Performance analysis of software-defined multi-hop wireless sensor networks

F. Fernando Jurado-Lasso, *Graduate Student Member, IEEE,* Ken Clarke, and Ampalavanapillai Nirmalathas, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a model-based characterization of energy consumption in a software-defined wireless sensor network (SD-WSN) architecture in an effort to examine the implications for network performance when making the WSN reprogrammable. The proposed model consists of breaking down all key functions involved in the correct functioning of an SD-WSN, namely; neighbor discovery, neighbor advertisement, network configuration, and data collection. The model is analyzed from a multi-hop network perspective. We consider two static SD-WSN scenarios to examine scalability, and one scenario to assess the performance implications in a pseudo-dynamic SD-WSN. Extensive simulation results are presented regarding the control overhead introduced, the percentage of alive nodes and remaining energy, and the impacts on network lifetime. We show that the accumulated control overhead is inversely proportional to the interaction period with the controller, whereas the remaining energy and the network lifetime are directly proportional to this parameter. Results show that the control overhead, for static SD-WSNs, can take up 10-29% of the total data flowing to the controller for the large SD-WSN and 6-19% for the small SD-WSN. For a pseudo-dynamic network, the control overhead can take up to two-thirds of the total data sent to the controller, and the network lifetime was reduced by up to 80% compared with the static scenarios.**

***Index Terms*—Energy consumption, software-defined wireless sensor networks, wireless sensor networks, Internet of Things, control overhead.**

I. Introduction

A Wireless sensor network (WSN) is built from a large number of sensor nodes, each embedded with sensors to detect a physical phenomenon, a communication radio to transfer the data of interest, a processing unit to handle calculations, a memory to store information, and a power supply to provide power to the sensor node [1]. The latest advances in Micro-Electro-Mechanical Systems (MEMS) technology has enabled the fabrication of sensor nodes that are inexpensive compare to traditional sensors [2]. Their size, robustness, intelligence, and low power consumption make them attractive in the practical implementation of the Internet of Things (IoT).

The emergence of the IoT paradigm has extended the scope of WSNs. Different types of real-world applications of WSNs such as smart cities, smart agriculture, smart grids, etc., could potentially require the deployment of thousands of nodes [3], [4]. However, WSNs have limited resources, namely *energy*, *processing power*, *memory* and *communication* capabilities. In order to improve the wireless sensor network performance, these resources have to be managed effectively. The efficient management of large-scale WSNs is also a major concern. Wireless sensor nodes are originally designed as an autonomous system, where each sensor node has all the functionalities from the physical- up to application-layer. As the network size grows, the management complexity also grows. By employing effective management systems, the operational cost of the network can be reduced.

In order to solve the complexity of management of WSNs, the Software-Defined Networking (SDN) paradigm has been introduced. SDN allows a new network architecture where the principle is to divide the network into two different planes; the control- and data-planes. The control plane is in charge of executing process- and energy-intensive functions such as routing protocols, whereas the data plane devices act as forwarding devices where packet forwarding is done based on control plane instructions. SDN was originally designed for wired networks, where control messages are sent in a dedicated channel, whereas in wireless networks the medium is shared. Therefore, the introduction of SDN-based architecture can directly affect the network performance metrics such as energy consumption, Packet Delivery Ratio (PDR), delay, and increased control overhead [5].

The software-defined wireless sensor network (SD-WSN) paradigm has emerged as a solution to satisfy the IoT requirements such as management and scalability. Many of the expected billions of devices connected to the internet, as predicted by [6], will need to be managed remotely. For example, at peak hours when some links can get congested, the network should be able to smartly redirect data through less congested links. This may also result in reducing the overall network energy consumption since sensor nodes can avoid re-transmitting packets. Also, for priority tasks, the network topology could be reconfigured in near real-time to provide the best service. In deployments with a large number of sensor nodes, the reconfiguration for a specific task (e.g. pollution monitoring) could be adjusted in a centralized manner and so avoid on-site physical intervention with sensor nodes. Moreover, in network deployments with difficult access, operators may, at any time, choose to prioritize the network lifetime so with SD-WSN energy-efficient routing protocols could be implemented on the fly.

SD-WSN simplifies the creation of new functions and abstractions in the network by moving the most processing- and energy-intensive functions to a centralized controller.

Manuscript received June 28, 2019.

The authors are with the Department of Electrical and Electronic Engineering, The University of Melbourne, Victoria 3010, Australia (e-mail: fjurado@student.unimelb.edu.au, clak@unimelb.edu.au, nirmalat@unimelb.edu.au).

However, it does need a communication control mechanism to maintain the connection with the data plane. The amount of control packets mainly depends on network-specific requirements and the characteristics of the network deployment [5]. However, the potentially negative impacts of SD-WSN in terms of network lifetime and control overhead still need to be studied in detail. To the authors' knowledge, this is the first attempt to study the performance implications of SD-WSN in terms of impacts on control overhead, energy consumption and network lifetime.

A. Contribution

In order to reduce the management complexity of WSNs by making the data plane reprogrammable, it is necessary to introduce a new communication control mechanism. Control packets flowing in the network directly affect the WSN performance in terms of energy consumption and network lifetime. For the first time, this work makes an attempt to examine the real impacts of such a control mechanism in medium to large scale WSN's. The work presented here offers researchers something new: a practical model to build upon in an area previously limited to theoretical studies and proof-of-concepts. To address the aforementioned issues and to quantify the energy consumption and control overhead introduced in an SD-WSN, we propose a model-based characterization of energy consumption for SD-WSNs. In order to mathematically express the energy consumed by the network, we first break down the functions involved, namely; *neighbor discovery*, *neighbor advertisement*, *network configuration* and *data collection*. *Neighbor discovery* is used by sensor nodes to scan and detect any node changes in their neighborhood. Sensor nodes keep the centralized controller updated on the network status by using *neighbor advertisement* messages. The controller reconfigures individual sensor nodes through *configuration packets*, and, lastly, the *sensing data* of interest is collected. Extensive simulation results were carried out to show the impacts affecting the network performance; energy consumption, control overhead and network lifetime. In brief, the *contributions* in this work are as follows;

1) We break down all the processes involved in the correct functioning of the SD-WSN.
2) We provide a mathematical energy model to calculate the energy consumed by the SD-WSN and the control overhead introduced.
3) We perform extensive simulation experiments to quantify the performance implications of SD-WSN in 'small' (50 nodes), 'large' (100 nodes) and 'pseudo-dynamic' WSNs deployments. The last is so-called because it emulates the behavior of a dynamic network by increasing the communication frequency with the controller.

The remainder of this paper is organized as follows. Section II discusses the existing research efforts in SD-WSNs. In Section III, the network and energy model for the WSN is presented. Section IV presents the mathematical model characterization for the energy consumption of SD-WSNs. Section V provides the simulation and result discussion and, finally, in Section VI, the conclusions are drawn.

II. Related Works

The SDN paradigm has been introduced as a novel solution to enable management and real-time reconfiguration of WSN's. Most of the research efforts in SD-WSN have been focused on proof of concept SDN-based WSNs, rather than showing the performance implications of using SDN in a multi-hop WSN.

Bera *et al.* [7] proposed a software-defined wireless sensor network architecture (Soft-WSN) that enables two management policies: device and network management. The device manager controls the sensors and sensing delay. The network management controls the network topology which can be modified during runtime, thus supporting dynamic environments. The experimental evaluation was carried out with five sensor nodes and they evaluated their approach based on Packet Delivery Ratio (PDR), energy consumption and message overhead. Even though their approach outperformed the traditional WSN protocol, it lacked a scalability analysis of the energy consumption and control overhead introduced in small to large network deployments. Luo *et al.* [8] proposed a customized flow-table implementation, named as Sensor-OpenFlow, to tackle the rigidity in policy changes and management. Two distinct flow-table rules were proposed: value- and ID-based. In the value-based rule, forwarding of packets is done by comparing the sensed data and sending only new values. Whereas, in the ID-based approach, the sensor ID is used to forward packets to the sink node. Sensor-OpenFlow was one of the early adopters of SD-WSN and it was meant to pinpoint SDN as a potential solution for the management complexities in WSN. However, no performance metrics were presented. SD-WSN6Lo [9] was proposed to reduce the management complexities in WSNs. The ease of changing the network topology and transmission power through an SDN controller, without any firmware modification, was demonstrated. Although individual energy consumption could be reduced by reducing the transmission power, the paper lacked analysis of energy consumption and control overhead introduced into the network. Misra *et al.* [5] proposed situation-aware protocol switching for SD-WSNs. It consisted of two phases: decision making and protocol deployment. The decision making used a supervised learning approach to select the proper routing protocol that fits the application requirements. The protocol is deployed in the network by the SDN controller in an adaptive manner. The objective of the scheme is to maximize the network performance by minimizing energy consumption and delay, as well as maximizing the PDR and throughput. The performance of the proposed scheme was tested with four routing protocols. The proposed scheme deployed multiple routing protocols based on the decision made by the controller while considering the application-specific requirements.

Other SD-WSN works such as [10], [11], [12] try to improve the network performance by taking into account different performance parameters. In [10] the control information flowing in the network is reduced by programming sensor nodes as Finite State Machines (FSM). This allowed sensor nodes to make decisions, thus reducing the interaction with the controller. In [11], they eliminated the WSN dependency to a single controller, by using multiple controllers. In [12], a statistical machine learning approach is proposed to determine the interference affecting network performance.

As explained above, most of the research efforts in SD-WSN have been focused on demonstrating the advantages that software-defined networks can bring to the performance and management of the WSN, rather than studying the impacts of introducing new abstractions into the network. Our contribution is in providing a mathematical energy model that enables the quantification of the energy consumption and control overhead introduced in the network when making a WSN reprogrammable.

III. System Model

In this section, we describe the network and energy models for a SD-WSN.

A. Network model

To simplify the network model, we have adopted the following assumptions:

- Sensors are uniformly distributed within a square field and the controller is located in the middle of the field.
- All nodes are stationary. (We will examine 'dynamic' deployments in the first instance by simply adjusting the frequency of node messaging as we will see later).
- The energy spent in sensing and processing data is taken as constant across all the network, therefore, is not considered, besides the major component of the energy consumption of sensor nodes is attributed to communication [13].
- All nodes are homogeneous and energy-constrained.
- The controller has access to mains power; therefore its energy consumption is not considered.
- Control packets are routed using the shortest path.
- Nodes are unaware of their location. But they can estimate the distance to neighboring nodes using the Received Signal Strength Indicator (RSSI).
- Nodes sense the environment regularly and send the collected data at a fixed rate.

Based on the control information collected by the sensor nodes, the controller constructs a connected graph $G = (N, E)$ where N is the set of alive nodes on the field and E is the set of communication links between sensor nodes. We use the distance to construct a cost matrix [9].

B. Energy Model

In this work, we refer to a simplified energy model for wireless communications as it is used in [14], [15]. The energy consumed for transmitting a $k - bit$ message to a distance d can be expressed as follows:

$$E_T = E_{elec}^T * k + \epsilon_{amp} * k * d^{\lambda} \tag{1}$$

Where E_{elec}^T refers to the energy dissipated by the circuitry in transmission and ϵ_{amp} is the energy dissipated by the transmit amplifier. d denotes the distance between transmitter and receiver. λ refers to the channel path-loss exponent of the antenna, and this is affected by the surrounding environment and satisfies $2 \leqslant \lambda \leqslant 4$.

TABLE I
List of symbols

Symbol	Description
n	Element of alive nodes
d	distance between Tx and Rx
k	packet size [Bits]
λ	Path-loss exponent
m	Number of hops
E_{elec}	Energy cost of electronics
ϵ_{amp}	Energy cost of transmit amplifier
E_{dis}^n	Energy consumed by node n when performing ND
E_{dis}^{Total}	Total energy consumed by the network in an ND cycle
E_{adv}^n	Energy consumed by node n when performing NA
E_{adv}^{Total}	Total energy consumed by the network in an NA cycle
E_{conf}^n	Energy consumed transmitting a configuration message to node n
E_{conf}^{Total}	Total energy consumed by the network during reconfiguration of all alive nodes
E_{ctrl}^n	Energy consumed, by node n, during control interaction with the controller
E_{data}^n	Energy consumed, by node n, when sending collected data
E_{data}^{Total}	Total energy consumed when sending collected data to the controller
E_{Total}	Total energy consumed by the network

On the other side of the communication link, the energy consumed, by the receiver E_R, to process a $k - bit$ message can be expressed as follows:

$$E_R = E_{elec}^R * k \tag{2}$$

In our model, as considered in [14], [15], [16], we consider that $E_{elec}^R = E_{elec}^T = E_{elec}$ and that only the transmitting node can adjust its transmission power to reach a minimum E_T value.

IV. SD-WSN energy model

In order to estimate the energy consumed by a SD-WSN, we need to break down all the processes involved in the functioning of a WSN with N nodes. Where N is the set of alive nodes and $n \in N$. The definitions of the parameters used in the energy model are summarized in Table I.

A. Neighbor discovery

This is an essential procedure in WSNs. Neighbor discovery (ND) is vital in the initial set-up phase of the network as well as during the network lifetime. In the initial phase, sensor nodes need to discover neighbors and find the best path to the controller. During the other times, ND allows the detection of any changes in the network topology (e.g. due to interference, battery depletion, etc.). The frequency of the ND messages mainly depends on the application-specific requirements. In static WSNs, changes in the topology will not be as frequent as it is in dynamic WSNs. Dynamic WSNs require a higher

ND rate in order to promptly detect, and react to, changes in the network.

Sensor nodes use broadcast messages [9], to discover neighbors within the maximum transmission range. To do this, sensor nodes need to transmit and listen to ND messages.

1) Energy consumed, by node n, during the transmission of ND message: The energy consumed, by node n, for transmitting a $k_{dis}-bit$ discovery message to a distance d_{dis} can be expressed as follows:

$$E^n_{T-dis}(k_{dis}, d_{dis}) = E_{elec} * k_{dis} + \epsilon_{amp} * k_{dis} * d^{\lambda}_{dis} \tag{3}$$

Where d_{dis}, is the maximum distance reached when transmitting at full power.

2) Energy consumed, by node n, during reception of ND message: The energy consumed, by node n, for receiving a $k_{dis}-bit$ discovery message from their neighbors B_n, where $B_n \subseteq N$, is.

$$E^n_{R-dis}(k_{dis}, |B_n|) = E_{elec} * k_{dis} * |B_n| \tag{4}$$

Where k_{dis} is the message size.

3) Energy consumed by node n during a ND cycle: The energy consumed by node n during a ND cycle can be expressed as follows:

$$E^n_{dis} = E^n_{T-dis}(k_{dis}, d_{dis}) + E^n_{R-dis}(k_{dis}, |B_n|) \tag{5}$$

Using equations 3 and 4, the energy consumed by node n during a ND cycle is:

$$E^n_{dis} = E_T(k_{dis}, d_{dis}) + E_R(k_{dis}) * |B_n| \tag{6}$$

4) Total energy consumed by the network during a ND cycle: The total energy consumed by the network when a ND cycle is performed, can be expressed as follows:

$$E^{Total}_{dis} = \sum_n E^n_{dis} \tag{7}$$

We know that k_{dis} and d_{dis} are constants, so

$$E^{Total}_{dis} = |N| * E_T(k_{dis}, d_{dis}) + E_R(k_{dis}) \sum_n |B_n| \tag{8}$$

B. Neighbor advertisement

This is a key procedure in an SD-WSN. The neighbor advertisement (NA) is essential in the initial phase of the network setup as well during the network lifetime. In the initial phase, all sensor nodes advertise their neighbors. The centralized controller makes use of NA messages to build a global view of the network. Based on the application-specific requirements, the controller acts to reconfigure the network to meet those requirements. Sensor nodes also use NA messages to keep the controller updated about any changes in their neighborhood. The frequency of NA messages also depends on application-specific requirements. Sensor nodes can generate control packets either using a reactive or periodical approach.

Sensor nodes use the Shortest Path algorithm [17], [9] to send NA messages to the controller as shown in Fig. 1. Where m is the number of hops to the controller.

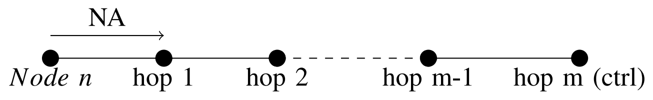

Fig. 1. Neighbor advertisement

1) Energy consumed, by node n, during transmission of NA message: The energy consumed transmitting a $k_{adv}-bit$ NA message to the controller from node n to the controller, can be expressed as follows:

$$E^n_{adv} = \sum_{i=1}^{m_n} E_T(k^n_{adv}, d_i) + \sum_{i=1}^{m_n-1} E_R(k^n_{adv}) \tag{9}$$

Where k^n_{adv} refers to the size of the NA message of node n, which differs from node to node, because they could have a different number of neighbors. $i = \{1, 2, ..., m_n\}$ and m_n refers to the number of hops from node n to the controller. Note that there are $m_n - 1$ receiving nodes since the energy spent by the controller when receiving a packet is not considered as the controller is assumed to have access to mains power. Using equations 1 and 2, we have:

$$E^n_{adv} = \sum_{i=1}^{m_n}(E_{elec} * k^n_{adv} + \epsilon_{amp} * k^n_{adv} * d^{\lambda}_i) + \sum_{i=1}^{m_n-1} E_{elec} * k^n_{adv} \tag{10}$$

We know that E_{elec} and k^n_{adv} are constant for node n, then $E_{elec} * k^n_{adv} \sum_{i=1}^{m_n} 1 = m_n * E_{elec} * k^n_{adv}$. Equation 10 can be rewritten as,

$$E^n_{adv} = m_n * E_{elec} * k^n_{adv} + \epsilon_{amp} * k^n_{adv} \sum_{i=1}^{m_n} d^{\lambda}_i + (m_n - 1) * E_{elec} * k^n_{adv} \tag{11}$$

Which can be simplified to:

$$E^n_{adv} = E_{elec} * k^n_{adv}(2m_n - 1) + \epsilon_{amp} * k^n_{adv} \sum_{i=1}^{m_n} d^{\lambda}_i \tag{12}$$

Thus, the energy consumed when transmitting a NA message from node n to the controller can be expressed as follows:

$$E^n_{adv} = E_R(k^n_{adv}) * (2m_n - 1) + \epsilon_{amp} * k^n_{adv} \sum_{i=1}^{m_n} d^{\lambda}_i \tag{13}$$

Where d_i is the Euclidean distance in R^2 between hop $m_i = (x_i, y_i)$ and hop $m_{i-1} = (x_{i-1}, y_{i-1})$, where $m_0 = (x_0, y_0) = n$. Additionally, k^n_{adv} can be defined as follows:

$$k^n_{adv} = H + P * |B_n| \tag{14}$$

Where H refers to the size of the header in the NA message and P is the number of bits used to represent a single neighbor status.

2) Total energy consumed during transmission of NA messages: During the initial setup phase and over the periodical NA approach, every node in the network sends NA to the controller. Thus, in those cases, the total energy consumed by the network during NA can be expressed as the sum of the individual nodes' energy consumptions.

$$E_{adv}^{Total} = \sum_{n} E_{adv}^{n} \tag{15}$$

Thus, the energy consumed by the network when nodes transmit NA messages to the controller can be expressed as follows:

$$E_{adv}^{Total} = \sum_{n} E_R(k_{adv}^{n}) * (2m_n - 1) + \epsilon_{amp} \sum_{n} \sum_{i=1}^{m_n} k_{adv}^{n} * d_i^{\lambda} \tag{16}$$

C. Configuration packets

The logically centralized controller, which acts as the Network Operating System (NOS) [4], controls and manages the overall behavior of the network. Sensor nodes simply forward packets based on instructions provided by the controller. Sensor nodes are devices with no intelligence since process- and energy-intensive functions, such as routing and management, are handled at the controller [3], [9]. This approach enables the network configuration to be performed globally whereas distributed management schemes require individual reconfiguration of network devices to modify the network behavior [4]. Although SD-WSN offers realtime network configuration and management [18], SD-WSN introduces extra control overhead in the network to make the data plane reconfigurable. Here, we also assume that the controller uses the shortest path algorithm to deliver configuration packets to sensor nodes as shown in Fig. 1, but packets travel in the opposite direction.

1) Energy consumed by reconfiguring node n: The energy consumed transmitting a $k_{conf}^{n} - bit$ control message to sensor node n can be defined as follows:

$$E_{conf}^{n} = \sum_{j=1}^{m_n - 1} E_T(k_{conf}^{n}, d_j) + \sum_{j=1}^{m_n} E_R(k_{conf}^{n}) \tag{17}$$

Where k_{conf}^{n} refers to the size of the control message transmitted to sensor node n, which differs from node to node, because of the number of routes to be reconfigured. k_{conf}^{n} can also be defined as equation 14. $j = \{1, 2, ..., m_n\}$ and m_n refers to the number of hops to sensor node n. Note that there are $m_n - 1$ transmissions since the energy spent by the controller while transmitting a packet is not considered, as previously discussed in the energy consumption of a NA message. Using equations 1 and 2, and following a similar procedure to find equation 13 we have:

$$E_{conf}^{n} = E_R(k_{conf}^{n}) * (2m_n - 1) + \epsilon_{amp} * k_{conf}^{n} \sum_{j=1}^{m_n - 1} d_j^{\lambda} \tag{18}$$

2) Energy consumed reconfiguring the network: During the initial setup phase and over the periodical reconfiguration approach, the controller delivers control packets to all sensor nodes in the network. Thus, the total energy consumed by reconfiguring the network can be expressed as the sum of all nodes' energy consumption during these periods.

$$E_{conf}^{Total} = \sum_{n} E_{conf}^{n} \tag{19}$$

Therefore, the total energy consumed, by the network, during the reconfiguration of all alive nodes in the network can be expressed as:

$$E_{conf}^{Total} = \sum_{n} E_R(k_{conf}^{n}) * (2m_n - 1) + \epsilon_{amp} \sum_{n} \sum_{j=1}^{m_n - 1} k_{conf}^{n} * d_j^{\lambda} \tag{20}$$

D. Control packets

In this work, we consider control-overhead packets as packets that are flowing from sensor nodes to the controller, or vice versa, other than data packets. Control overhead packets are necessary for the correct functioning of the SD-WSN. Thus, the energy consumed by control overhead packets (E_{ctrl}^{n}) is the summation of the energy consumed by NA and network configuration packets. Mathematically,

$$E_{ctrl}^{n} = E_{adv}^{n} + E_{conf}^{n} \tag{21}$$

Thus, using equations 13 and 18, the energy consumed, by the network, in control-overhead packets can be calculated as:

$$\begin{aligned} E_{sd}^{n} &= E_R(k_{adv}^{n}) * (2m_n - 1) + \epsilon_{amp} * k_{adv}^{n} \sum_{i=1}^{m_n} d_i^{\lambda} \\ &+ E_R(k_{conf}^{n}) * (2m_n - 1) + \epsilon_{amp} * k_{conf}^{n} \sum_{j=1}^{m_n - 1} d_j^{\lambda} \end{aligned} \tag{22}$$

The calculation of the energy consumed by control-overhead packets can be simplified if the size of NA and configuration messages are equal. We know that $\sum_{i=1}^{m_n} d_i^{\lambda} = \sum_{i=1}^{m_n - 1} d_i^{\lambda} + d_{m_n}^{\lambda}$, where d_{m_n} is the Euclidean distance between the hop m (controller) and the previous hop. Mathematically,

$$\begin{aligned} E_{sd}^{n} &= 2E_R(k_{sd}^{n})(2m_n - 1) + 2\epsilon_{amp} * k_{sd}^{n} \sum_{i=1}^{m_n - 1} d_i^{\lambda} \\ &+ \epsilon_{amp} * k_{sd}^{n} * d_{m_n}^{\lambda} \end{aligned} \tag{23}$$

Where $k_{sd}^{n} = k_{adv}^{n} = k_{conf}^{n}$ and $i = j$.

E. Data Packets

The information gathered by the WSN needs to be shared with the centralized controller for further processing. Sensor nodes use the routing algorithm, previously reconfigured by the controller, to send the collected data to the controller. In a multi-hop network, the collected data has to travel through multiple hops before reaching the controller. The data size depends on application-specific requirements. Here, we assume that all nodes send the same data size.

1) Energy consumed when sending data-packets: The energy consumed by the network when node n sends the collected data to the controller can be calculated as follows:

$$E_{data}^{n} = E_R(k_{data}) * (2m_n - 1) + \epsilon_{amp} * k_{data} \sum_{i=1}^{m_n} d_i^{\lambda} \tag{24}$$

Depending on the routing algorithm programmed by the controller, the number of hops (m_n) and the distance between hops (d_i) can vary, so too can the energy consumed by the network.

2) Total energy consumed when sending data-packets: In networks that are periodically sensing the phenomenon of interest, the collected data must be sent to the controller in a specific time interval. Thus, the total energy consumed by the network when all alive nodes send their sensed data can be calculated as:

$$E_{data}^{Total} = \sum_{n} E_R(k_{data}) * (2m_n - 1) + \epsilon_{amp} \sum_{n} \sum_{i=1}^{m_n} k_{data} * d_i^{\lambda} \tag{25}$$

F. Total Energy

Lastly, the total energy consumed by the SD-WSN can be calculated by summing up all the functions or processes involved in the functioning of the SD-WSN.

$$E_{Total} = E_{dis}^{Total} + E_{adv}^{Total} + E_{conf}^{Total} + E_{data}^{Total} \tag{26}$$

As mentioned earlier, all packets flowing from sensor nodes to controller, or vice versa, other than data packets are considered as control packets. Thus, the total energy consumed by control packets (E_{ctrl}^{Total}) is $E_{adv}^{Total} + E_{conf}^{Total}$.

V. Simulation and Results

Even for relatively small-sized WSNs with tens of nodes, it is difficult to analyze the interactions between all sensor nodes involved [19], [20]. Thus, we use computer-based simulation software to evaluate the performance implications of using software-defined networking in terms of energy consumption and control overhead. In this work, we use MATLAB to simulate the SD-WSN with two different network sizes. We chose MATLAB because it is one of the most commonly used simulators and it provides a numeric power consumption model, which can be easily adapted to fit different scenarios, the processing time is quick for large networks, it is not constrained to a specific type of sensors, and for other features as stated in [21]. MATLAB was used in many of the prior works when only the energy consumption was being evaluated [21].

A. Simulation setup

The simulation experiments were run on high-performance computing infrastructure [22]. The SD-WSN was tested under 'small' (50 node) and 'large' (100 node) network deployments. These numbers were chosen as giving reasonable simulation times for comparisons to be drawn, as we found that even our high-performance computing infrastructure struggled with several hundred nodes. The E_{elec} and ϵ_{amp} parameters are set as $50{\times}10^{-9}$ J/bit and $100{\times}10^{-12}$ $J/bit/m^2$, respectively [14]. For simplicity, we consider that the path-loss exponent (τ) is always constant and equal to 2, as used in [16], [14], [15]. The simulation parameters are summarized in Table II.

TABLE II
SIMULATION PARAMETERS

Parameter	Value
WSN area	$50 \times 50{,}100 \times 100$ m^2
Node distribution	Random-uniformly distributed
Controller position	$(25, 25)$ and $(50, 50)$
Sensor nodes	$50, 100$
Max. Tx. range	$20, 40$ m
Initial energy	0.5 J
E_{elec}	50×10^{-9} J/bit [14]
ϵ_{amp}	100×10^{-12} $J/bit/m^2$ [14]
Data rate	1 $packet/min$
ND period	5 mins
NA period (NAP)	5, 10, 15, 20 mins
B_n	$variable$
k_{adv}^n	H = 8 Bytes, P_{adv} = 5 Bytes and $\lvert B_n \rvert$: variable, thus k_{adv}^n ≥13 Bytes
k_{conf}^n	H = 8 Bytes, P_{conf} = 4 Bytes and max 20 routes
k_{dis}	3 $bytes$
k_{data}	72 $bytes$ [23]

The location of the controller directly impacts the WSN performance. Among the performance parameters to optimize are the energy and lifetime. The controller can be located in a specific part of the network to reduce the energy consumed by sensor nodes or to extend the network lifetime. The energy-oriented approach finds the optimal position of the controller which makes the energy consumption minimum. However, it usually is not the optimal solution for the network lifetime since the placement for the energy-oriented solution can be found in a low node density area, which results in insufficient resources (sensor nodes) in the controller neighborhood [24]. Therefore, sensor nodes in proximity to the controller can exhaust their energy first, resulting in a shorter network lifetime. Since the aim of this work is not to find the optimal location of the controller but to present the impacts of the software-defined network approach in WSN then we place the controller in the middle of the WSN area.

k_{data} consists of 72 data bytes collected by sensor nodes such us; battery level, temperature, humidity, pressure, luminosity and acceleration. The data packet consists of a 10-byte header, a 2-byte Cyclic Redundancy Check (CRC) and 60 information bytes that have sensing parameters [23].

k_{dis} are small packets that only contain information about the node rank (1 byte) and the sum of RSSI (2 bytes) values received from the controller. In total, only three bytes are needed so nodes can choose the best RSSI path to the controller between two equal ranks.

The k_{adv}^n parameter depends on the number of neighbors and the header has 8 bytes distributed as: control packet type (1 byte), rank of node (1 byte), remain energy of node (2 bytes), packet length (2 bytes) and CRC (2 bytes). P_{adv} (payload) contains the node address (2 bytes), RSSI (2 bytes) and rank (1 byte).

The k_{conf}^n parameter contains the same header as k_{adv}^n and P_{conf} (payload) has the destination (2 bytes) and forwarding addresses (2 bytes). Also, we assume that sensor nodes can maintain up to 20 routes in their routing table. In the worst-case scenario, the controller may need to reconfigure the entire routing table.

For static WSNs, neighbor discovery is typically performed every 5 mins whereas, in dynamic WSNs, the neighbor discovery is performed perhaps up to twice a minute [25]. Since sensor nodes for environmental monitoring are often static, then we initially set the ND period as 5 mins.

In this work, we consider that sensor nodes generate NA packets in a periodical approach. The period of NA packets depends on the application-specific requirements. Here, we vary the period of NA packets to observe the impacts of control overhead in the SD-WSN. Since the ND period is set to 5 mins, a sensor node cannot detect a change in its neighborhood in a shorter time. Therefore, the period of control packets has to be greater or equal to 5 mins. We will later examine the results for periods of 5, 10, 15 and 20 mins. Configuration packets are sent by the controller every time it detects a change in the network topology. We generate configuration packets to all nodes every time a sensor node dies.

B. Performance metrics

We defined three performance metrics to evaluate the impacts of software-defined network management approaches in WSNs.

1) *Control overhead:* We have defined control overhead as the number of NA and configuration packets. We measure the control overhead during the entire simulation time. This allows us to observe the behavior of control overhead packets introduced in a SD-WSN.
2) *Alive nodes:* This metric reflects the number of sensor nodes that have not yet exhausted all of their energy.
3) *Network lifetime (NL):* We define the NL as the time taken from the deployment until the instant a network partition occurs. Note that even when a network partition occurs, there may be energy remaining in the network.

C. Simulation and analysis

The simulation experiments are performed based on the performance metrics previously discussed in Section V-B. We have considered two scenarios: (i) The first scenario is a 'large' SD-WSN deployed in a $10{,}000$ square meter area containing 100 sensor nodes randomly distributed and a controller located centrally. (ii) The second scenario is a 'small' SD-WSN deployed in a $2{,}500$ square meter area, containing 50 sensor nodes randomly distributed, with a controller located centrally, but note that the node density is twice that of the large SD-WSN. In both cases, the controller's transmitter covers approximately 50% of the total area. Nodes outside this area have to reach the controller through multiple hops. Since nodes have programmable transmission power, we limit the transmission range of nodes in the small scenario to allow us to create a multi-hop network. Other parameters used are shown in Table II. At the end of this section, we also compare the large SD-WSN scenario with a large *'pseudo-dynamic'* SD-WSN, to get some insight into the effects of reducing both the ND and NA periods, although note that the nodes do not actually move in this scenario as this would make the model too complex to simulate.

1) Static SD-WSN: Fig. 2 shows the control overhead and the collected data for the large SD-WSN. It can be observed that the *control overhead* is inversely proportional to the NAP, and that the total collected data is over twice the total control overhead for NAP=5 mins. It grows to over 9 times the amount of the total control overhead for NAP=20 mins. The total number of bits flowing in the network is mainly dictated by the

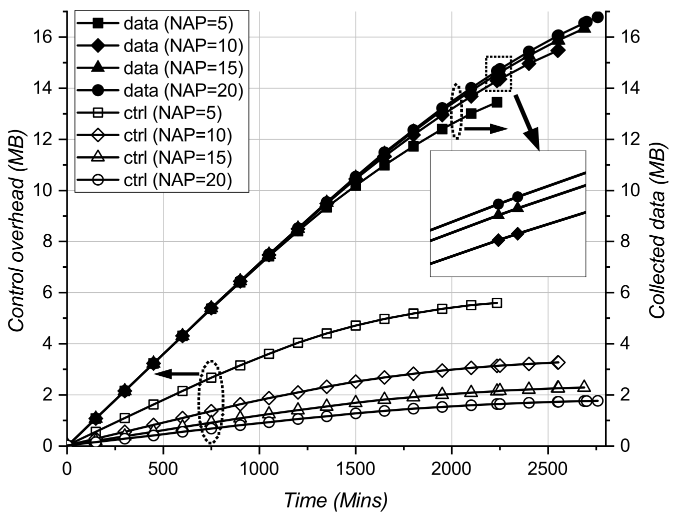

Fig. 2. Accumulated control overhead vs. accumulated collected data for large SD-WSN.

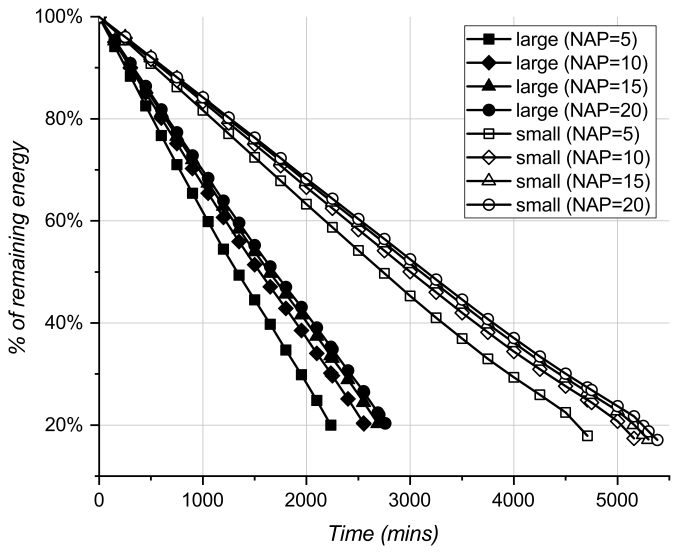

Fig. 3. Remaining energy of large vs. small SD-WSN over simulation time.

data packet rate, and this parameter depends on the particular application. Note that the control overhead can be reduced by adopting the reactive approach, where nodes reduce their interaction with the controller by only sending NA packets when a change in the network is detected, or when they receive a packet whose destination is unknown.

As shown in Fig. 3 and Table III, the *remaining energy* and the *network lifetime* are directly proportional to the NAP as might be expected, but we can see the precise scale of it here for the first time. As the NAP increases, less energy is expended in the network and longer network lifetimes result. For example, an application with a NAP of 5 mins will reduce the network lifetime by 19% compared with an application with a NAP of 20 mins for a large SD-WSN and by 13% for a small SD-WSN. In addition, it can be seen that the shortest path protocol does not exhaust all the energy available in the network as the network becomes partitioned. Fig. 3 shows that approximately 20% of the network energy remains in both

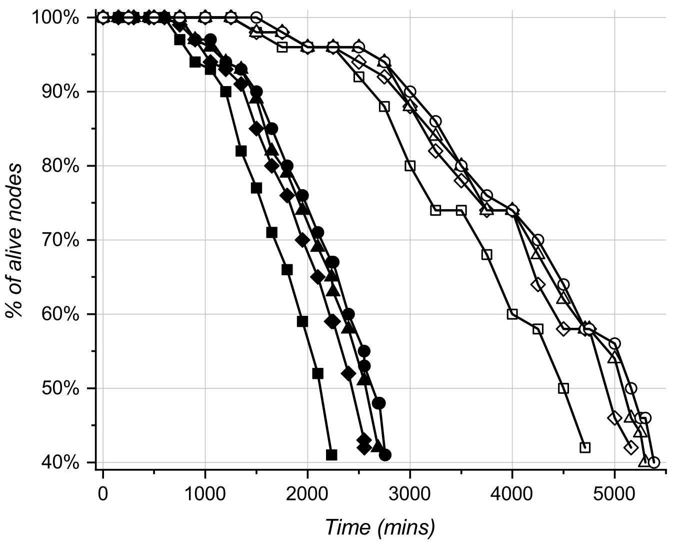

Fig. 4. Alive nodes of large vs. small SD-WSN over simulation time.(Same key as Fig. 3)

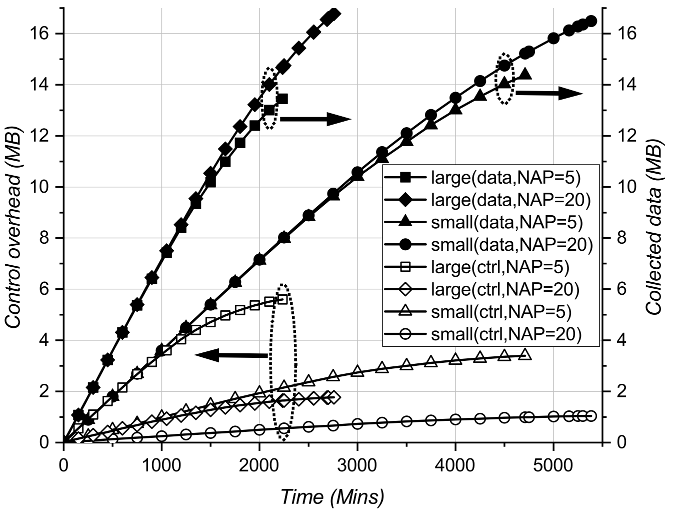

Fig. 5. Large vs. small SD-WSN accumulated control overhead and collected data.

scenarios after partition occurs for all values of NAP. Thus, further research on energy-efficient SD-WSN routing protocols needs to be done to make the most of the remaining energy.

The number of *alive nodes*, as shown in Fig. 4, decays faster, as one might expect, as the interaction with the controller increases (smaller NAPs). In dynamic environments, sensor nodes increase the frequency of communication with the controller to inform it of any changes in the network topology.

TABLE III
SD-WSN LIFETIME

Scenario	**NAP [mins]**						**Units**
	1	**3**	**5**	**10**	**15**	**20**	
large	x	x	37	42	45	46	Hours
small	x	x	78	86	88	90	
pseudo-dyn	16	25	28	31	32	33	

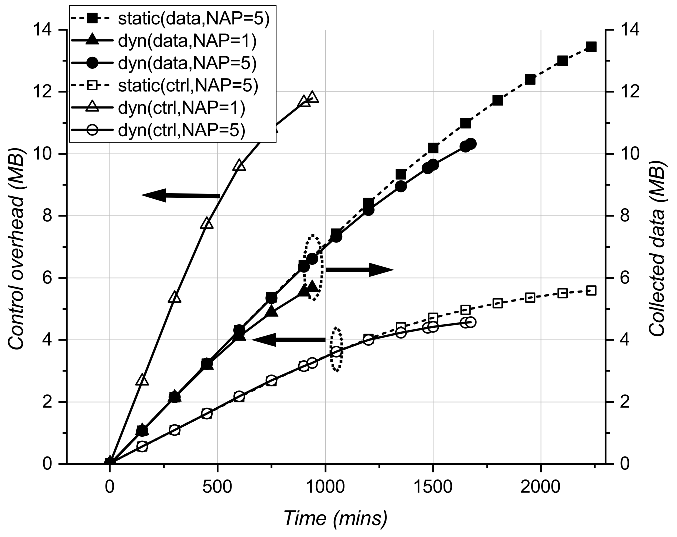

Fig. 6. Pseudo-dynamic vs. static SD-WSN accumulated control overhead and collected data.

Therefore, the percentage of alive nodes will decay faster than in static environments. Fig. 4 shows that the small SD-WSNs have a lifetime approximately twice that of the large SD-WSNs before the first node dies. As the number of sensor nodes in the small scenario is less, sensor nodes close to the controller have to forward fewer control and data packets. The sensor density in the small network is twice that of the large scenario which also helps maintain network lifetime as there are more routing options to share the load.

The results of varying the NAP for the small SD-WSN scenario are not shown here due to space constraints but we rather compare in Fig. 5 the control overhead and the collected data for both large and small SD-WSN scenarios. The control overhead and the collected data scale up faster for the large SD-WSN than for the small SD-WSN. This is due to the number of sensor nodes interacting with the controller. The network lifetime is affected by the number of sensor nodes and node density as the large SD-WSN get partitioned faster than the small SD-WSN as shown in Fig. 5 and Table III. The network lifetimes and the control overhead for the small SD-WSN are shown in Table III and Table IV, respectively.

2) Static vs. pseudo-dynamic large SD-WSN: For this comparison, we have set the ND period to be 30 seconds and the NA period starting from 1 min up to 20 mins. Fig. 6 shows the accumulated control overhead and the collected data for the large *pseudo-dynamic* WSN with NAP of 1 and 5 mins and the large static WSN with NAP=5 mins. From Fig. 6 and Table IV, it can be seen that for the large pseudo-dynamic network with NAP=1 min, the control overhead scales up faster and it is bigger than the collected data at all times. This represents, in a simplified fashion, a highly dynamic environment where sensor nodes might be moving and the controller has to be constantly informed of changes in the topology. In this scenario, the accumulated control overhead is twice the size of the accumulated collected data.

The number of alive nodes (Fig. 7) also started decaying faster than for the static WSN, as expected. The static WSN

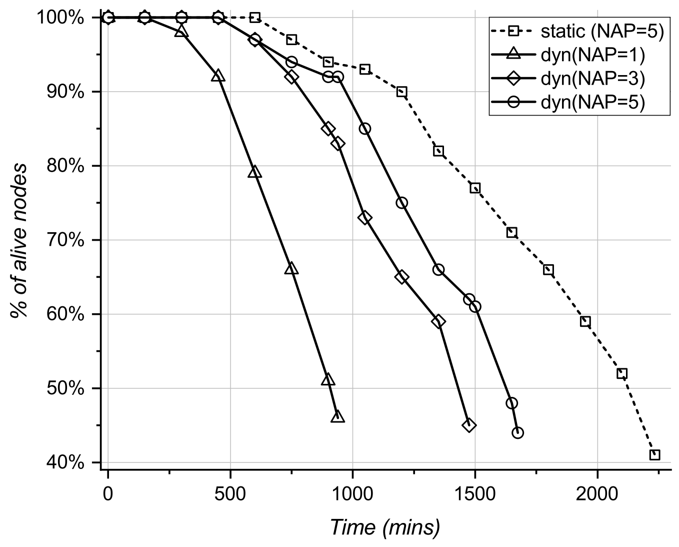

Fig. 7. Alive nodes of pseudo-dynamic vs. static SD-WSN over simulation time.

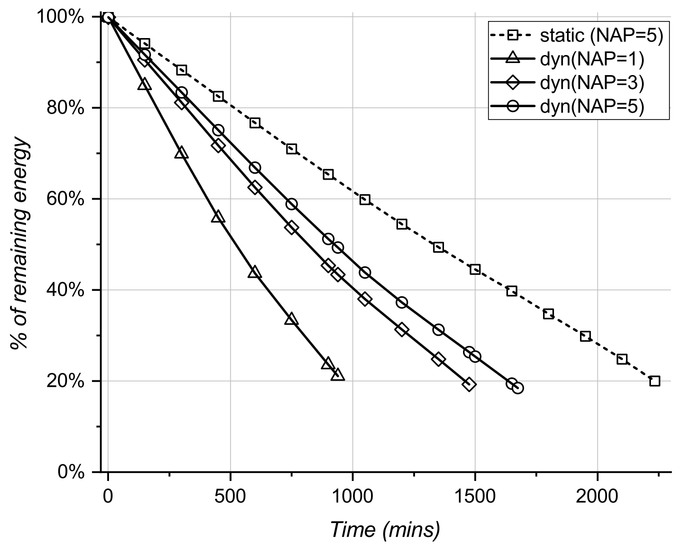

Fig. 8. Remaining energy of pseudo-dynamic vs. static SD-WSN over simulation time.

lasted 4 times longer without any dead nodes. From Fig. 8, it can be clearly seen that the large pseudo-dynamic SD-WSN lasts for less time than all the other scenarios. The network lifetime is shown in Table III.

In general, the best performance was achieved by the small SD-WSN (Fig. 3) that has double the node density in comparison to the other scenarios. The controller and nodes have more resources in their neighborhood to send and receive from other nodes in the network. The worst performance was achieved by the pseudo-dynamic scenario as shown in Fig. 6, Table IV and Table V.

VI. Conclusion

In this paper, we proposed a model-based characterization of the energy consumption in a SD-WSN architecture for the first time. We included each process required for the correct functioning of the SD-WSN. The model was built based on individual energy analysis of single sensor nodes for each process, and the analysis was extended to calculate the overall network energy consumption. Two different scenarios were studied to examine the scalability impacts in the network performance, and one additional scenario to show the likely effects of a highly dynamic network. Results show that for the static SD-WSN scenarios, the control overhead can take up $10-29\%$ of the total data flowing to the controller for the large SD-WSN and $6-19\%$ for the small SD-WSN. The number of bits flowing in the network is mainly dictated by the rate of the collected data, which is a parameter defined specifically by the application. Moreover, the small SD-WSN lasted longer than the large SD-WSN due to the lower number of sensor nodes in the network and the higher sensor density. The small SD-WSN has more resources (nodes) in its neighborhood and the number of control packets was less as the number of sensor nodes was reduced. For the pseudo-dynamic network (NAP=1), the control overhead can consume over two-thirds of the total data sent to the controller due to the controller is being constantly informed of changes in the topology. The network lifetime was reduced by 58% and 80% in comparison with the large and small static SD-WSN, respectively.

This work is the first attempt to explore the actual performance involved in making WSNs reprogrammable. Future works include refining our model further to turn it into a tool suitable for practical SD-WSN performance comparisons, and extending the analysis in hardware and including real-world effects of interference, etc. on the packet delivery ratio of the network.

TABLE IV
Total data sent to controller

Scenario	Total	
	Data	**Control**
large (NAP=20-5 mins)	90-71%	10-29%
small (NAP=20-5 mins)	94-81%	6-19%
pseudo-dyn (NAP=5-1 mins)	69-33%	31-67%

TABLE V
Lifetime reduction for the pseudo-dynamic SD-WSN

Scenario		large (NAP=5)	small (NAP=5)
pseudo-dyn	NAP=1	-58% (21 h)	-80% (62 h)
	NAP=5	-25% (9 h)	-65% (51 h)

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[4] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software defined networking for improved wireless sensor network management: A survey," *Sensors*, vol. 17, no. 5:1031, pp. 1–32, 2017.

[5] S. Misra, S. Bera, M. Achuthananda, S. K. Pal, and M. S. Obaidat, “Situation-aware protocol switching in software-defined wireless sensor network systems,” *IEEE Systems Journal*, vol. 12, no. 3, pp. 2353–2360, 2018.

[6] F. Computing, “The Internet of Things: Extend the cloud to where the things are,” Cisco Syst., San Jose, CA, USA, Report, 2016.

[7] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, “Soft-WSN: Software-defined WSN management system for IoT applications,” *IEEE Systems Journal*, 2016.

[8] T. Luo, H.-P. Tan, and T. Q. Quek, “Sensor OpenFlow: Enabling software-defined wireless sensor networks,” *IEEE Communications letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[9] F. F. J. Lasso, K. Clarke, and A. Nirmalathas, “A software-defined networking framework for IoT based on 6LoWPAN,” in *Wireless Telecommunications Symposium (WTS), 2018*. IEEE, Conference Proceedings, pp. 1–7.

[10] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks,” in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, Conference Proceedings, pp. 513–521.

[11] B. T. De Oliveira, L. B. Gabriel, and C. B. Margi, “TinySDN: Enabling multiple controllers for software-defined wireless sensor networks,” *IEEE Latin America Transactions*, vol. 13, no. 11, pp. 3690–3696, 2015.

[12] C. Orfanidis, “Ph. D. forum abstract: Increasing robustness in WSN using software defined network architecture,” in *Information Processing in Sensor Networks (IPSN), 2016 15th ACM/IEEE International Conference on*. IEEE, Conference Proceedings, pp. 1–2.

[13] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, “Energy-efficient routing protocols in wireless sensor networks: A survey,” *IEEE Communications surveys & tutorials*, vol. 15, no. 2, pp. 551–591, 2012.

[14] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd annual Hawaii international conference on system sciences*. IEEE, Conference Proceedings, p. 10 pp. vol. 2.

[15] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Transactions on wireless communications*, vol. 1, no. 4, pp. 660–670, 2002.

[16] J. Luo, J. Hu, D. Wu, and R. Li, “Opportunistic routing algorithm for relay node selection in wireless sensor networks,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 1, pp. 112–121, 2015.

[17] R. K. Ahuja, *Network flows: theory, algorithms, and applications*. Pearson Education, 2017.

[18] S. Bera, S. Misra, and A. V. Vasilakos, “Software-defined networking for Internet of Things: A survey,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[19] P. Bahl, V. N. Padmanabhan, V. Bahl, and V. Padmanabhan, “RADAR: An in-building RF-based user location and tracking system,” 2000.

[20] Y. Liu, K. Xu, Z. Luo, and L. Chen, “A reliable clustering algorithm base on LEACH protocol in wireless mobile sensor networks,” in *2010 International Conference on Mechanical and Electrical Technology*. IEEE, Conference Proceedings, pp. 692–696.

[21] L. Xu, R. Collier, and G. M. O’Hare, “A survey of clustering techniques in WSNs and consideration of the challenges of applying such to 5G IoT scenarios,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1229–1249, 2017.

[22] L. Lafayette, G. Sauter, L. Vu, and B. Meade, “Spartan performance and flexibility: an HPC-cloud chimera,” 2016.

[23] M. Rodriguez-Sanchez, S. Borromeo, and J. Hernandez-Tamames, “Wireless sensor networks for conservation and monitoring cultural assets,” *IEEE Sensors Journal*, vol. 11, no. 6, pp. 1382–1389, 2011.

[24] F. Chen and R. Li, “Single sink node placement strategy in wireless sensor networks,” in *2011 International Conference on Electric Information and Control Engineering*. IEEE, Conference Proceedings, pp. 1700–1703.

[25] H. Pham and S. Jha, “Addressing mobility in wireless sensor media access protocol,” *International Journal of Distributed Sensor Networks*, vol. 1, no. 2, pp. 269–280, 2005.

F. Fernando Jurado-Lasso (GSM’18) received the B.Sc. degree in electronic engineering from the Universidad del Valle, Cali, Colombia, in 2012. He also received the M.Sc. in Telecommunications Engineering degree from The University of Melbourne, Australia, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronic Engineering, University of Melbourne.

His research interests include software-defined particularly focus on wireless sensor networks, protocols and applications for the Internet of Things.

Ken Clarke received his B.Sc.(hons) in Applied Physics from Heriot-Watt University, Edinburgh, Scotland, in 1985.

He worked in the optoelectronic and telecommunications industries in various R&D and engineering roles in both the UK and Australia from 1985-2010, before moving to the University of Melbourne where he is currently Deputy Director of the Networked Society Institute. He has published over 50 articles and book chapters, and produced five patents.

Ampalavanapillai Nirmalathas (M’98 - SM’03) received the B.Eng. and Ph.D. degrees in electrical engineering from The University of Melbourne, Australia, in 1993 and 1998, respectively. He is currently a Professor with the Electrical and Electronic Engineering Department, The University of Melbourne.

His research interests include microwave photonics, optical wireless network integration, broadband networks, and stability of Internet and telecom services. He is a member of OSA and a Fellow of the Institution of Engineers Australia.