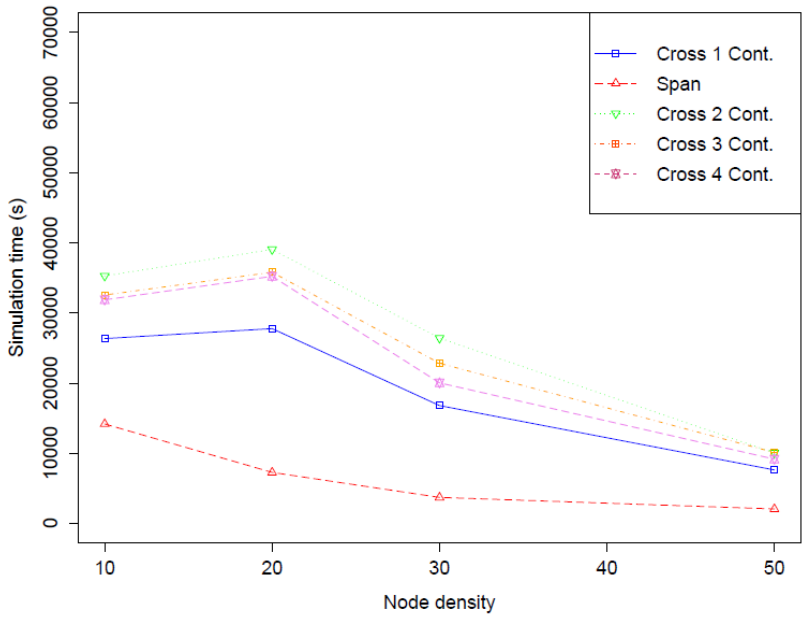


Fig. 8. Energy factor evolution with different node density

Table 3. Energy and lifetime comparison with different node density

Scenario	Cross - 1contgwy	Span	Cross - 2contgwy	Cross - 3contgwy	Cross - 4contgwy
10 node (Energy Factor)	26354s	14177s	35290s	32565s	31885s
20 node (Energy Factor)	27775s	7270s	39066s	35812s	35233s
30 node (Energy Factor)	16801s	3707s	26423s	22836s	20039s
50 node (Energy Factor)	7620s	2057s	10058s	10114s	9164s
10 node (Network Lifetime)	11730s	4572s	16492s	14686s	15808s
20 node (Network Lifetime)	12223s	1716s	15448s	10412s	10603s
30 node (Network Lifetime)	1945s	1816s	5588s	5906s	5703s
50 node (Network Lifetime)	730s	1062s	1610s	1915s	1894s

4.5 Network lifetime

In this section, we conduct tests to show the network lifetime evolution with different node densities. We have defined the network lifetime as the time when the first node death occurs. **Fig. 9** and **Table 3** present the results of the experiments considering the initial parameter values described in **Table 2**. As expected, the number of controller gateways affects the network lifetime. The best results are obtained with the deployment of 2 controller gateways. We noticed that with 1 controller gateway, Span performs better in a 50 node scenario. However, Cross-over-net is more consistent again when we deploy 2, 3, and 4 controller gateways. In summary, Span is more stable in large mobility scenarios when 1 controller gateway is considered. However, since the network behaviour is related to the density of the nodes near the controller gateway, the deployment of more controller gateway increases the

entire network lifetime. Cross-over-net introduces a method to place the network graphs within the central intelligence. The graphs are dynamic structures that change in time in order to support the network services. The central intelligence manages the graphs information even if some nodes change their position continually. Thereby, the central intelligence is capable of maintaining the network services when different nodes reach the energy depletion.

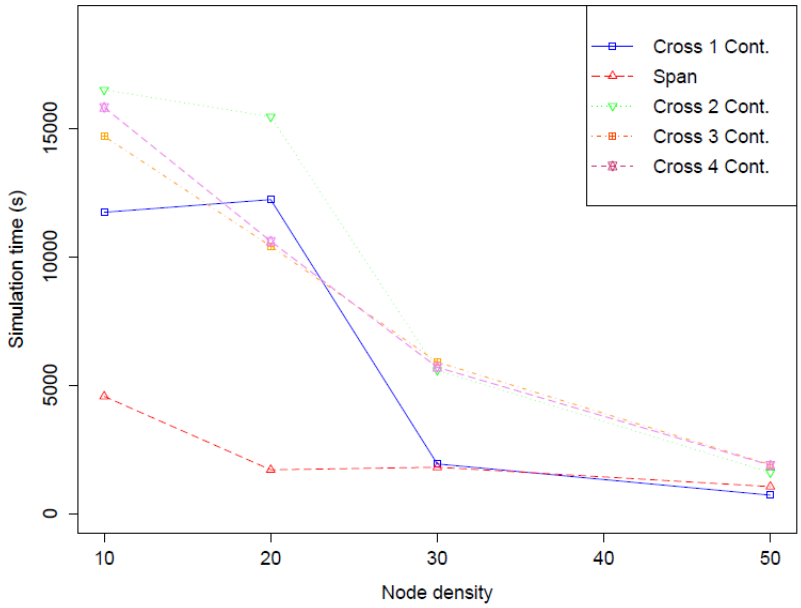


Fig. 9. Network lifetime evolution with different node density

4.6 Hopcount analisys

This section compares the hopcount between wireless sensors nodes. The hopcount is defined like the number of packets needed to announce the routing changes in each neighbourhood. Simulations follow the same initial parameter values described in **Table 2**. As shown in **Fig. 10**, our algorithm reduces the packet activity by 4 times compared to Span. We observed that in 10, 20, and 30 node scenarios, Cross-over-net uses less hops than Span. As expected, the hopcount activity increases as we add controller gateways in each scenario, but the results show that the hopcount used by Span is never exceeded. Nevertheless, for a 50 node scenario both algorithms tend to waste all the energy in the system; as a result, the hopcount is similar. It is clear that our solution reduces the use of packets when $n<30$. These results prove that a separate control plane in the mobile network can result in energy savings and better network lifetime with a reasonable growth in routing complexity. Moreover, the mobile nodes in Span take decisions only with local information and the nodes have to waste transmission messages in order to update its neighbours routing tables and coordinator announcements. In our algorithm, we can provide this updates with the control check of the network. However, when we increase the number of nodes ($n > 30$), the used hopcount tends to be similar. This result shows that Cross-over-net performs better when n is small. Also, the control plane is available to add new metrics besides hopcount, like the metrics that

estimate the routing based on length of the node links or node mobility ranges. Moreover, our algorithm is capable to deploy reliable routing based from hierchical and cluster energy efficient perspectives. This feature increases fault-tolerance capacity by offering some suitable alternative routes to forward packets in the presence of a link or node failure.

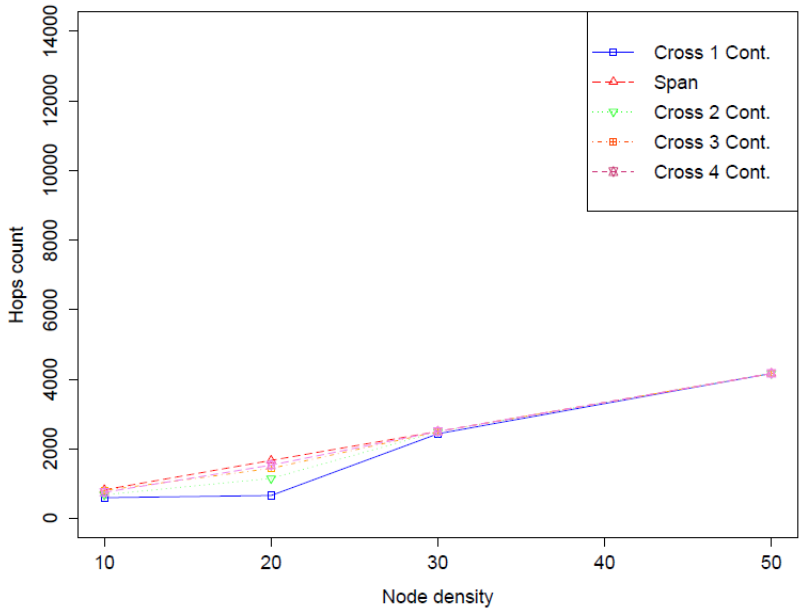


Fig. 10. Hopcount evolution with different node density

4.7 Coordination nodes analisys

A key behaviour in order to get our energy saving goal is the coordinator selection process. The **Fig. 11** shows the mean value of coordinator calls triggered in the set of ControlCheck messages. A coordinator call is defined as the number of nodes that assumed the role of coordinator in a particular check net iteration. We observed that the necessity of coordination calls is higher in Span when $n < 30$. These results imply that in these scenarios, our centralize approach benefits the network in terms of energy efficiency, because we reduce the average use of messsages that are necessary for the coordinators election. Also, we noticed that the use of coordinators is similar when we deploy 1, 2, 3 or 4 controller gateways; as a result, the distributed approach of Span is more efficient than Cross-over-net in managing the coordinator calls when $n = 50$.

Consequently, the **Fig. 12** compares the average of the total number of coordinators used in each scenario. We realized that in Cross-over-net, the number of coordinators increases as we add network nodes; however, Span tends to use less coordinators compared to Cross-over-net when the number of network nodes increases. These results show that when $n < 30$, Cross-over-net reduces redundant messages/collisions due to the distributed operation, i.e. reduce the number of coordinators.

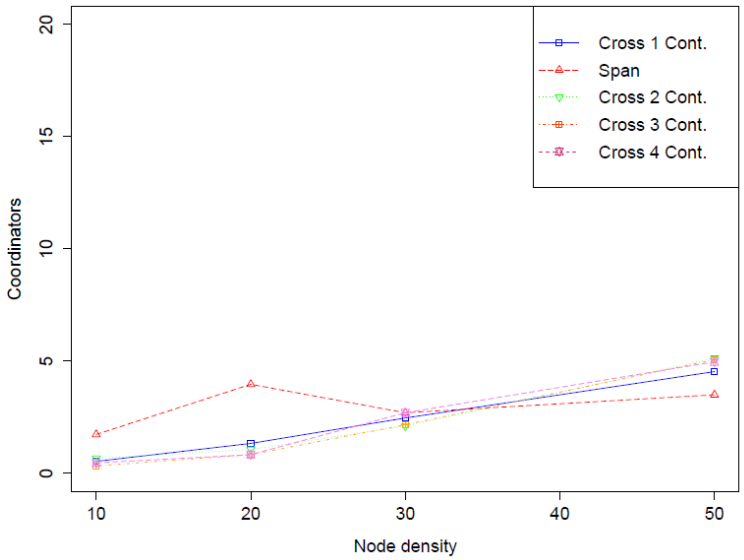


Fig. 11. Coordination calls with different node density

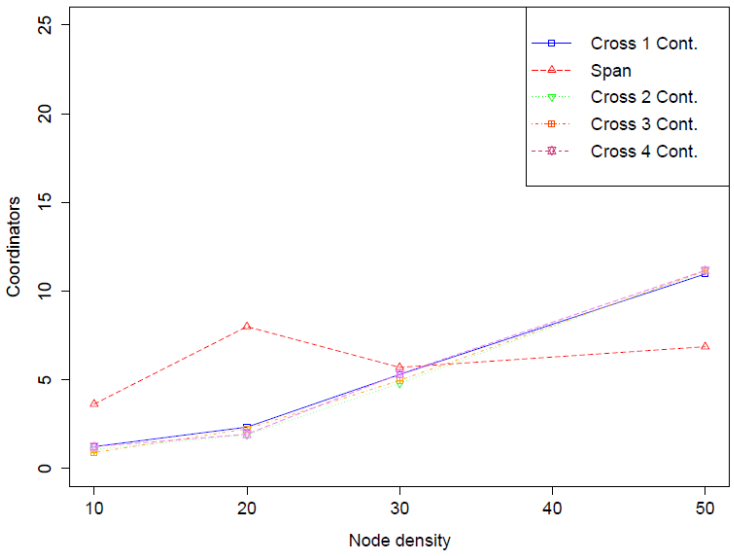


Fig. 12. Total number of coordinators with different node densities

Additionally, the coordinators information is collected at the controller. Then, the control plane uses the information to manage the graph knowledge that determines the network behaviours. Thus, the graph knowledge in the data plane is expected to go stale depending on the time rate that is defined by the frequency of the ControlCheck message. This type of system automatically performs more reliable topological changes than a system based on a distributed approach with partial network graphs. However, the control plane may require more time to execute the calculations.

5. Conclusion

In this work we propose an algorithm that addresses the problem of energy consumption in MWSNs. We propose a SDN approach that enables a separation of concerns, thus the network nodes are abstracted from the underlying IoT domain management. At the same time, the nodes are independent of a specific application; so, data query policies, TC mechanism, and routing schemes can be easily deployed. Cross-over-net decreases the number of transmission messages within the nodes neighborhoods; also, we noticed that the centralized utilities coordination reduces the time delay used in Span. However, this approach comes with a cost on the controller, as it will need more storage and computation capacities. Our results show that our approach uses energy more efficiently, which extends the network lifetime in low and medium size ($n < 30$) static and mobile WSNs. In order to obtain significant energy savings, we realized that the density of nodes, control check frequency, number of controller gateways, and the controller gateways' positions have a deep impact in the results. We get better energy preservation with 2 controller gateways, but nodes coverage gets penalized regarding to the scenarios with 3 and 4 controller gateways. The best deployment will always depend on the network requirements and use cases. Finally, the central controller is capable to manage a knowledge base of the entire network and changes the network topology according to user patterns or customized infrastructure characteristics.

Future work will focus on queue mechanisms in the control channel, knowledge base improvements in order to use graph theory algorithms, and the impact of sharing the controller tasks among network nodes.

References

[1] J. Gubbi, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013. Article (CrossRef Link).

[2] slange, "Internet of Things - Architecture — IOT-A: Internet of Things Architecture." [Online]. Available: http://www.iot-a.eu/public.

[3] Y. YIN, Y. Zeng, X. Chen, and Y. Fan, "The internet of things in healthcare: An overview," *J. Ind. Inf. Integr.*, vol. 1, pp. 3–13, Mar. 2016. Article (CrossRef Link).

[4] S. Li, L. D. Xu, and X. Wang, "Compressed Sensing Signal and Data Acquisition in Wireless Sensor Networks and Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2177–2186, Nov. 2013. Article (CrossRef Link).

[5] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Trans. Ind. Inform.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014. Article (CrossRef Link).

[6] D. Christin, A. Reinhardt, P. S. Mogre, and R. Steinmetz, *Wireless Sensor Networks and the Internet of Things: Selected Challenges*. Article (CrossRef Link).

[7] H. Karl, A. Wolisz, and A. Willig, Eds., *Wireless Sensor Networks*, vol. 2920. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. Article (CrossRef Link).

[8] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008. Article (CrossRef Link).

[9] G. Abdul-Salaam, A. H. Abdullah, M. H. Anisi, A. Gani, and A. Alelaiwi, "A comparative analysis of energy conservation approaches in hybrid wireless sensor networks data collection protocols," *Telecommun. Syst.*, vol. 61, no. 1, pp. 159–179, Jan. 2016. Article (CrossRef Link).

[10] M. H. Anisi, G. Abdul-Salaam, M. Y. I. Idris, A. W. A. Wahab, and I. Ahmedy, "Energy harvesting and battery power based routing in wireless sensor networks," *Wirel. Netw.*, vol. 23, no. 1, pp. 249–266, Jan. 2017. Article (CrossRef Link).

[11] H. K. D. Sarma, A. Kar, and R. Mall, "Energy efficient and reliable routing for mobile wireless sensor networks," in *Proc. of 2010 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops (DCOSSW)*, pp. 1–6, 2010. Article (CrossRef Link).
[12] P. Santi, "Topology Control in Wireless Ad Hoc and Sensor Networks," *ACM Comput Surv*, vol. 37, no. 2, pp. 164–194, Jun. 2005. Article (CrossRef Link).
[13] Yi-Han XU, Yin WU, and Jun SONG, "A Routing Metric to Improve Route Stability in Mobile Wireless Sensor Networks," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 5, May 2016. Article (CrossRef Link).
[14]M. H. Anisi and A. H. Abdullah, "Efficient Data Reporting in Intelligent Transportation Systems," *Netw. Spat. Econ.*, vol. 16, no. 2, pp. 623–642, Jun. 2016. Article (CrossRef Link).
[15] M. Heni and R. Bouallegue, "Power control in reactive routing protocol for Mobile Ad Hoc Network," ArXiv12051657 Cs, May 2012.
[16] I. Ku, Y. Lu, and M. Gerla, "Software-Defined Mobile Cloud: Architecture, services and use cases," in *Proc. of Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*, pp. 1–6, 2014. Article (CrossRef Link).
[17] Valdivieso Caraguay *et al.*, "SDN: Evolution and Opportunities in the Development IoT Applications," *Int. J. Distrib. Sens. Netw.*, vol. 2014, p. e735142, May 2014. Article (CrossRef Link).
[18] D. Pompili, A. Hajisami, and T. X. Tran, "Elastic resource utilization framework for high capacity and energy efficiency in cloud RAN," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 26–32, Jan. 2016. Article (CrossRef Link).
[19] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A Software Defined Networking Architecture for the Internet-of-Things." Article (CrossRef Link).
[20] D. Zeng, T. Miyazaki, S. Guo, T. Tsukahara, J. Kitamichi, and T. Hayashi, "Evolution of Software-Defined Sensor Networks," in *Proc. of 2013 IEEE Ninth International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pp. 410–413, 2013. Article (CrossRef Link).
[21] G. Suciu, S. Halunga, A. Vulpe, and V. Suciu, "Generic platform for IoT and cloud computing interoperability study," in *Proc. of 2013 International Symposium on Signals, Circuits and Systems (ISSCS)*, pp. 1–4, 2013. Article (CrossRef Link).
[22] S. Li, L. D. Xu, and S. Zhao, "The internet of things: a survey," *Inf. Syst. Front.*, vol. 17, no. 2, pp. 243–259, Apr. 2014. Article (CrossRef Link).
[23] H. Yan, L. D. Xu, Z. Bi, Z. Pang, J. Zhang, and Y. Chen, "An emerging technology – wearable wireless sensor networks with applications in human health condition monitoring," *J. Manag. Anal.*, vol. 2, no. 2, pp. 121–137, Apr. 2015. Article (CrossRef Link).
[24] L. D. Xu Editor-in-Chief, "Inaugural issue," *J. Ind. Inf. Integr.*, vol. 1, pp. 1–2, Mar. 2016. Article (CrossRef Link).
[25] Y. Wang, H. Chen, X. Wu, and L. Shu, "An energy-efficient SDN based sleep scheduling algorithm for WSNs," *J. Netw. Comput. Appl.*, vol. 59, pp. 39–45, Jan. 2016. Article (CrossRef Link).
[26] A. A. Aziz, Y. A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich, "A Survey on Distributed Topology Control Techniques for Extending the Lifetime of Battery Powered Wireless Sensor Networks," *IEEE Commun. Surv. Tutor.*, vol. 15, no. 1, pp. 121–144, First 2013. Article (CrossRef Link).
[27] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 481–494, Sep. 2002. Article (CrossRef Link).
[28] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *Proc. of 2011 International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 1, pp. 594–600, 2011. Article (CrossRef Link).
[29] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012. Article (CrossRef Link).

[30] N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput Commun Rev*, vol. 38, no. 2, pp. 69–74, Mar. 2008. Article (CrossRef Link).
[31] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc Sens. Wirel. Netw.*, vol. 1, no. 1–2, pp. 89–124, 2005. Article (CrossRef Link).
[32]A. Whitmore, A. Agarwal, and L. D. Xu, "The Internet of Things—A survey of topics and trends," in *Proc. of Inf. Syst. Front.*, vol. 17, no. 2, pp. 261–274, Mar. 2014. Article (CrossRef Link).
[33] G. Pantuza, F. Sampaio, L. F. M. Vieira, D. Guedes, and M. A. M. Vieira, "Network management through graphs in Software Defined Networks," in *Proc. of 2014 10th International Conference on Network and Service Management (CNSM)*, pp. 400–405, 2014. Article (CrossRef Link).
[34] D. Katsaros, N. Dimokas, and L. Tassiulas, "Social network analysis concepts in the design of wireless Ad Hoc network protocols," *IEEE Netw.*, vol. 24, no. 6, pp. 23–29, Nov. 2010. Article (CrossRef Link).
[35] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. of IEEE INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, vol. 3, pp. 1548–1557 vol.3, 2001. Article (CrossRef Link).
[36] E. Casilari, J. M. Cano-García, and G. Campos-Garrido, "Modeling of current consumption in 802.15.4/ZigBee sensor motes," *Sensors*, vol. 10, no. 6, pp. 5443–5468, 2010. Article (CrossRef Link).
[37] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wirel. Commun. Mob. Comput.*, vol. 2, no. 5, pp. 483–502, Aug. 2002. Article (CrossRef Link).
[38] J. Aparicio, J. Legarda, J. Larranaga, and J. J. Echevarria, "Cross-over-net: An energy-aware coordination algorithm for WANETs based on software-defined networking," in *Proc. of 2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 37–41, 2015. Article (CrossRef Link).

Joaquin Aparicio was born in Santa Ana, El Salvador, in 1980. He received the B.S. degree in information systems engineering from the Catholic University of El Salvador, in 2004 and a MBA degree from Central American University Jose Simeon Cañas, El Salvador, in 2010. He is currently pursuing the Ph.D. degree in computer science at University of Deusto, Bilbao, Spain. He has worked at Ernst & Young, PwC and DELL as IT-consultant. His main research areas of interest are: Internet of things, mobile wireless networks, energy-aware systems, software defined networks and virtualization.

Juan Jose Echevarria was born in Bilbao, Spain, in 1985. He received the B.S degree in telecommunications engineering from University of Deusto, Bilbao, Spain, in 2008 and the M.S degree in embedded systems engineering from UNED and Complutense University of Madrid, Madrid, Spain, in 2012. He is currently pursuing the Ph.D. degree in computer science and telecommunications at University of Deusto. Since 2009 he has been a Research Assistant with the Deusto Institute of Technology, University of Deusto. His current research interests include pervasive computing and energy-aware embedded systems.

Dr. Jon Legarda received the M.S. degree in Electrical and Electronic Engineering (2001) and the PhD in Engineering (2004) from the University of Navarra. In 2004 he joined CEIT as project manager in communication electronics and embedded systems. In 2011 he joined DeustoTech (Institute of Technology of the University of Deusto) as principal investigator (TELECOM unit). His research areas of interest are embedded systems, communication networks and textile antennas. He has published 16 articles, 7 conferences, and 4 books and chapters. Since 2012 he is recognize as Senior Lecturer by the National Agency for Quality Assessment and Accreditation (Spain).