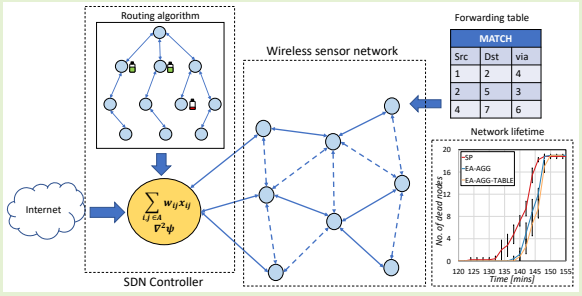
Energy-aware routing for software-defined multihop wireless sensor networks

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

***Abstract*— In this paper, we propose an energy-aware routing algorithm and a control overhead reduction technique for prolonging the network lifetime of software-defined multihop wireless sensor networks (SDWSNs). This is an effort to optimize the energy consumption of WSNs that provide services to the Industrial Internet of Things (IIoT). A centralized controller grants a global view of the sensor network by introducing extra control overhead in the network, but this leads to extra energy costs. However, our new algorithm takes advantage of this global view and balances the network energy by selecting paths with the highest remaining energy level among multiple paths for each sensor node. We also identify key functions draining energy from the SDWSN and minimize their impact by implementing a data packet aggregation function, and minimizing the control overhead by keeping track of the sensor nodes' routing tables using a simple checksum function. We show that the proposed approach prolongs the network lifetime of the WSN by 6.5% on average compared to the standard shortest-path algorithm, and that the control overhead is reduced by approximately 12% while still maintaining a very high packet delivery ratio.**



***Index Terms*— Centralized routing, Industrial Internet of Things, network lifetime, software defined wireless sensor networks, software-defined routing, wireless sensor networks.**

I. Introduction

THE Internet of Things (IoT) paradigm is realized upon the interconnection of computing devices installed in everyday objects, enabling the exchange of information to provide services to users [1]. As estimated by [2], billions of IoT devices are expected to be interconnected to the network by the beginning of this decade. This paradigm is envisioned to soak through into the industrial manufacturing and production, creating the Industrial IoT (IIoT), that is anticipated to contribute to the economic growth and fast-pace production to various manufacturing processes [3]. Most IIoT applications including smart cities, smart grids, smart agriculture, smart homes, etc., will be built upon sensors and actuators. The networking of these devices have expanded the scope of networked sensing technologies such as Wireless Sensor Networks (WSNs).

A WSN is a set of spatially distributed sensor devices used for: monitoring the physical conditions of the environment and sending the collected information in a centralised manner. Individual sensor device, often called "sensor node", have limited resources including central processing unit, communication bandwidth, memory, and power source. Novel network architectures are required to extend the network lifetime and minimize the resource management complexities currently found in WSNs. Currently, sensor nodes are used as autonomous systems, meaning each individual sensor node has embedded all the components required to operate and make decisions without supervision. As a consequence, the management of the limited resources of the network becomes challenging and increases as the network size grows. Recently, researchers have proposed Software-Defined Networking (SDN) as a potential pathway to overcoming the above-mentioned challenges.

SDN is a networking paradigm that divide the network in three different planes: application-, control- and data-plane. The control plane is centralized and performs the most process- and energy-intensive functions, whereas the data plane simply forwards packets based on the controller's instructions. The introduction of SDN concepts into WSNs creates Software-Defined Wireless Sensor Networks (SDWSNs).

The SDWSN paradigm has emerged for Low-Rate Wireless Personal Area Networks (LR-WPANs) [4]. An SDWSN allows dynamic reconfiguration of the functionalities or behaviors of sensor nodes based on network-specific or network-application requirements. In wired SDN, control packets mostly travel in an exclusive control channel, while in wireless SDN, the communication medium is shared. Thus, further control

Manuscript received July 30, 2020.

F. Fernando Jurado-Lasso, Ken Clarke and Ampalavanapillai Nirmalathas are with the Department of Electrical and Electronic Engineering, The University of Melbourne,Victoria 3010, Australia (e-mail: fjuradolasso@unimelb.edu.au, clak@unimelb.edu.au, nirmalat@unimelb.edu.au).

Andres Navarro Cadavid is with the Department of Information and Communications Technologies. ICESI University, Cali, Colombia. (e-mail: anavarro@icesi.edu.co)

overhead is needed in the latter to enable communication with the controller. As in most WSNs, the impacts on Key Performance Indicators (KPIs) including control overhead, data packet delivery ratio (PDR) and energy consumption are a concern when implementing SDN abstractions into WSNs.

The replacement or manipulation of sensor nodes is generally difficult and expensive [5], [6]. Consequently, it is expected that sensor nodes run without battery replacement for prolonged periods of time. However, introducing SDN concepts in WSNs may lead to high energy cost due to additional functions such as: *data collection*, *neighbor advertisement* and *network configuration* [7]. Thus, choosing the most energy-efficient path to the controller, and minimizing the interaction with sensor nodes is critical to extending the network lifetime of the SDWSN.

A. Contribution

Given the above concerns, we focus on finding the most energy-efficient path for *data* and *neighbor advertisements* packets while reducing the number of *network configuration* packets in the SDWSN. This paper presents a novel *energy-aware routing algorithm* and a *control overhead reduction* technique for enhancing the overall network lifetime of the SDWSN while maintaining the PDR high. Firstly, we propose a software-defined routing algorithm based on the shortest path in terms of energy consumption. Secondly, we propose to minimize the interaction with the controller by adding a data-aggregation function into the routing protocol. Lastly, we further minimize the control overhead by keeping track of the sensor nodes' routing table using a checksum function. In order to evaluate these concepts, we performed multiple emulation runs in Contiki Cooja [8], to exhibit the elements influencing in KPIs which are: energy consumption, control overhead, network lifetime and packet delivery ratio. In brief, the *contributions* of this work are summarized below.

1) We propose a software-defined energy-aware routing algorithm for WSNs.
2) We minimize the interaction with the controller using a data-aggregation function.
3) We minimize the SDWSN control overhead by keeping track of sensor nodes' routing table.
4) We evaluate our approach by emulating the routing protocol using Contiki OS and Cooja network simulator.

The remainder of this research is structured as follows. Section II relates to prior research works in software-defined routing protocols. Section III presents a thorough description of the SDWSN architecture and packet formats. In Section IV, we introduce our energy-aware routing algorithm for SDWSNs and the data-aggregation function. In Section V, the minimization of control overhead is described. In Section VI, we present the experimental layout and results. Lastly, Section VII draws the conclusions and presents potential areas for future research.

II. Related works

The SDWNS paradigm has been proposed as a potential pathway to alleviate inherent problems of WSNs such as rigidity in run-time reconfiguration and management [4]. The centralized view of the WSN permits the controller to promptly act to any topology change in the network. Most research efforts in topology reconfiguration have been carried out via numerical analysis rather than emulating the wireless sensor network as we will show below.

There are multiple research efforts in reducing the energy consumption of WSNs based on clustering. Heinzelman *et al.* [9] presented LEACH-C; a centralized low-energy adaptive clustering hierarchy protocol. A base station (BS) distributes the energy load evenly among nodes. It uses a simulated annealing algorithm to find the k optimal cluster, which is an NP-hard problem. The algorithm attempts to minimize the energy consumption of the non-cluster nodes in transmitting packets to the cluster head. This work assumed that every node is within the communication range of all other nodes and the BS, which is not a practical solution in IIoT. Xiang *et al.* [6] put forward an energy-efficient routing algorithm for SDWSNs. Their architecture consisted of a control server, and sensor nodes divided into clusters where there is one control node per cluster. The main server decides the controlling node for every cluster. They manage intra-cluster nodes to perform different jobs. The selection of controlling nodes considers the remaining energy and communication range of sensor nodes. The solution to this problem is NP-hard. They proposed a particle swarm optimization algorithm that solves the selection of controlling nodes. The algorithm was capable of prolonging the network lifetime of the WSN. However, the conclusions were based on numerical analysis without considering other practical factors affecting the network performance such as control overhead, retransmissions, PDR and end-to-end delay. Din *et al.* [10] presented a scheme for energy-efficient topology management based on clustering. They proposed a multilayered clustering architecture for changing cluster heads, as well as the selection of forwarding nodes, and inter- and intra-clustering routing. The change of the forwarding node is done by implementing a routing table at each node. The results show that the presented approach uses less energy, but they presented no evidence for network lifetime extension and their study also lacked control overhead analysis.

Research efforts on energy-efficiency based on energy levels have also been investigated. Wenxing *et al.* [11] presented the idea of a multidimensional energy-space based on residual energy. Sensor nodes with less remaining energy level are moved down to a lower energy-space dimension with different transmissions principles. Although simulations showed that the proposed approach could balance the overall energy expenditure and extend the network lifetime, the paper assumed a perfect Media Access Control (MAC) layer. Their simulation is numerically based, which does not capture practical limitations influencing the performance of the SDWSN. Bo *et al.* [12] presented a controller, building the network topology based upon the energy levels of nodes and distances between nodes. Their numerical simulation shows that using energy levels improved the network's lifetime. However, the major concerns of transmission delay and energy expenditure of the extra overhead were not addressed.

Other works such as [13], [14] seek to reduce interaction and dependency on a single controller. In [13] they try

to reduce the control overhead by configuring the network infrastructure as Finite State Machines (FSM), whereas in [14] they removed the single controller dependency using multiple controllers. However, these studies present no evidence for improved network lifetimes of the SDWSN.

As shown above, prior research efforts in topology reconfiguration in SDWSN often utilize numerical analysis, which ignores major factors influencing the performance of the SDWSN. Our contribution is in providing a novel routing protocol for SDWSNs that prolongs the network lifetime by exploiting the maximum remaining energy path and minimizing the number of data- and control-overhead packets in the network, while incorporating these practical performance limitations. In order to achieve this, experiments were conducted using Contiki OS [15] and a Cooja network simulator [8] which we describe in detail later.

III. SDWSN ARCHITECTURE

In this section, we describe the network model, packet formats, and all tables managed by the controller.

A. Network model

The feasibility and performance of SDWSNs have been demonstrated in [13], [16]–[19]. In common with traditional WSNs, the controller and sensor nodes have a wireless communication transceiver to communicate with each other.

Below we will present our novel energy-aware routing protocol for SDWSNs where the collected- and control-data are transmitted to the controller in a multihop fashion, and the SDWSN is formed as a hierarchical tree structure. Sensor nodes report their own and neighbor's information status to the controller, which builds up a comprehensive view of the WSN. Depending on the network application, the controller reconfigures the sensor nodes' routing table accordingly. As would be expected, the traffic coming from multiple nodes, as well as the controller, will exhaust the energy of a common node faster. Therefore, balancing the energy consumption across sensor nodes, and reducing the overall number interactions with the controller is the main focus of this research paper. We have adopted the below assumptions:

- All nodes in the SDWSN are static.
- Sensor nodes are energy-constrained.
- The initial energy is equal for every sensor node in the WSN.
- The controller has access to mains power, thus we do not consider the energy consumed by this node.
- Sensor nodes are unaware of their location.

The controller uses *neighbor advertisement* packets received from sensor nodes to construct a connected graph $G = (N, L)$, where N is the set of alive nodes and L is the set of communication links among sensor nodes. It then computes the energy-aware routing algorithm and transmits *network configuration* packets to reconfigure the network topology if required.

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	0	Ver.			Agg.		HL			Total Length								Time to Live								Protocol							
4	32	Header Checksum																Source Address															
8	64	Destination Address																Options															

Fig. 1: SDWSN forwarding packet format.

TABLE I: Description of fields in the header of the forwarding packet

Field	Description
Ver.	Version of the protocol.
Agg.	Identifies if the packet can be aggregated.
HL	Header length in bytes.
Total length	Payload length in bytes.
Time to Live	This valued is decremented at each hop.
Protocol	Upper layer protocol.
Header Checksum	Error-checking of the header.
Source Address	Address of the sender.
Destination Address	Address of the receiver.
Options	To be used in future features.

B. Packet formats

Here, we present the packet formats that enable communication in the SDWSN. The MAC and PHY layers are left as standard without modification.

The packet format for forwarding packets is shown in Fig. 1. The total length of the header is 10 bytes. The header fields are described in Table I.

As stated in [7], the essential components needed in the correct operation of an SDWSN are: *neighbor discovery*, *data collection*, *neighbor advertisement*, and *network configuration*.

1) Neighbor Discovery: Neighbor Discovery (ND) is used to discover neighbors, detect changes and discover the gateway to the controller. The rate of ND packets depends on the application [7]. Increasing the rate of ND packets allows us to rapidly detect changes in the network but at the expenses of an increased in the energy expenditure of sensor nodes.

All nodes in the network broadcast an ND packet containing: the rank of the node (2 bytes), an accumulated Received Signal Strength Indicator (RSSI) to the controller (2 bytes) and a checksum (2 bytes). This packet is placed in the payload of the forwarding packet.

2) Data collection: The sensed information is shared with the controller for further processing. These data packets are routed to the controller using the routing table at each sensor node, previously configured by the controller. The sensed data depends on the application. We assume that the size of the sensed data is fixed for all sensor nodes. This data comprises a length field (1 byte), source address (2 bytes), sequence (2 bytes), sensed measurand 1 (2 bytes) and sensed measurand 2 (2 bytes).

3) Control packets: There are two SDWSN functions carried in the control packet: Neighbor Advertisement (NA) and Network Configuration (NC). These two functions are encapsulated in the payload of the control packet format shown in Fig. 2. The header fields are described in Table II.

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	0	Type								Length								Sender Rank															
4	32	Sender Energy																Routing Checksum															
8	64	Checksum																Options															

Fig. 2: SDWSN control packet format.

TABLE II: Description of fields in the header of the control packet

Field	Description
Type	Type of control packet in the payload.
Length	Length of the payload.
Sender Rank	The rank of the sender.
Sender Energy	The remaining energy of the sender.
Routing Checksum	Checksum of the sender's routing table.
Checksum	Error-checking of the header.
Options	To be used in future features.

(i) Neighbor Advertisement (NA): Sensor nodes use NA messages to report their own and their neighbors' status to the controller. The controller makes use of NA messages to build information tables of sensor nodes and links between nodes. It can then compute the routing tables. The NA packet contains the neighbor address (2 bytes), RSSI of neighbor (2 bytes) and the neighbor's rank (2 bytes). Sensor nodes also put their remaining energy levels in the 'Sender Energy' field in the control packet header of Fig. 2.
Sensor nodes calculate their remaining energy level based on the initial energy and the current energy consumption. The current energy consumption is measured by using [20], which simply measures the time the sensor spends in each state. The sensor states are: processing (cpu), low power mode (lpm), transmitting (tx), listening (rx), idle transmitting ($idle\_tx$) and idle listening ($idle\_rx$). The energy consumption (E) is calculated as follows:

$$E = V * [t_{cpu} * i_{cpu} + t_{lpm} * i_{lpm} + t_{tx} * i_{tx} + t_{rx} * i_{rx} + t_{idle\_tx} * i_{idle\_tx} + t_{idle\_rx} * i_{idle\_rx}] \quad (1)$$

Where V is the supply voltage of the node, t is the time spent in a particular state and i is the current consumption for that particular state. The energy consumption, in the current sample, can be expressed as:

$$E = V \sum_{k \in S} t_k * i_k \quad (2)$$

Where S is the set of the sensor states. Therefore, the remaining energy (RE), at time $t+1$, can be expressed in terms of the previous RE_t as follows:

$$RE_{t+1} = RE_t - E_t \quad (3)$$

Where RE_0 is the initial energy of the sensor node and $E_0 = 0$. All nodes use the above-mentioned energy consumption model to estimate the remaining energy consumption except for the controller which is assumed to have access to mains power. Sensor node n computes the remaining energy level using Algorithm 1.

Algorithm 1: Remaining energy (RE) of node n

Result: RE consumed by node n.
Input : voltage (V), current consumed in each state (i_s)
Let be $S := \{cpu, lpm, tx, rx\}$.
Let t_s be the time in s, where $s \in S$.
Let E_p be the energy spent in sample p.
Let RE_p be the remaining energy in sample p.
Let RE_0 be the initial energy.
Let $E_0 = 0$
$sum = 0$
for s *in* S **do**
  $sum = sum + (i_s * t_s)$
end
$E_p = sum * V$ `// Total energy consumed.`
$RE_{p+1} = RE_p - E_p$ `// if` $p = 0$, $RE_{p+1} = RE_0$
$RE_p = RE_{p+1}$
return RE_{p+1}

TABLE III: SDWSN links table

Source	Destination	Cost	Lifetime [s]
1.0	6.0	120	100
2.0	7.0	89	110
8.0	15.0	56	50

(ii) Network Configuration (NC): The controller uses the NC packet to reprogram the sensor node's routing table. This packet contains the destination address (2 bytes) and the next-hop address (2 bytes) of all routes to be reconfigured.

C. Controller's tables

The controller maintains two main tables which are constructed based upon the reports sent by the sensor nodes:

1) links table: this table holds the connections between sensor nodes. An example is shown in Table III. Each row refers to a link between two nodes, and also specifies the cost of going from source to destination, as well as the remaining lifetime of the entry. The cost can be any metric such as ETX (Expected Transmission Count) [21] or RSSI. The remaining lifetime is used to remove dead links in the network.

2) Sensor nodes table: this table holds the status and properties of sensor nodes. Upon reception of an NA message, the controller can either add an entry or update fields regarding the sender. Fields of this table are shown in Table IV. The energy field stores the remaining energy for the given node, the Ranks and NB field identify key nodes in the network, and the Alive field is a flag used for computing the routing algorithm.

IV. Energy-aware routing protocol for SDWSNs

In this section, we propose an energy-aware SDWSN routing algorithm that seeks to prolong the network lifetime until

TABLE IV: SDWSN sensor nodes table

Addr.	RE [mJ]	Ranks				NB	Alive	LT [s]
		M	L	H	O			
1.0	18597	3	2	2	6	6	1	250
3.0	0	1	1	2	3	3	0	100
8.0	19568	4	1	2	3	3	1	300

M: Sensor node rank
L, H: No. of sensor nodes with lower and higher ranks, respectively
O: No. of sensor nodes with other ranks
NB: Number of neighbors
LT: Remaining lifetime for this entry

the first node dies due to energy depletion. The main objective of the routing algorithm is to balance the energy consumption across the WSN and reduce the number of packets flowing on it.

Firstly, the controller and sensor nodes start broadcasting ND packets. Sensor nodes that receive an ND packet with a lower rank than the current rank, update the path to the controller. After sensor nodes discover the path to the controller, they start sending NA packets. Then, the controller builds the topology of the entire network as described in Section III-C.

A. Energy balancing

The routing protocol balances the network energy by selecting for each node a path with the highest remaining energy level among multiple paths. The algorithm chooses this 'energy-shortest' path such that sensor nodes with less energy also use less energy in forwarding packets. Thereby, the energy level across the network is maintained relatively uniform, and the lifetime and reliability of the network is improved.

Firstly, the algorithm forms a tree topology containing information from the *links table*, *ranks* of the sensor nodes, and the *remaining energy* level for each sensor node, as described in Section III-C. An example of the tree topology is shown in Fig. 3. Secondly, the algorithm then deals with nodes of lower rank, namely, sensors with rank one (Sensor nodes 1, 2 and 3 in the figure). It chooses the energy-shortest path with the highest remaining energy level and updates the total remaining energy for the path. After finalizing all lower rank nodes, the algorithm then includes nodes one rank higher, and so on. The algorithm once more calculates the energy-shortest path for each sensor node with rank two and it updates the total remaining energy for the path for each sensor node, as shown in rank two sensor nodes of Fig. 3. For example, sensor node 4 has a remaining energy of 18622 mJ. This node could potentially reach the controller with minimum hops both through sensor nodes 1 and 2. However, for this particular period of time, the maximum remaining energy path is through sensor node 1 which has 18596 mJ remaining. This is larger than the 18320 mJ for sensor node 2. The algorithm then updates the total remaining energy for the path for node 2 to $18622+18596 = 37218$ [mJ]. The same procedure is repeated for all sensor nodes with the same rank. The algorithm then continues calculating the energy-shortest path for all nodes in increasing rank order. The Algorithm 2 shows the steps involved in balance the energy consumption of the WSNs in a particular period of time.

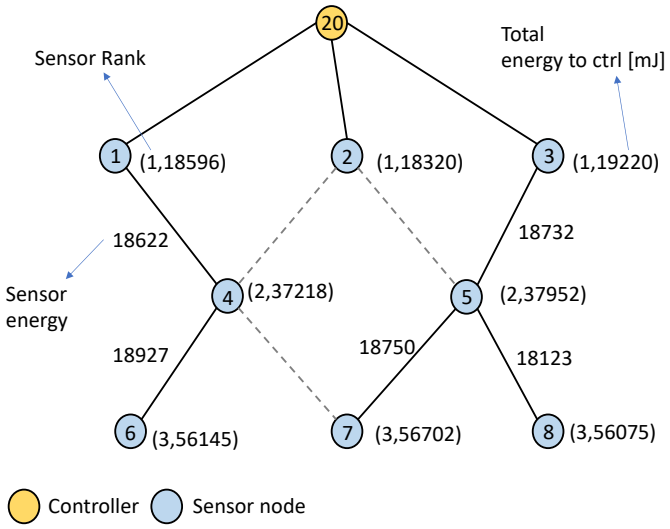


Fig. 3: Example of the tree topology.

Once the algorithm completes formation of the tree, the controller constructs a *configuration routing table*. This holds information about the target sensor node and a list of its routes to be configured. The controller builds this table using the Depth First Search (DFS) Algorithm [22]. Lastly, the controller places the list of routes in an NC packet then it delivers the control packet as proposed in [17].

After the first iteration, the algorithm then waits until the next reconfiguration time commences. It then computes the energy-aware algorithm once more and checks whether the network topology has changed or not. If it has, the controller reconfigures the network. This procedure will be repeated until the network lifetime of the sensor network is exhausted. The frequency of network reconfiguration is an open research question that is still being investigated. For example, frequent reconfiguration will cause high control overhead in the network. This, in turn, will increase energy consumption and impact the PDR. Infrequent reconfiguration will reduce the control overhead, but the network may not react appropriately to changes in network conditions. It would make sense, therefore, to use the measured energy levels to intelligently adapt the reconfiguration frequency, and this will be the focus of future work.

B. Reducing number of packets

As mentioned in [7], the number of packets flowing in the SDWSN directly affects the performance of the network. There are two main options where we can minimize the number of packets. One option is to aggregate control packets, which can be NA or NC. The other option is to aggregate the data packets sent to the controller. The first option is not desirable since we normally need control packets delivered quickly and reliably to maintain network functionality. The better option is then to aggregate data packets (collected sensor data).

To achieve this, a queue system is added in each sensor node that caches data packets from other nodes. The selection criteria to cache a data packet is based on the aggregate flag

Algorithm 2: Energy-aware routing algorithm.

```
Result: Tree topology
Input  : energy_cost C, source S, Vertices N
for i = 1 → N do
    sptSet[i] ← false      // true if vertex i
     is included
    pred[i] ← ∞
    energy[i] ← 0
end
sptSet[S] ← true
energy[S] ← ∞
for rank = 0 → maxRank do
    for node = 0 → N do
        if node = rank & node.energy > 0 then
            while pred[node] = −1 do
                u ← maxEnergy(N)
                sptSet[u] ← true
                for v = 1 → N do
                    if !sptSet[v] &
                     energy[u] + C[u][v] >= energy[v]
                     then
                        energy[v] = energy[u]+C[v][u]
                        pred[v] = u
                    end
                end
            end
        end
    end
end
```

in the SDWSN forwarding packet in Fig. 1. A hop can only aggregate a packet if the aggregate flag is set, and it is then placed in the queue. The aggregating node then waits until it is time to send a data packet to the controller. The node can aggregate a fixed number of packets defined by the application and must be less than the maximum IEEE 802.15.4 [23] packet size (127 bytes) and the remaining memory space. For ContikiMAC, the minimum packet size is 23 bytes to ensure the transmission is long enough so that it does not fall between to subsequent Clear Channel Assessments (CCAs) [24]. This includes preamble, start of frame delimiter, and length field, and leaves 16 bytes of packet data. Experimentally, the length of the ContikiMAC header (using our addressing scheme) is 9 bytes, the size of the SDWSN forwarding packet is 12 bytes and the header of the data aggregation packet is 1 byte, therefore, there are 22 bytes used for headers, leaving $127 - 22 = 105$ bytes for the payload. When aggregating packets we use the same packet format for data packets but we remove the length field as it is redundant, thus, the payload of the aggregation packet is 8 bytes. Therefore, the maximum number of data collection packets that a node can aggregate is $105/8 \approx 13$. The frequency of sending data packets will normally be set by the requirements of the application. The aggregating node places all aggregated data and its own data in a data collection packet, and forwards the packet to the next-hop to the destination. This packet is delivered with the aggregate flag set to zero to ensure subsequent nodes do not hold onto this packet and delay the delivery time even further. The controller processes the packet by looking at the number of data packets aggregated in the length field of the data packet header, and then it processes all data according to the sensor node address.

Aggregating data packets reduces both the overhead and the number of packets flowing across the network, which improves network performance in terms of energy consumption and PDR.

V. Control overhead reduction

In this section, we proposed a technique to reduce the control overhead in the SDWSN by reducing the communication between the controller and sensor nodes (NC packets).

As mentioned above, there are two types of control packets flowing in the network: neighbor advertisement (NA) and network configuration (NC) packets. It would be possible to reduce the number of neighbors included in an NA packet by programming sensor nodes to only advertise neighbors that have a significant change from their previously advertised messages. This could significantly reduce NA bytes flowing to the controller and, thus, the energy consumption of intermediate nodes could also be reduced. However, this approach would not allow the controller to capture link status or changes between sensor nodes. Thus, any dead links may go unnoticed, leading to misconfiguration of the network. Therefore, we opted to reduce the number of NC packets in a way that does not interfere with collection of network status data, but still increases the PDR while simultaneously reducing the overall energy consumption of the network.

The above control overhead reduction technique is programmed in both controller and sensor nodes. Sensor nodes, still report their own and their neighbors' information status to the controller using NA packets, but now they also set a routing checksum field in the header of the control packet. It contains a calculation of the checksum of all the current routes in the sensor node's routing table as shown in Algorithm 3. This calculation is simply the one's complement of the one's complement sum of all 16-bit words in the routes [25]. On the controller side, it receives the NA packet and it processes the attached neighbors' information. It then stores the received checksum in the configuration routing table. After the energy-aware routing protocol finishes processing, the network configuration function starts building the routes for each sensor node in the tree topology formed. The controller then calculates the routing checksum over the configuration routes for that specific sensor node and compares it with the received checksum of the sensor node. If both checksum values match, the controller skips that configuration packet and only sends packets with different checksum values. The sending process starts from the lowest ranked nodes to make sure that routes for packets going deeper in the network are already set up.

VI. Experiments and results

We now evaluate the performance of the proposed energy-aware routing algorithm for software-defined multihop wireless sensor networks, as well as the efficacy of the control

Algorithm 3: Calculates the routing table checksum

Result: Checksum
$sum = 0$
$B_n \subseteq N$ `// Neighbors of node n.`
foreach *route in routing table* **do**
  $dest \leftarrow route.dest$
  $via \leftarrow route.via$
  if $dest \notin B_n \vee dest = controller$ **then**
    $sum = checksum(sum, dest, via)$
end

overhead reduction technique. We compare it with the familiar shortest-path routing algorithm as a baseline. As already shown, prior software-defined routing approaches described in Section II fail to provide implementation details of their algorithms, and none have been tested in a network simulator environment. In contrast, we have implemented our proposed SDWSN architecture, energy-aware algorithm and the control overhead reduction technique in Contiki OS, which is a lightweight and open-source operating system for tiny networked embedded systems. The experiments were conducted in Cooja [8], which is a sensor network simulator for Contiki OS. As the experiment requires multiple runs and relatively long running time for comprehensive results, we opted to run Cooja on high-performance computing infrastructure [26].

A. Simulation setup

The sensor node distribution used is shown in Fig. 4. This topology is similar to the network topology used in [27] for a smart city simulation. The topology has 20 sensor nodes: one software-defined controller, which is sensor node 20, as well as 19 sensor nodes. We used a WiSMote [28] sensor node for the controller as it has a relatively large memory to host tables and the routing algorithm, while we used simpler Tmote Sky [29] for sensor nodes. The current consumption values were taken from [29]. The optimal timing for NC packets is still an open research question; however, a small NC period will tend to increase the control overhead and prevent sensor network stabilization, whereas a large NC period will hamper the prompt balancing of energy across the network. Therefore, we use a compromise period (≈ 5 times ND) as used in [7], [18]. The controller is embedded in the WiSMote node (sink) and the placement of the controller directly affects the WSN performance. It can be located in such way that minimizes the energy consumption of sensor nodes; however, this is not always the optimal position to extend the NL since the solution to this optimization problem can be located in a low density area, resulting in an inefficient resource management in the controller neighborhood [30]. Nodes in the vicinity of the controller deplete their energy first, resulting in a shorter NL. Since the objective of this paper is not to spot the optimal placement of the controller but to show the advantages of software-defined routing protocol, then we opted to place the controller in one corner of the WSN. The experimental parameters are summarized in Table V.

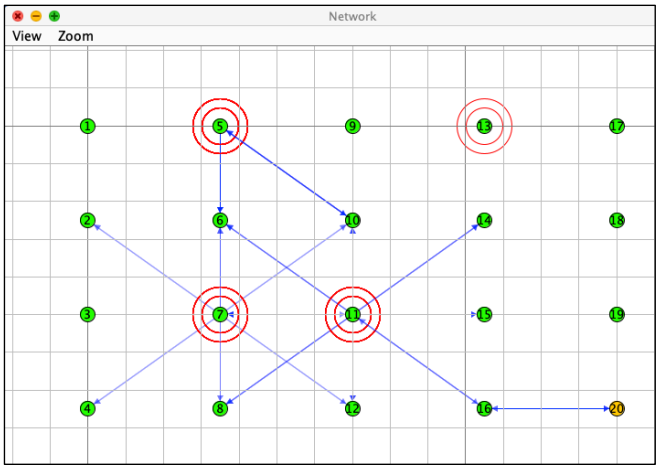


Fig. 4: Deployment of the SDWSN.

TABLE V: Experiment parameters

Parameter	Value
Radio Medium Model	Unit Disk Graph Medium Distance Loss
Simulation time	3 h
MAC protocol	ContikiMAC [24]
Node types	Sky & wismote
Data packet	3 mins
Max No. of aggr. packets	10
Neighbor Discovery (ND)	3 mins
Neighbor Advertisement (NA)	4 mins
Network Configuration (NC)	14 mins
Transmission range	50 m
Interference range	100 m
Supply voltage	3 V
Initial Energy (RE_0)	20 J
CPU active (i_{cpu})	1.8 mA
CPU low power mode (i_{lpm})	0.545 mA
TX current (i_{tx})	17.4 mA
RX current (i_{rx})	20 mA

B. Performance metrics

The performance and comparison of the proposed approaches presented in this paper are tested for four performance metrics.

1) Dead Nodes: the number of sensor nodes that have consumed all of their energy.

2) Control overhead: defined as the total accumulated number of NA and NC packets over simulation time.

3) Network lifetime (NL): defined as the time from the SDWSN starts functioning until the time the first node dies.

4) Packet Deliver Ratio (PDR): this metric is taken as the ratio of data packets received at the controller to the number of data packets sent.

C. Results discussion

In this section, we discuss the experiment results for dead nodes, control overhead, network lifetime and PDR. We ran multiple simulations with different seeds for each routing protocol to give more realistic results and plot the averages in Fig. 5 with a confidence interval of 95%.

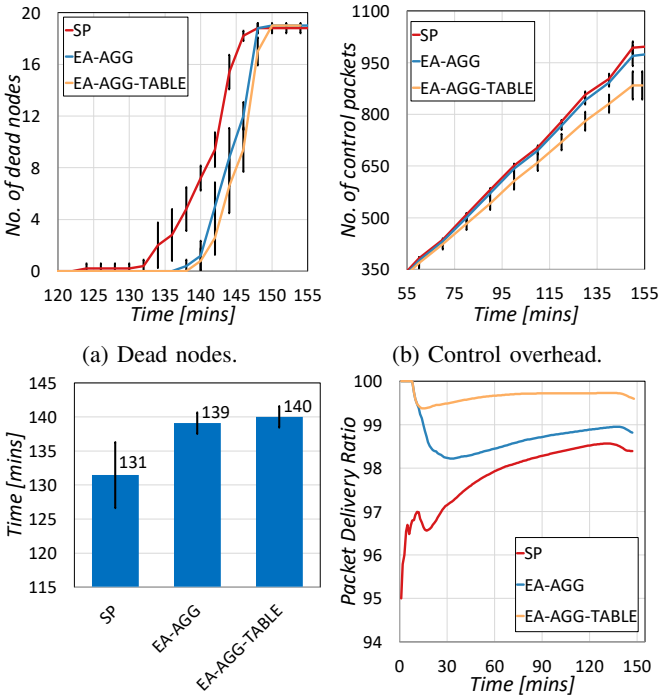
1) Dead nodes: The number of dead nodes over time is presented in Fig. 5a. For the shortest-path algorithm (SP), the

sensor nodes exhausted their energy faster in comparison with the energy-aware routing algorithm with data aggregation (EA-AGG) and the EA-AGG with routing table tracking (EA-AGG-TABLE). The SP algorithm uses the same path to forward packets, so common hops to the destination exhaust their energy first. For the EA-AGG algorithm, the improvement was mainly achieved via route changes from time to time that took paths with the most remaining energy. For the EA-AGG-TABLE algorithm, there is an extra improvement compared to EA-AGG, due to the reduction in the number of NC packets sent.

2) Control overhead: The number of control packets for the three routing protocols over time are shown in Fig. 5b. As we can see, the number of control packets for EA-AGG-TABLE was reduced by approximately 12% in comparison with the other two protocols. As the controller has knowledge of the routing table for each node, it will only send NC packets to those nodes with routing tables that do not match the routing table as stored by the controller. This reduces the number of NC packets flowing in the network.

3) Network lifetime: The time until the first node dies is shown in Fig. 5c. It is clear that the SP algorithm has the shortest lifetime, 6.5% less on average. The SP algorithm also exhibits much larger variability with network lifetimes lying in a range of 3-10% shorter than the other two protocols. The SP algorithm uses the same path to forward packets until there is a change in the topology due to a dead node. Therefore, packets coming from deep in the network to the controller through a common node will exhaust the common node's energy first. In the two new protocols, the controller continually tries to balance the energy by reconfiguring the sensor nodes' routing tables based on the energy-shortest path, giving longer node lifetimes. This also explains the narrower range in the results mentioned above.

4) Packet Delivery Ratio (PDR): The ratio of the number of data packets received at the controller divided by the number of packets sent to the controller is shown in Fig. 5d. All three protocols experienced a drop in the PDR in the first 15 minutes. This is due to sensor nodes deep in the network still finding a path to the controller, which means some packets get lost in the network. For the SP algorithm, the drop is more pronounced, because, while the network settles down, multiple NA and data packets are transmitted, increasing the collision probability whereas, for the other two protocols, the number of transmissions is less. Overall, all three protocols have high PDR mainly because the MAC protocol provides reliability to packets by retransmitting packets that are not acknowledged. However, the most PDR-efficient protocol is the EA-AGG-TABLE as it transmits fewer packets thanks to (i) data aggregation, and (ii) the checksum function, that reduces the number of data- and control-packets transmitted, lessening the probability of collisions in the network. Near the end of the simulation time, we can see that all protocols experienced a slight reduction in the PDR. The reason is that some common nodes to the controller die preventing packets from reaching their destination.



(a) Dead nodes. (b) Control overhead.

(c) First dead node. (d) PDR over time.

Fig. 5: Experimental results.

VII. Conclusions and Future Work

In this paper, we present a novel energy-aware (EA) routing algorithm and a control overhead reduction technique for enhancing the overall network lifetime of a SDWSN, while also maintaining a high PDR. We performed our experiments in Cooja that allowed us for the first time to realistically simulate key elements: the network conditions and physical factors influencing the WSN performance, the underlying operating system and the machine code instruction set that enable us to program them as well as capture physical events happening in the sensor nodes' hardware. Overall, the combination of this new EA algorithm, along with the data aggregation and a control overhead technique, outperforms the shortest-path algorithm in terms of dead nodes, control overhead, network lifetime and packet delivery ratio. We have demonstrated that the proposed approach could prolong the network lifetime of the WSN by approximately 6.5% compared to the shortest-path algorithm and that the control overhead was reduced by approximately 12% while maintaining a high PDR.

This paper finds that the software-defined approach in WSNs (SDWSN) is a promising approach to prolonging the network lifetime of a multihop WSN. However, the proposed routing protocol is only a starting point to explore innovative routing protocols. Future work could include: the study of reducing neighbor advertisement (NA) packets to further reduce the control overhead of the network; alternative routing protocols that further prolong the lifetime of the network; intelligently adapting the timing for the frequency of network configuration (NC) packets; and include performance metrics such as channel status to select intermediate nodes, amongst others.

REFERENCES

[1] C. U. Smith and C. M. Lladó, "Spe for the internet of things and other real-time embedded systems," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, Conference Proceedings, pp. 227–232.

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

[3] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Fragmentation-based distributed control system for software-defined wireless sensor networks," *IEEE transactions on industrial informatics*, vol. 15, no. 2, pp. 901–910, 2018.

[4] ——, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[5] L. Militano, M. Erdelj, A. Molinaro, N. Mitton, and A. Iera, "Recharging versus replacing sensor nodes using mobile robots for network maintenance," *Telecommunication Systems*, vol. 63, no. 4, pp. 625–642, 2016.

[6] W. Xiang, N. Wang, and Y. Zhou, "An energy-efficient routing algorithm for software-defined wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7393–7400, 2016.

[7] F. F. Jurado-Lasso, K. Clarke, and A. Nirmalathas, "Performance analysis of software-defined multihop wireless sensor networks," *IEEE Systems Journal*, 2019.

[8] F. Osterlind, "A sensor network simulator for the contiki os," Swedish Institute of Computer Science, Report T2006-05, 2006. [Online]. Available: http://eprints.sics.se/2296/1/SICS-T–2006-05–SE.pdf

[9] 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.

[10] S. Din, A. Paul, A. Ahmad, and J. H. Kim, "Energy efficient topology management scheme based on clustering technique for software defined wireless sensor network," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 348–356, 2019.

[11] L. Wenxing, W. Muqing, and W. Yuewei, "Energy-efficient algorithm based on multi-dimensional energy space for software-defined wireless sensor networks," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, Conference Proceedings, pp. 309–314.

[12] H. Bo, W. Muqing, Z. Min, and L. Wenxing, "An energy aware routing algorithm for software defined wireless sensor networks," in *2017 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, Conference Proceedings, pp. 1–6.

[13] 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.

[14] 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.

[15] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, Conference Proceedings, pp. 455–462.

[16] 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.

[17] F. F. Jurado-Lasso, K. Clarke, and A. Nirmalathas, "A software-defined management system for ip-enabled wsns," *IEEE Systems Journal*, 2019.

[18] 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.

[19] C. Buratti, A. Stajkic, G. Gardasevic, S. Milardo, M. D. Abrignani, S. Mijovic, G. Morabito, and R. Verdone, "Testing protocols for the Internet of Things on the EuWIn platform," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 124–133, 2016.

[20] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level power profiling for low-power wireless networks," Swedish Institute of Computer Science, Kista, Sweden, Technical Report T2011:05 1100-3154, 2011. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:1042895/FULLTEXT01.pdf

[21] D. S. J. De Couto, "High-throughput routing for multi-hop wireless networks," Thesis, 2004.

[22] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[23] J. A. Gutierrez, E. H. Callaway, and R. L. Barrett, *Low-rate wireless personal area networks: enabling wireless sensors with IEEE 802.15. 4*. IEEE Standards Association, 2004.

[24] A. Dunkels, "The contikimac radio duty cycling protocol," 2011.

[25] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error-control coding," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2531–2560, 1998.

[26] L. Lafayette, G. Sauter, L. Vu, and B. Meade, "Spartan performance and flexibility: An hpc-cloud chimera," *OpenStack Summit, Barcelona*, 2016.

[27] R. J. Tom, S. Sankaranarayanan, V. H. C. de Albuquerque, and J. J. Rodrigues, "Aggregator based rpl for an iot-fog based power distribution system with 6lowpan," *China Communications*, vol. 17, no. 1, pp. 104–117, 2020.

[28] WiSMote. [Online]. Available: http://www.aragosystems.com/produits/wisnet/wismote/

[29] M. Corporaton, "Tmote sky: Datasheet," 2006.

[30] 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.

F. Fernando Jurado-Lasso (GS'18) received the B.Eng. degree in Electronic engineering in 2012 from the Universidad del Valle, Cali, Colombia, the M.Eng. degree in telecommunications engineering and the Ph.D. degree in Engineering both from The University of Melbourne, Melbourne, VIC, Australia, in 2015 and 2020, respectively.

His research interests include networked embedded systems, software-defined wireless sensor networks, protocols and applications for the Internet of Things.

Ken Clarke received the B.Sc. (Hons.) degree in applied physics from Heriot–Watt University, Edinburgh, U.K., in 1985.

He has worked in the optoelectronic and telecommunications industries in various RD and engineering roles in both the U.K. and Australia from 1985–2010, before moving to the University of Melbourne where he is currently a Deputy Director with the Networked Society Institute. He has authored more than 50 articles and book chapters, and produced five patents.

Andres Navarro Cadavid is an Electronic Engineer (1993), with a Master on Technology Management (1999), both from Universidad Pontificia Bolivariana in Medellín. He obtained his PhD in Telecommunications from Universitat Politécnica de Valencia in 2003. As ITU expert, he advised different governments in Digital TV and Spectrum Management. He is a participant on COST Action CA15104 IRACON and former actions COST IC1004 and COST 2100. He is also a member of IEICE and EurAAP. He is currently the coordinator of the PhD program on engineering and director of the i2t research group at Universidad Icesi since 1999. His research interests are Spectrum Management, mobile radio planning, radio propagation and m-health.

Ampalavanapillai Nirmalathas (M'98–SM'03) received the B.Eng. and Ph.D. degrees in electrical engineering from The University of Melbourne, Melbourne, VIC, 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.

Author/s:
Fernando Jurado-Lasso, F; Clarke, K; Cadavid, AN; Nirmalathas, A

Title:
Energy-aware routing for software-defined multihop wireless sensor networks

Date:
2021-02-16

Citation:
Fernando Jurado-Lasso, F., Clarke, K., Cadavid, A. N. & Nirmalathas, A. (2021). Energy-aware routing for software-defined multihop wireless sensor networks. IEEE Sensors Journal, 21 (8), pp.1-1. https://doi.org/10.1109/jsen.2021.3059789.

Persistent Link:
http://hdl.handle.net/11343/265797

File Description:
Accepted version