Article

Centralized Energy Prediction in Wireless Sensor Networks Leveraged by Software-Defined Networking

Gustavo A. Nunez Segura * and **Cintia Borges Margi**

Laboratório de Arquitetura e Redes de Computadores, Escola Politécnica, Universidade de São Paulo, São Paulo 05508-010, Brazil; cintia@usp.br
* Correspondence: gustavoalonso.nunez@usp.br

Abstract: Resource Constraints in Wireless Sensor Networks are a key factor in protocols and application design. Furthermore, energy consumption plays an important role in protocols decisions, such as routing metrics. In Software-Defined Networking (SDN)-based networks, the controller is in charge of all control and routing decisions. Using energy as a metric requires such information from the nodes, which would increase packets traffic, impacting the network performance. Previous works have used energy prediction techniques to reduce the number of packets exchanged in traditional distributed routing protocols. We applied this technique in Software-Defined Wireless Sensor Networks (SDWSN). For this, we implemented an energy prediction algorithm for SDWSN using Markov chain. We evaluated its performance executing the prediction on every node and on the SDN controller. Then, we compared their results with the case without prediction. Our results showed that by running the Markov chain on the controller we obtain better prediction and network performance than when running the predictions on every node. Furthermore, we reduced the energy consumption for topologies up to 49 nodes for the case without prediction.

Keywords: software-defined networking; energy consumption; centralized prediction

Citation: Nunez Segura, G.A.; Borges Margi, C. Centralized Energy Prediction in Wireless Sensor Networks Leveraged by Software-Defined Networking. *Energies* **2021**, *14*, 5379. https://doi.org/10.3390/en14175379

Academic Editor: Moez Esseghir

Received: 19 July 2021
Accepted: 6 August 2021
Published: 30 August 2021

1. Introduction

Wireless sensor networks (WSN) are formed by interconnected nodes with the capacity of sensing, processing, and communicating. Advances in digital electronics, wireless communication, and microelectromechanical systems (MEMS) have enhanced WSN popularity, a technology currently used in a vast number of applications, such as environmental, industrial and health monitoring, tracking and detection, home automation, and security [1]. WSN is also a key technology in the development of the Internet of Things (IoT) [2]. These networks are resource constrained, typically having low processing power, low storage capacity, limited bandwidth, and limited energy. Those restrictions are important to reduce the cost of hardware production but limit the network flexibility, resource reuse, security, and network lifetime.

Kobo et al. [3] state that to solve WSN-inherent problems, including energy and computational constraints, a holistic solution is necessary. Previous works propose integrating WSN with Software-Defined Networking (SDN) as a potential solution to the previously mentioned limitations. The SDN approach decouples the control plane from the data plane, centralizing the routing decisions in the SDN controller [4,5]. The integration of WSN and SDN is referred to in the literature as Software-Defined Wireless Sensor Networks (SDWSN) [6–9], and results show that SDWSNs perform as well as RPL [9] (IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550).

Routing in WSN considers the resource constraints and the performance demands from applications [10,11]. For example, an energy-aware routing protocol uses energy consumption metrics to extend the network lifetime [12]. In SDN-based networks, the controller is in charge of all routing decisions, and for this reason, it needs information about the network topology and status (i.e., performances metrics), such as energy consumption metrics. To understand the importance of this information in the SDN controller, Oliveira and Margi [12] applied energy-aware route selection to SDWSN. In their proposal, each node reports its remaining energy to the controller every time there is a decrease higher than 1% of the total energy supply. The energy-aware route selection performance was compared to a link quality route selection using the expected transmission count (ETX), a routing metric based on active probing measurements [13]. Their results showed that using the energy-aware route selection increases the network lifetime and reduces the number of route changes. However, this requires that the network devices periodically inform their energy information (i.e., average energy consumption or residual energy) to the SDN controller to maintain the routing metric updated, which increases packets traffic and impacts the network performance.

Several authors proposed mechanisms to reduce the transmissions of these packets. Their main objective is to predict future values of a monitored variable to reduce the frequency of transmissions to update the metric value [14–17]. For example, Dias et al. [16] implemented autoregressive algorithms in an IoT context to predict sensing measurements, with results depicting a reduction in data transmission and energy consumption. Mini et al. [17] compared Markov chain and time series based node energy consumption prediction algorithms in WSN, and obtained a reduction in the number of messages with energy consumption information for both cases. On the other hand, the prediction requires processing and energy resources that are scarce in WSN nodes.

In this work, we study the benefits and consequences of moving the prediction execution from every WSN node to one centralized node without resource constraints. For this, we decided to implement our proposal using an SDN architecture. Thus, we can use the SDN controller processing power to predict the energy consumption of every node in the network and its communication structure to share the information between the WSN nodes and the controller. Our hypothesis is that by centralizing the prediction we can improve the prediction accuracy and network performance if compared with the case that every node runs the prediction. This is important for two main reasons: (i) improving network performance means improving the delivery rate and decreasing delay of energy information required for control decisions without impacting on other packets traffic, and (ii) in case of missing packets, the controller will be able to predict the missing information. The second point is not possible to solve in the cases without prediction or executing the prediction on every node unless the node retransmits the packet, which means more packets traffic.

Previous works tested Markov chain [18,19] for energy consumption prediction in SDWSNs but, to the best of our knowledge, only our previous work [20] implemented the Markov chain running in the controller to predict the network energy consumption of each node in the network. Here, we extend our previous research by implementing and comparing two strategies: the prediction algorithm runs on all the sensor nodes and the prediction algorithm runs only on the controller node. The metrics we used to compare the performance were prediction accuracy, energy consumption, packets delay, delivery rate, and control packets overhead. Despite being widely used to support prediction in networks, machine learning strategies do not apply to our problem as one could not run the ML prediction algorithm on resource-constrained devices given a large amount of data necessary [21].

Our key findings are that by moving the prediction processing overhead to the SDN controller, we obtain a better prediction accuracy, improved network performance, and reduced energy consumption. However, this approach increases the traffic of packets related to energy consumption information.

The remainder of this paper is organized as follows. Section 2 reviews related works about energy consumption prediction using Markov chain in WSN. Section 3 explains the Markov chain based energy consumption prediction. Section 4 details our energy prediction proposal. Section 5 shows and discusses the experiments and results, and Section 6 concludes the paper.

2. Related Work

In this section, we analyze the literature about energy consumption prediction in WSN using Markov chain models. We emphasize the analysis of works studying energy consumption prediction on SDN-based networks.

Main results presented by the works implementing Markov chain for energy consumption prediction in WSN have shown a reduction of the traffic of packets containing energy information and the energy consumption itself [17,22,23]. Moreover, the energy models used in these works did not include important elements that affect the accuracy, e.g., the sensing module or the processing module energy consumption [24]. More accurate models provide more accurate and precise network lifetime estimation but require more memory and processing resources. Thus, the implementation of accurate energy models in resource-constrained devices is a current challenge.

Recent works propose different strategies to improve energy efficiency and network performance based on the prediction of energy-wise information. In [25], authors propose to predict link quality metric based on an energy density factor using a fuzzy interference system. Then, in [26] authors propose a black widow optimization based algorithm to predict sinks location in mobile WSNs. Their results showed an improvement in energy efficiency; moreover, these works did not address the problem of centralization of energy information and its impact on the network performance.

In traditional WSN networks, it is common that individual nodes run the energy consumption prediction algorithm and inform the result to a central node. The SDN paradigm has open the opportunity to migrate processes from individual nodes to the SDN controller, such as communication channel assignment [27], traffic analysis [28], intrusion detection [29,30], or energy consumption prediction.

In a previous work [20], we implemented a centralized energy consumption prediction for SDWSNs. Similar approaches were implemented by Rahimifar et al. [18] and by Miranda et al. [19]. In [18,20], the SDN controller receives energy consumption information from the nodes in the network; the main difference is that in [18] the nodes do the prediction and send an energy consumption rate to the controller, while in [20] the nodes send to the controller the information required to do the prediction using Markov chain. Unlike the works in [18,20], in [19] the WSN nodes communicate the energy information to a specific sink instead of to the controller. Moreover, they propose a centralized detection as well.

From the results in [18–20], we know that the energy prediction strategy can reduce the average energy consumption of the nodes and delay of the packets containing energy information. However, these works do not investigate the impact of their proposals in performance metrics such as control and data packets overhead, delivery rate, and delay. Moreover, the energy models used are not "complete". According to the works in [31,32], processing and sensing processes represent around 15% of the total consumption. Furthermore, the authors of [24] showed that the consumption by states transitions could have a significant part in the final result. Thus, accurate energy consumption models should consider processing, sensing, communication and transitions energy consumption.

From previous works, we noticed that (i) energy prediction proposals for WSN using Markov chain are based on energy models that omit the processing and/or sensing energy consumption, (ii) the energy consumption prediction proposals have been tested through simulations but there is a lack of experiments in real devices, and (iii) the centralized prediction works based on Markov chain have shown promising results, but there are not enough results to measure the impact of its implementation in the network performance.

In this work, we addressed these gaps in the literature by implementing a centralized Markov chain-based energy consumption prediction using an energy model that considers processing, sensing, communication, and transitions consumption. Our objective is to propose an energy consumption prediction algorithm leveraged by SDN that is accurate and lightweight in terms of processing and communication resources. We evaluated the performances of our proposal through simulations, emulating sky motes, and experiments on real devices using TelosB motes. To evaluate the impact on the network performance, we measured the delivery rate, delay, packets overhead, and energy consumption.

3. Energy Consumption Prediction

This section has three main parts: first, we explain the basics of Markov chain and how we apply it to predict energy consumption in WSN; second, we propose an energy consumption model for WSN; and third, we explain the integration of the energy consumption model and the Markov chain for energy consumption prediction.

3.1. Markov Chain

A Markov chain is a stochastic process commonly used to model dynamic systems, such as computer behavior [33]. A Markov chain is composed of a set of states and the transition probabilities from one state to another, where the probability of the next state depends only on the current state (Markov property).

The Markov chain is represented by a probability matrix or by a transitions diagram, among other representations. In the case of using a matrix, the entries are the probabilities of a transition from one state to another. For example, for a probability matrix P, the element P_{12} is the probability of a transition from the state "1" to the state "2". Assuming X_0 is the initial state-space distribution and X_1 is the state-space distribution in the next time-step, $X_1 = X_0P$. Then, $X_2 = X_1P = X_0PP$, hence $X_n = X_0P^n$, where P^n is the probability matrix to the power of n.

Then, assume E_R represents the state's energy consumption prediction. To calculate E_R, we need to predict the number of visits to each state for the time we want to predict the energy consumption. The number of visits to state j starting from state i for n steps is $N_j = \sum_{t=1}^{n} P_{ij}^t$, where P_{ij}^t is the probability of going from state i to state j to the power of t. Moreover, the number of visits for all states is calculated as shown in Equation (1), where P^t is the probability matrix to the power of t.

$$N = \sum_{t=1}^{n} P^t \tag{1}$$

Then, the energy consumption by all the states E_R is calculated as shown in Equation (2), where E_s^T is a vector that contains the average energy consumption of each state transpose.

$$E_R = X_0 N E_s^T \tag{2}$$

Substituting E_S^T in Equation (2) by a matrix K_s that represents the cost of one transition, we predict the energy consumption for all transitions during n steps. We obtain K_s calculating the element-wise product ($\circ$) of probability matrix P and transitions cost matrix B, as shown in Equation (3). Then, E_B in Equation (4) is the energy consumption only by transitions after n steps, where e is a vector with all the entries 1.

$$K_s = P \circ B \tag{3}$$

$$E_B = X_0 N \left(K_s e^T \right) \tag{4}$$

Therefore, the total energy consumption predicted for the next n steps is

$$E_{total} = E_R + E_B \tag{5}$$

3.2. Energy Consumption Model

The energy consumption prediction using Markov chain requires that the energy consumption of each state in the set is known. Furthermore, we need a mechanism to evaluate the prediction accuracy of our proposal. To fulfill both requirements, we propose an energy consumption model that considers processing, sensing, communication, and transitions consumption.

3.2.1. Sensing Module

The sensing modules operate mostly using two states: active and sleep mode. In the active state, the module takes the measurement and transmits it to the MCU. In the sleep mode, it turns off most of its submodules to reduce the energy consumption and remains waiting for a signal to acquire the next measurement. Considering both states and the transitions between them, the sensing module energy consumption (represented as E_{Sen}) is calculated using Equation (6). The term $E_{sen-act}$ represents the active state, $E_{sen-sleep}$ the sleep mode, E_{sen-on} the turning on transition, and $E_{sen-off}$ the turning off transition.

$$E_{Sen} = E_{sen-act} + E_{sen-sleep} + E_{sen-on} + E_{sen-off} \quad (6)$$

To calculate the energy consumption for a certain period, the terms $E_{sen-act}$ and $E_{sen-sleep}$ are expressed in terms of power consumption and time. Then, we calculate the transitions energy consumption by multiplying the average transition energy consumption by the total transitions counted in this period of time. Thus, E_{Sen} is expressed as shown in Equation (7), where $P_{sen-act}$ and $P_{sen-sleep}$ are the active and sleep mode states average power consumption; $T_{sen-act}$ and $T_{sen-sleep}$ are the time spent on the active and sleep mode states; e_{sen-on} and $e_{sen-off}$ are the average energy consumption of one turning on and turning off transitions; and N_{on} and N_{off} are the number of transitions.

$$E_{Sen} = P_{sen-act}T_{sen-act} + P_{sen-sleep}T_{sen-sleep} + N_{on}e_{sen-on} + N_{off}e_{sen-off} \quad (7)$$

3.2.2. Processing Domain

The processing module's energy consumption model is based on the work in [31], which considers at least three states: active, idle, and sleep. For our proposal, we change the sleep state for power-saving states. This means more than one state is meant to reduce energy consumption. Thus, the processing energy consumption E_{CPU} for a certain period is calculated as shown in Equation (8).

$$E_{CPU} = P_{cpu-act}T_{cpu-act} + P_{cpu-id}T_{cpu-id} + \sum_{s=1}^{S} P_{\{pp,s\}}T_{\{pp,s\}} + \sum_{g=1}^{G} N_g P_{\{cput,g\}} \quad (8)$$

The terms $P_{cpu-act}$, P_{cpu-id}, and $P_{\{pp,s\}}$ are the *active*, *idle*, and *power-saving* states average power consumption, respectively. The terms $T_{cpu-act}$, T_{cpu-id}, and $T_{\{pp,s\}}$ are the time spent in the *active*, *idle*, and *power-saving* states, respectively. The terms N_g and $P_{\{cput,g\}}$ are the number of g transitions and their average power consumption. The transitions energy consumption is calculated by multiplying the number of transitions and their average energy consumption.

3.2.3. Communication Domain

Two approaches are used to calculate the energy consumption in the communication module. One approach is based on the number of bits transmitted and received [32,34]. The other approach is based on operation states, measuring the time spent on each state to estimate the total energy consumption [31,35], as explained for the processing and sensing modules. We chose to use the operation states approach to give homogeneity to the model.

Figure 1 depicts a simplified block diagram for a WSN communication module, based on four IEEE 802.15.4-compliant radios modules (CC2420 [36], CC1120 [37], MRF24J40 [38], and CC2520 [39]). Then, based on the simplified block diagram, four states were defined: transmitting, receiving, idle, and sleeping. When the module is in the transmitting state, the transmitter circuitry and the control logic block are active. In the receiving state, the receiver circuitry and control logic block are active. When the receiver and transmitter circuitry are turned off and the control logic block is turned on, the module is in the idle state. Last, when the receiver and transmitter circuitry are turned off and the control logic block is in the sleeping mode, the module is in the sleeping state.

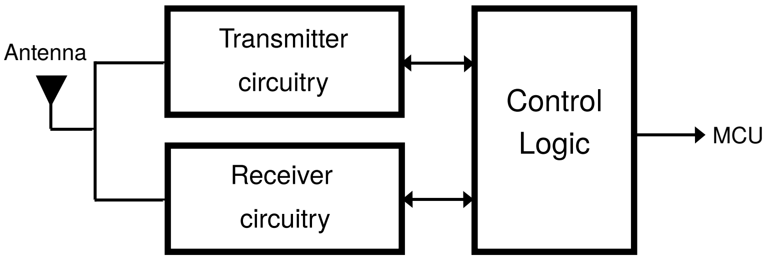


Figure 1. Communication module: simplified diagram.

Equation (9) express the energy consumption for a certain period of time. The terms P_{Tx}, P_{Rx}, P_{com-id}, and P_{com-sl} are the *transmitting*, *receiving*, *idle*, and *sleeping* states average power consumption. The terms T_{Tx}, T_{Rx}, T_{com-id}, and T_{com-sl} are the time spent on *transmitting*, *receiving*, *idle*, and *sleeping* states. The terms N_q and $P_{\{ct,q\}}$ are the number of q transitions and its average power consumption, respectively.

$$E_{Com} = P_{Tx}T_{Tx} + P_{Rx}T_{Rx} + P_{com-id}T_{com-id} + P_{com-sl}T_{com-sl} + \sum_{q=1}^{Q} N_q P_{\{ct,q\}} \tag{9}$$

3.3. Energy Consumption Prediction Implementation

As our proposal is for WSN, we decided to implement our energy consumption model on TelosB motes, an IEEE 802.15.4-compliant platform with an MSP430 microprocessor and 48 kB of flash memory. We modeled the TelosB mote energy consumption considering the device would run a temperature monitoring application, considering that the whole WSN was programmed using Contiki-3.0 [40]. Next, we describe the states of each module.

The temperature sensor of the TelosB is a Sensirion STH11 and it operates in two states: active and sleep. Regarding the processing energy consumption, Contiki avoids staying idle and it supports only two low-power modes: LPM1 and LPM3. Using the default configuration, the microprocessor never enters in LPM3 state. Thus, the processing module operates in two states: active and LPM1. The communication module behavior is defined by the CC2420 Contiki implementation. Contiki uses a radio duty cycle mechanism named ContikiMAC [41]. This mechanism uses a power-aware wake-up mechanism with a set of timing constraints aiming to keep the transceiver turned off as long as possible. An important issue is that using ContikiMAC, the communication module does not enter in the *idle* state. Furthermore, whenever ContikiMAC turns off the radio, the frequency synthesizer is disabled, which means the module is in the power down state. Based on these characteristics, the communication module operates in three states: transmitting, receiving, and power down.

Based on the TelosB energy consumption characteristics, we defined five states for the Markov chain: Sensing, Processing, LPM1, Transmitting, and Receiving. Each state is composed of three substates, one for each module. Table 1 describes the configuration of each Markov chain state.

Table 1. Set of states of the Markov chain.

State	**Domain**		
	Sensing	**Processing**	**Communication**
Sensing	active	active	sleeping
Processing	sleeping	active	sleeping
LPM1	sleeping	LPM1	sleeping
Transmitting	sleeping	active	transmitting
Receiving	sleeping	LPM1	receiving

The probability matrix is constructed after a training period where each node monitors its transitions. Each node constructs a transitions matrix to organize the number of transitions between states and a transitions vector to store the total of transitions of each state. Then, dividing each row of the transitions matrix by the total number of transitions, as shown in Equation (10), we obtained the probability matrix. Then, we can calculate Equations (2) and (4) to obtain the energy consumption prediction. Note that P depends on the node's behavior, which means that it can change every time it is calculated, according to the transitions (a and v) counted.

$$P = \begin{pmatrix} a_{11}/v_1 & a_{12}/v_1 & a_{13}/v_1 & a_{14}/v_1 & a_{15}/v_1 \\ a_{21}/v_2 & a_{22}/v_2 & a_{23}/v_2 & a_{24}/v_2 & a_{25}/v_2 \\ a_{31}/v_3 & a_{32}/v_3 & a_{33}/v_3 & a_{34}/v_3 & a_{35}/v_3 \\ a_{41}/v_4 & a_{42}/v_4 & a_{43}/v_4 & a_{44}/v_4 & a_{45}/v_4 \\ a_{51}/v_5 & a_{52}/v_5 & a_{53}/v_5 & a_{54}/v_5 & a_{55}/v_5 \end{pmatrix} \tag{10}$$

4. Centralized Energy Consumption Prediction

According to the literature review, previous works implementing centralized energy consumption prediction in WSN maintain the prediction processing overhead on the sensor nodes. The WSN nodes monitor their behavior, execute the prediction algorithm, estimate an energy consumption rate, and send this information to a sink or controller. We propose to move most of the processing required for the prediction to the SDN controller, assuming that the SDN controller does not have energy, processing power, or memory space constraints. In this manner, the WSN nodes monitor their behavior and the controller executes the prediction algorithm.

In detail, the WSN nodes execute a training period to calculate their probability matrix. Each node has to register all the transitions and order them in a matrix, just as explained in Section 3.3. When the training process ends, the node calculates the probability matrix and the energy spent up to this moment; then, it sends this information to the SDN controller. The SDN controller creates an energy profile for each node. This profile contains the probability matrix and an energy consumption time series, constructed from the information sent by the WSN nodes and the prediction results.

The SDN controller executes the prediction algorithm (Algorithm 1) to calculate the energy consumption for the next n steps. Then, the SDN controller informs the prediction value obtained to the respective WSN node. In this manner, after n steps the node can calculate the prediction error by comparing the prediction and the energy consumption obtained through the energy model. If the error is above a threshold previously defined, the WSN node initiates a new training period to update the energy information in the energy profile in the SDN controller. On the other hand, if the error is below the threshold, the SDN controller continues the time series construction using the value predicted.

Algorithm 1 Markov chain running on the controller

```
1: $n$: number of steps
2: $prediction$: energy consumption prediction for $n$ steps
3: $error$: prediction error
4: $threshold$: maximum error accepted
5: $p\_matrix$: transitions matrix
6: $s\_consumption(n)$: energy consumption prediction during $n$ steps for states
7: $t\_consumption(n)$: energy consumption prediction during $n$ steps for transitions
8: ——————————————————————————-
9: **Running on WSN nodes**
10: launch the training
11: wait the training period ends
12: calculate $p\_matrix$
13: send $p\_matrix$ to the controller
14: **while** true **do**
15:     wait $n$ steps
16:     calculate energy consumption
17:     calculate error(energy consumption, last prediction)
18:     **if** $error > threshold$ **then**
19:         obtain a new $p\_matrix$
20:         send new $p\_matrix$ to controller
21:     **end if**
22: **end while**
23: ——————————————————————————-
24: **Running on the SDN controller:**
25: wait for WSN information
26: **function** PREDICTION($p\_matrix, n$)
27:     calculate $s\_consumption(n)$ and $t\_consumption(n)$
28:     prediction=$s\_consumption + t\_consumption$
29:     update time series
30:     send $prediction$ to sensor nodes
31: **end function**
```

Implementation

The implementation was conducted using Contiki-3.0 [40], a widely used open-source operating system for WSN, and IT-SDN version 0.4 [9,42] (Available at http://www.larc.usp.br/users/cbmargi/www/it-sdn/, last time accessed 12 August 2021). IT-SDN is an open-source SDWSN framework implemented in our group, with results showing that its performance is similar to RPL for topologies up to 289 nodes.

Previous works used time steps of 1000 ms [17] and 100 ms [43]. Moreover, we decided to use a time step of 1 ms to obtain a more accurate probability matrix. For example, ContikiMAC [41] channel checking takes 0.884 ms plus the radio turning on and turning off time. Thus, time steps of 1000 ms and 100 ms lose information in our evaluation environment.

The communication between sensor nodes and the controller was implemented using the IT-SDN communication stack. IT-SDN [9,42] is an SDWSN framework composed of three communication protocols: the Southbound protocol, the Neighbor Discovery protocol, and the Controller Discovery protocol. The Southbound protocol defines the message formats for communication between the WSN and the controller. The WSN nodes use the Controller Discovery protocol to find a route to the controller and the Neighbor Discovery protocol to collect neighborhood information. The controller uses the neighbor information to define the routes according to the policies programmed. Then these routes are configured in the WSN nodes.

The Southbound protocol has six packet types: flow request, flow setup, flow ID register, acknowledgment, neighbor report, and data packet. Moreover, we created two more packets to address energy and prediction information communication: one packet exclusively to send energy information (i.e., from sensor nodes to the controller), and another packet exclusively to send the controller's reply (i.e., from the controller to sensor nodes). The energy information packets are routed using a flow identifier and the reply packets are source-routed from the controller.

All the energy information packets contain a byte specific for prediction mechanism identification. Thus, in the case of having more than one prediction algorithm, the receiver will know how to process the packet. For example, running the prediction on the WSN nodes and running the prediction on the SDN controller. Each case requires different information to be shared, thus using this identifier we inform which one we are implementing. Furthermore, using this identifier we are able to mix both approaches. Table 2 shows packets' sizes in bytes for both cases.

Table 2. Energy information packets' sizes.

Energy Information	
Approach	Packet size (B)
Running on the controller	89
Running on the WSN node	15
Controller reply	
Running on the controller	12 + 2 × hops

5. Experiments and Results

In this section, we describe the experiments we used to measure the performance of our energy consumption model and our centralized energy consumption prediction proposal. Then, we analyze and discuss the results obtained.

5.1. Energy Consumption Model Accuracy

The energy consumption model accuracy was evaluated using an N6705C DC power analyzer from Keysight. First, we measured the states and transitions average energy consumption using the power analyzer. Then, we deployed a network as shown in Figure 2 to obtain the energy consumption estimation from one node in an SDWSN and measure its energy consumption with the N6705C at the same time. The controller software runs on a desktop computer Intel Core i5 CPU 750 and 8 GB of memory RAM. All the nodes in the WSN are TelosB motes, including the gateway used to communicate the WSN and the SDN controller. Figure 3a depicts the connection to collect energy consumption measurements using the N6705C equipment and Figure 3b depicts the connection between the SDN controller and the WSN gateway.

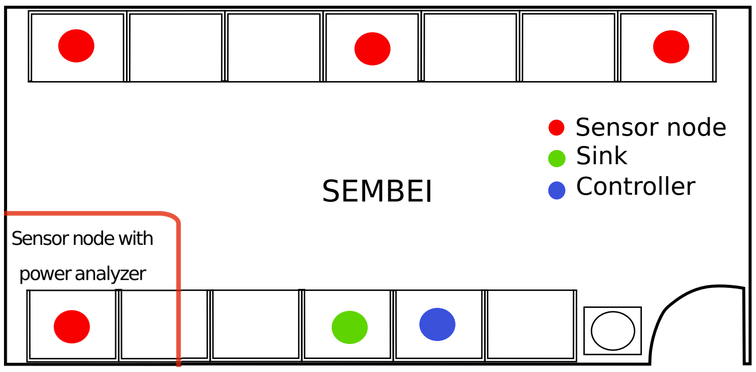


Figure 2. Network deployed for energy consumption measurements.

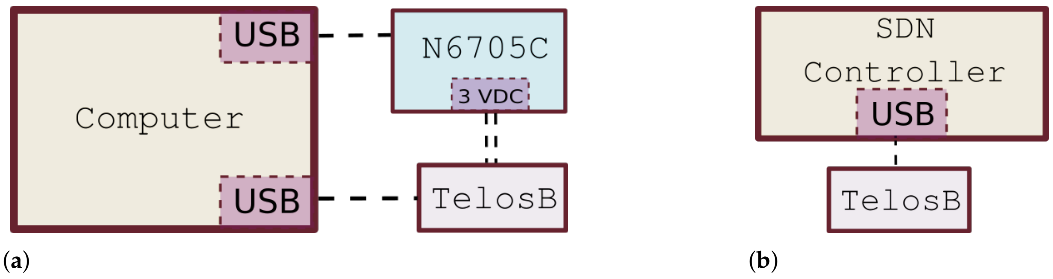


Figure 3. Connection diagrams for (**a**) energy consumption measurement using the N6705C of Keysight and (**b**) the SDN controller and TelosB gateway.

We compared the node estimation and the N6705C measurement to calculate the energy consumption model accuracy. During the testing, all the sensor nodes send temperature measurements to the sink and energy consumption measurements to the controller every 20 s. The experiment was replicated ten times, taking ten energy consumption measurements per time. We obtained an average error of 4.98% and a confidence interval of 1.53% using a significance level of 0.05.

5.2. Prediction Mechanisms Performance Evaluation

The energy consumption prediction proposal performance was evaluated from simulations using COOJA [44] and emulating sky motes, which is a TelosB equivalent available in COOJA simulator. We also executed experiments running the prediction algorithm in the WSN nodes, such as proposed by Rahimifar et al. [18], which for the best of our knowledge, it is the most recent work about energy consumption prediction using Markov chain in SDWSN.

Both approaches were simulated using multihop grid topologies from 16 nodes to 81 nodes, including one controller node and one sink node. We also simulated the scenario without prediction to use it as a baseline. In this case, all WSN nodes send their energy consumption to the SDN controller every 60 s. In parallel, we run a temperature monitoring application where every node sends a temperature sample (2 bytes message) every 60 s as well. Furthermore, all nodes wait 60 s plus a random value between 0 and 60 s before initializing the prediction algorithm and the temperature application. We use the random value to prevent all nodes from transmitting energy and temperature packets at the same time, which would create packets collisions.

The metrics to evaluate the energy consumption prediction proposals include prediction performance and network performance metrics. These metrics are prediction accuracy, total delay, total delivery rate, packets overhead, and energy consumption. The total delay is the average time the packets spent to reach the destination. The prediction accuracy is evaluated in two ways: prediction error and valid predictions rate. The prediction error is the average error of all the predictions during the test. The valid predictions rate is the number of predictions with an error lower than 5% divided by the total predictions performed. We chose 5% based on the average estimation error we obtained using our energy consumption model (Section 5.1). The total delivery rate is calculated by dividing the number of packets successfully received by the number of packets sent. The packets overhead is quantified as the number of packets per minute. The energy consumption is the average energy consumption per minute of all nodes, excluding the sink node and the controller node.

The total delay, total delivery, and overhead metrics are classified by type of packet. Control packets are those related to the IT-SDN control plane, data packets are the application packets sent to the sink, and energy packets are the packets related to the prediction mechanisms and energy information. The graphics in this section show the average results calculated among replications. Table 3 shows the abbreviations used for each scheme, and Tables 4 and 5 summarize the simulation and energy consumption parameters, respectively.

Table 6 represents the cost per transitions in microjoules, i.e., matrix B in Equation (3). The cases with cost zero are transitions that never occurred or that we were not able to measure with our equipment. Last, according to our tests, the initial state is always Processing. Thus, the initial vector $X_0 = [0 \quad 1 \quad 0 \quad 0 \quad 0]$.

Table 3. Abbreviations for the schemes simulated.

Abbreviation	Scheme
No-Prediction	Without prediction
Markov-Node	Markov chain running in the sensor node
Markov-PC	Markov chain running in the controller

Table 4. Simulation parameters.

Simulation Parameters	
Topology	Square grid
Number of nodes	16, 25, 36, 49, 64, 81
Number of sinks	1
Data traffic rate	1 packet every 60 s
Data payload size	2 bytes
ContikiMAC channel check rate	8 Hz
Radio medium	Unit disk graph medium
Distance between nodes	50 m
Transmission range	55 m
IT-SDN parameters	
Neighbor discovery protocol	Collect-based
Link metric	ETX
Controller discovery protocol	none
Route calculation algorithm	Dijkstra
Route recalculation threshold	20%
Flow table size	15 entries

Table 5. Energy consumption parameters.

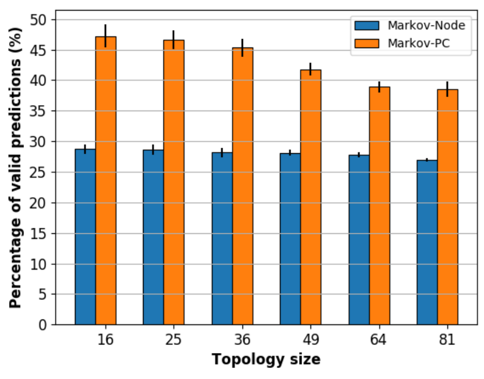
Energy Consumption Parameters	
Transmission current consumption	20.3 mA
Receiving current consumption	18.75 mA
Processing current consumption	2.21 mA
Sleeping mode current consumption	0.67 mA
temperature sensor current consumption	2.3 mA
Operation voltage	3 V

Table 6. Transitions' energy consumption (μJ).

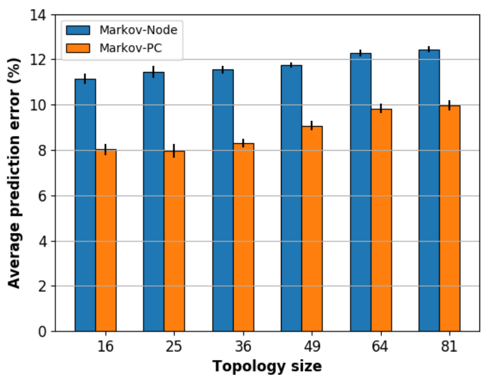
	Sensing	Processing	LPM1	Transmitting	Receiving
Sensing	0	0	0	0	0
Processing	75.6	0	1.35	5.94	0
LPM1	75.6	1.35	0	0	7.38
Transmitting	0	10.8	0	0	0
Receiving	0	21.6	0	0	0

5.2.1. Results Analysis

Figure 4a depicts the percentage of valid predictions and Figure 4b depicts the average prediction error metric. The Markov-PC scheme obtained the highest percentage of valid predictions and the lowest prediction error, which means it is the most effective to reduce energy packets updates. Moreover, we observed that the topology size has more impact on the Markov-PC scheme than on the Markov-Node scheme. In terms of percentage of valid predictions, the average results in the Markov-scheme decrease 8.68%, from 16 nodes to 81 nodes, while for the Markov-Node scheme the decrease is 1.76%.



(**a**)



(**b**)

Figure 4. Markov chain prediction accuracy: (**a**) shows the percentage of valid predictions (<5%) for the case running on the sensor node and running on the SDN controller and (**b**) shows the average prediction error for these two cases as well.

Figure 5 depicts the average energy consumption including the scenario without prediction. Comparing both Markov chain schemes with the No-Prediction scheme, the Markov-PC scheme reduces the average energy consumption of the networks with sizes from 16 to 49 nodes, whereas the Markov-Node scheme increases the average energy consumption for all sizes. This means that the Markov-Node scheme energy consumption trade-off is not positive: it spends more energy than the amount it is able to save. On the other hand, the Markov-PC scheme, the one with the better prediction performance, saves approximately 1% of energy consumption in the topologies between 16 and 49 nodes but increases this metric 1% for 64 nodes and 5% for 81 nodes. If we go back to the percentage of valid predictions in Figure 4a and compare those results with the energy consumption results for Markov-PC scheme, we can say that it is necessary over 40% of valid predictions to save energy.

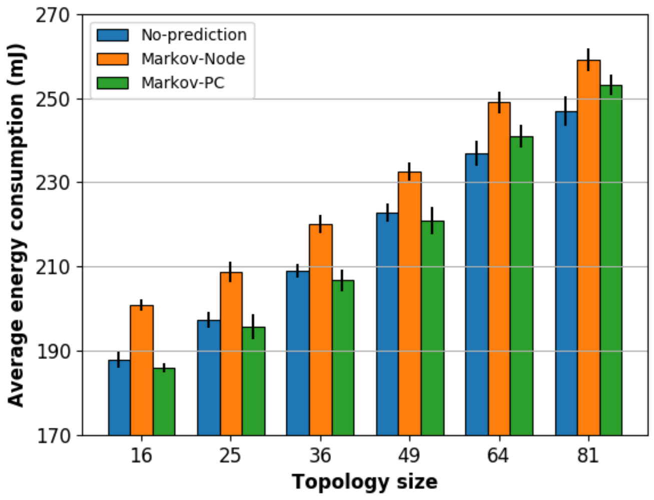
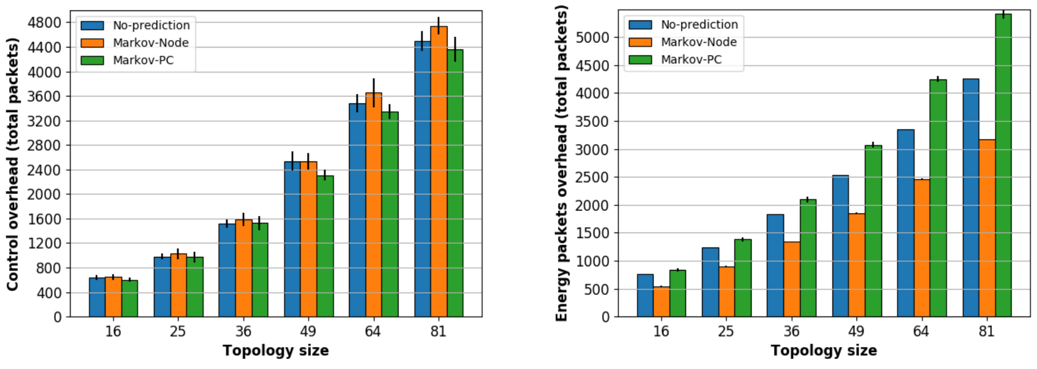


Figure 5. Energy consumption of No-Prediction and Markov chain schemes: values in milijoules.

Figure 6a,b depicts the control and energy packets overhead, respectively. Regarding control packets overhead, the results show similar values for all the schemes in topologies from 16 to 36 nodes. This means that the prediction schemes implementation do not increase the control overhead for small networks. Moreover, we observed that the Markov-PC reduces the control overhead in around 3% and 9% for 49, 64, and 81 nodes. On the other hand, the Markov-Node scheme increases the control overhead between 4.9% and 5.4% for topologies with 64 and 81 nodes. This increase is related to the control packets delivery rate, which is analyzed further below.



(**a**) (**b**)

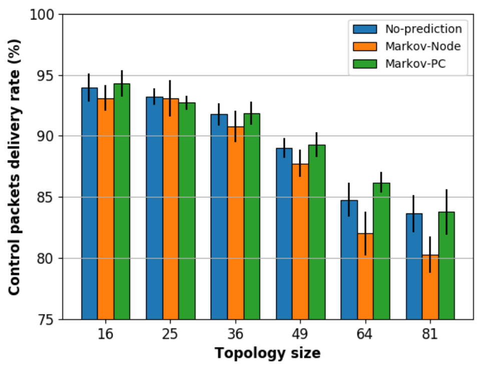
Figure 6. Packets overhead of Markov chain and No-Prediction schemes: (**a**) shows the number of control packets and (**b**) shows the number of energy packets.

Then, the results in Figure 6b show that the Markov-Node scheme reduces the energy packets traffic while the Markov-PC scheme increases it. Even though the Markov-PC scheme has a lower prediction error and a higher percentage of valid predictions than the Markov-Node scheme, the increase is because in the Markov-PC scheme the controller replies every energy information update with the energy consumption prediction result, but in the Markov-Node scheme this is not necessary. This means that using the Markov-PC we need 50% or more valid predictions to at least equal the No-Prediction scheme. On the other hand, around 27% of valid predictions (Figure 4a) in the Markov-Node scheme are enough to reduce up to 25% of the energy packets traffic.

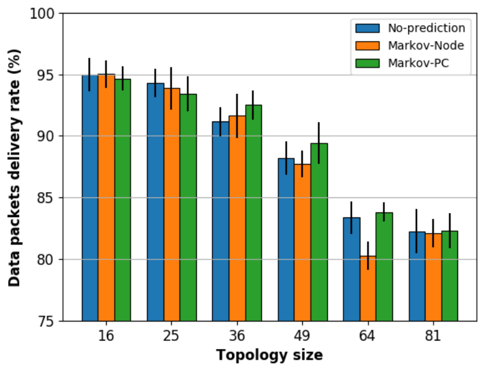
Figure 7a–c depicts the delivery rate for control, data, and energy packets, respectively. From those figures, we observed that the Markov-PC scheme equals or improves the delivery rate with respect to the No-Prediction scheme results. For example, in the case of the energy packets delivery rate, the result for the Markov-PC are between 1.8% and 4.4% over the No-Predictions scheme.

On the other hand, using the Markov-Node scheme the delivery rate is below the No-Prediction scheme results in most cases. This is interesting since the Markov-Node scheme obtained the lowest energy packets overhead, with up to 40% less than the Markov-PC scheme. To understand this, we have to remember that in WSNs all packets share the same band and channel in some cases. Despite the Markov-Node obtained the lowest energy packets overhead, it also obtained the highest control packets overhead, which can affect the energy packets delivery rate. Furthermore, in the Markov-Node scheme all energy packets go upward (i.e., from sensor nodes to the controller) while in the Markov-PC scheme half of them go upward and the other half go downward (i.e., from the controller to sensor nodes). This helps to alleviate the congestion of the routes hence the delivery rate.

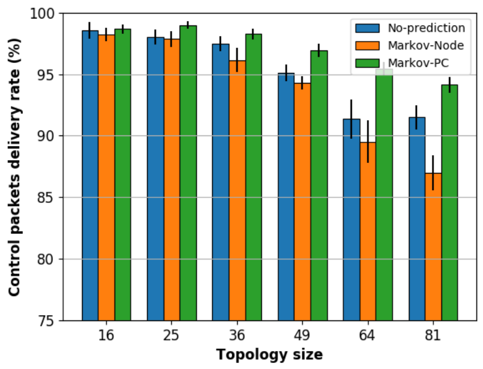
The control packets delivery rate results also helped us to understand the increase in the control overhead using the Markov-Node scheme for 64 and 81 nodes noticed before (Figure 6a). In Figure 7a, we observed that for topologies from 16 to 49 nodes, the decrease is less than 1.25%, while for 64 and 81 nodes, the decrease is around 3% and 4%, which is a match to the increase in the control packets overhead. These metrics are related since the overhead is caused by the retransmissions for the lost packets. A lower delivery rate is a sign that more retransmissions were required.



(**a**)



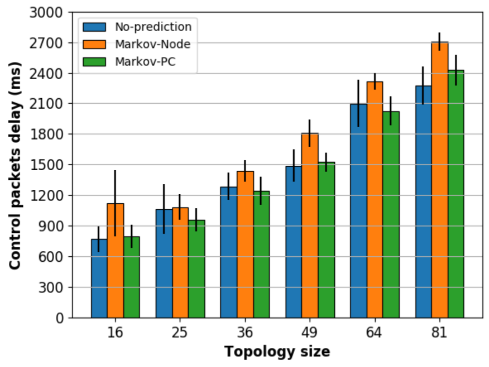
(**b**)



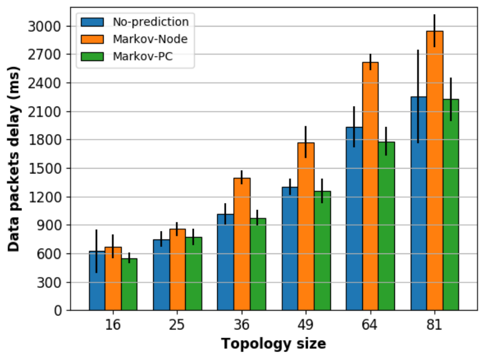
(**c**)

Figure 7. Total Delivery rate of Markov chain and No-Prediction schemes: (**a**) shows the delivery rate of control packets, (**b**) shows the delivery rate of the data packets, and (**c**) shows the delivery rate of energy packets.

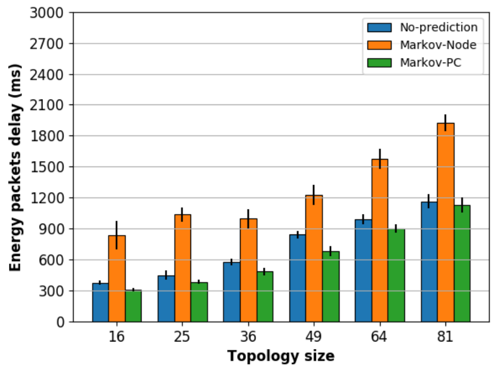
Figure 8a–c depicts the average delay for control, data, and energy packets, respectively. The first we observed is that the Markov-Node scheme obtained the highest delay in all the packet types, with a significant difference in the energy packet. In this case, the delay is from 65% to 122% higher than the delay in the No-Prediction scheme. Unlike the Markov-Node scheme, the Markov-PC scheme was able to maintain the control and data packets delay with respect to the No-Prediction scheme, but also it was able to reduce the delay up to 18% in the case of the energy packets.



(**a**)



(**b**)



(**c**)

Figure 8. Packets delay of Markov chain and No-Prediction schemes: (**a**) shows the average delay of control packets, (**b**) shows the average delay of the data packets, and (**c**) shows the average delay of energy packets.

The increase in the delay using the Markov-Node scheme is caused by its processing overhead. As shown in Figure 9, the Markov-Node scheme doubles the processing overhead of the Markov-PC and No-Prediction schemes. This processing overhead delays energy packets forwarding. Furthermore, as energy and data updates have the same period, it also delays data packets forwarding. The Markov-PC and No-Prediction schemes have similar processing overheads. Thus, recalling the overhead and delivery rate results, despite the fact that the Markov-Node scheme reduces the number of energy packets in the network, the high delay caused by processing overhead could be affecting both energy packets and data packets delivery rate.

Summarizing, the Markov-PC scheme overcomes the Markov-Node scheme in all the metrics evaluated, except for the energy packets overhead. This means, executing the prediction in the SDN controller we obtain better accurate prediction and delivery rate, and less energy consumption, control packets overhead, delay and sensor nodes' processing overhead. Then, with respect to the No-Prediction, the Markov-PC scheme reduces the energy consumption for topologies between 16 nodes and 49 nodes and increases it for topologies between 64 and 81 nodes. We also observed the Markov-PC scheme increases the energy packets delivery rate and reduces their delay. Moreover, in terms of control and data packets delivery rate and delay, the Markov-PC and No-Prediction scheme obtained a similar performance.

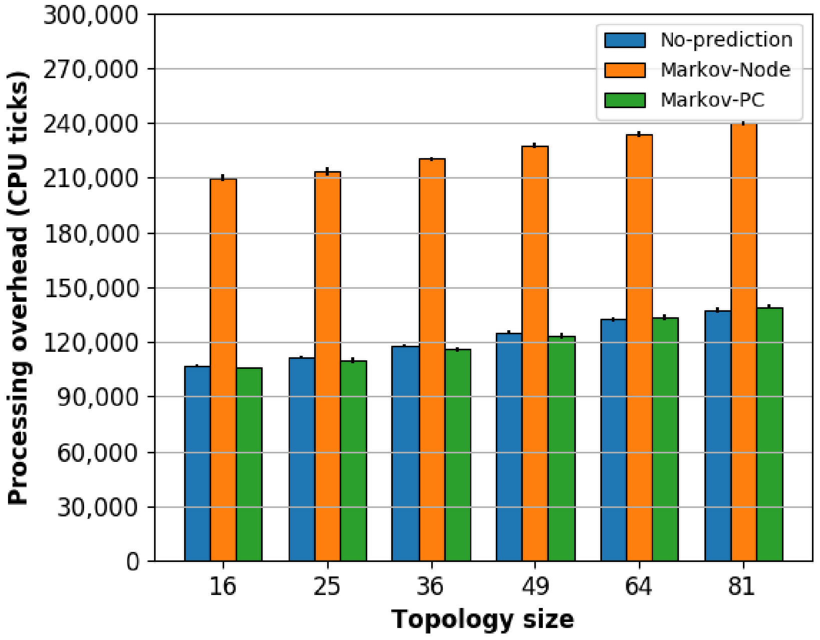


Figure 9. Processing overhead in seconds of No-Prediction and Markov chain schemes.

Thus, considering the whole picture, the Markov-PC scheme obtained better performance than the Markov-Node scheme, but also it provides benefits in terms of energy efficiency and network performance if compared with the No-Prediction scheme.

6. Conclusions

In this work, we proposed a centralized energy consumption prediction mechanism for WSN leveraged by SDN. Our hypothesis was that by moving the prediction processing overhead to the SDN controller we can improve the prediction accuracy and consequently the network performance. To test our hypothesis, we proposed an energy consumption model and implemented a prediction algorithm using Markov chain. Then, we evaluated the prediction and network performance when running the algorithm in every node and running it only in the SDN controller. Furthermore, to determine the real benefit, we compare the performance of both approaches with the case without prediction.

Results showed that by moving the prediction algorithm processing overhead from every node to the SDN controller we improved the prediction accuracy, delivery rate and delay of control, data and energy packets. Moreover, we reduced the control packets overhead, however there is a significant increase in the energy packets traffic.

Concerning the No-Prediction scheme, the Markov-PC scheme reduced the energy consumption, the control packets overhead and the energy packets delay. Furthermore, it improved the energy packets delivery rate. In the remaining metrics, excluding the energy packets overhead, the Markov-PC and the No-Predictions scheme obtained similar performance. These results show our centralized approach improves the prediction accuracy and network performance with less than 50% of predictions attempts correct. Therefore, there is room to improve the prediction accuracy and, thus, the network performance.

Author Contributions: Conceptualization, G.A.N.S. and C.B.M.; methodology, G.A.N.S. and C.B.M.; software, G.A.N.S.; validation, G.A.N.S. and C.B.M.; formal analysis, G.A.N.S.; investigation, G.A.N.S.; resources, C.B.M.; data curation, G.A.N.S.; writing—original draft preparation, G.A.N.S.; writing—review and editing, C.B.M.; visualization, G.A.N.S.; supervision, C.B.M.; project administration, C.B.M.; funding acquisition, G.A.N.S. and C.B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001. Cintia B. Margi is supported by CNPq fellowship (Pq) #307304/2015-9. Gustavo A. Nunez Segura is supported by Universidad de Costa Rica.

Data Availability Statement: The traces of the simulations we used for the performance analysis are available in the repository of Universidade de São Paulo in http://repositorio.uspdigital.usp.br/handle/item/303. The dataset's name is: *Data set of the paper: Centralized Energy Prediction in Wireless Sensor Networks Leveraged by Software-Defined Networking*.

Acknowledgments: We would also like to thank Keysight Technologies for lending their N6705C power analyzer, and Bruno Menezes for the training to use this equipment.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
MEMS	Microelectromechanical systems
SDN	Software-Defined Networking
SDWSN	Software-Defined Wireless Sensor Networks
WSN	Wireless sensor networks

References

1. Culler, D.; Estrin, D.; Srivastava, M. Overview of Sensor Networks. *IEEE Comput.* **2004**, *37*, 41–49. [CrossRef]
2. Mainetti, L.; Patrono, L.; Vilei, A. Evolution of wireless sensor networks towards the Internet of Things: A survey. In Proceedings of the SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks, Split, Croatia, 15–17 September 2011; pp. 1–6.
3. Kobo, H.I.; Abu-Mahfouz, A.M.; Hancke, G.P. A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. *IEEE Access* **2017**, *5*, 1872–1899. [CrossRef]
4. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69. [CrossRef]
5. Haleplidis, E.; Pentikousis, K.; Denazis, S.; Salim, J.H.; Meyer, D.; Koufopavlou, O. *Software-Defined Networking (SDN): Layers and Architecture Terminology*; Technical Report; Internet Research Task Force (IRTF): online, 2015. [CrossRef]
6. Luo, T.; Tan, H.P.; Quek, T.Q.S. Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Commun. Lett.* **2012**, *16*, 1896–1899. [CrossRef]
7. Trevizan, B.; Batista, G.; Borges, C. TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. *IEEE Lat. Am. Trans.* **2015**, *13*, 3690–3696. [CrossRef]
8. Galluccio, L.; Milardo, S.; Morabito, G.; Palazzo, S. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. *Proc. IEEE INFOCOM* **2015**, *26*, 513–521. [CrossRef]
9. Alves, R.C.A.; Oliveira, D.A.G.; Nunez Segura, G.A.; Margi, C.B. The Cost of Software-Defining Things: A Scalability Study of Software-Defined Sensor Networks. *IEEE Access* **2019**, *7*, 115093–115108. [CrossRef]
10. Pantazis, N.A.; Nikolidakis, S.A.; Vergados, D.D. Energy-Efficient Routing Protocols in Wireless Sensor Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 551–591. [CrossRef]
11. Radi, M.; Dezfouli, B.; Bakar, K.A.; Lee, M. Multipath Routing in Wireless Sensor Networks: Survey and Research Challenges. *Sensors* **2012**, *12*, 650–685. [CrossRef]
12. Oliveira, D.A.G.; Margi, C.B. Combining Metrics for Route Selection in SDWSN: Static and Dynamic Approaches Evaluation. In Proceedings of the 2018 IEEE 10th Latin-American Conference on Communications (LATINCOM), Guadalajara, Mexico, 14–16 November 2018; pp. 1–6. [CrossRef]
13. Parissidis, G.; Karaliopoulos, M.; Baumann, R.; Spyropoulos, T.; Plattner, B. Routing metrics for wireless mesh networks. In *Guide to Wireless Mesh Networks*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 199–230.
14. Rezaei, Z.; Mobininejad, S. Energy Saving in Wireless Sensor Networks. *Int. J. Comput. Sci. Eng. Surv.* **2012**, *3*, 23–37. [CrossRef]
15. Samarah, S.; Al-Hajri, M.; Boukerche, A. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Trans. Veh. Technol.* **2011**, *60*, 656–663. [CrossRef]
16. Dias, G.M.; Bellalta, B.; Oechsner, S. Using data prediction techniques to reduce data transmissions in the IoT. In Proceedings of the 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 12–14 December 2016; pp. 331–335. [CrossRef]
17. Mini, R.A.F.; Do Val Machado, M.; Loureiro, A.A.F.; Nath, B. Prediction-based energy map for wireless sensor networks. *Ad Hoc Netw.* **2005**, *3*, 235–253. [CrossRef]
18. Rahimifar, A.; Kavian, Y.S.; Kaabi, H.; Soroosh, M. Predicting the energy consumption in software defined wireless sensor networks: A probabilistic Markov model approach. *J. Ambient. Intell. Humaniz. Comput.* **2020**, 1–14 [CrossRef]

19. Miranda, C.; Kaddoum, G.; Bou-Harb, E.; Garg, S.; Kaur, K. A Collaborative Security Framework for Software-Defined Wireless Sensor Networks. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 2602–2615. [CrossRef]
20. Núnez, G.A.; Margi, C.B. Energy Map Model for Software-Defined Wireless Sensor Networks. In Proceedings of the XXXV Brazilian Symposium on Telecommunications and Signal Processing, Sao Pedro, Brazil, 3–6 September 2017; Volume 3.
21. Praveen Kumar, D.; Amgoth, T.; Annavarapu, C.S.R. Machine learning algorithms for wireless sensor networks: A survey. *Inf. Fusion* **2019**, *49*, 1–25. [CrossRef]
22. Jemal, A.; Hachicha, M.; Halima, R.B.; Kacem, A.H.; Drira, K.; Jmaiel, M. Energy saving in WSN using monitoring values prediction. *Procedia Comput. Sci.* **2014**, *32*, 1154–1159. [CrossRef]
23. Nagadivya, S.; Manoharan, R. Energy Efficient Markov Prediction Based Opportunistic Routing (Eempor) For Wireless Sensor Networks. In Proceedings of the 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 29–30 March 2019; pp. 1–6. [CrossRef]
24. Haas, C.; Wilke, J.; Stöhr, V. Realistic Simulation of Energy Consumption in Wireless Sensor Networks. *Wirel. Sens. Netw.* **2012**, *7158*, 82–97. [CrossRef]
25. Manoharan, L.; Leni, A.E.S. Distributed Uneven Clustering Mechanism for Energy Efficient WSN. *Wirel. Pers. Commun.* **2021**, 1–17.
26. Gupta, P.; Tripathi, S.; Singh, S. RDA-BWO: Hybrid energy efficient data transfer and mobile sink location prediction in heterogeneous WSN. *Wirel. Netw.* **2021**, 1–20. [CrossRef]
27. Tang, F.; Fadlullah, Z.M.; Mao, B.; Kato, N. An Intelligent Traffic Load Prediction-Based Adaptive Channel Assignment Algorithm in SDN-IoT: A Deep Learning Approach. *IEEE Internet Things J.* **2018**, *5*, 5141–5154. [CrossRef]
28. Thupae, R.; Isong, B.; Gasela, N.; Abu-Mahfouz, A.M. Machine Learning Techniques for Traffic Identification and Classifiacation in SDWSN: A Survey. In Proceedings of the IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018; pp. 4645–4650. [CrossRef]
29. Segura, G.A.N.; Chorti, A.; Margi, C.B. Multimetric Online Intrusion Detection in Software-Defined Wireless Sensor Networks. In Proceedings of the 2020 IEEE Latin-American Conference on Communications (LATINCOM), Santo Domingo, Dominican Republic, 18–20 November 2020; pp. 1–6. [CrossRef]
30. Gustavo, N.S.; Skaperas, S.; Chorti, A.; Mamatas, L.; Cintia, B.M. Denial of Service Attacks Detection in Software-Defined Wireless Sensor Networks. In Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 Jun 2020; pp. 1–7
31. Zhou, H.Y.; Luo, D.Y.; Gao, Y.; Zuo, D.C. Modeling of Node Energy Consumption for Wireless Sensor Networks. *Wirel. Sens. Netw.* **2011**, *3*, 18–23. [CrossRef]
32. Ahmad, A.; Javaid, N.; Imran, M.; Guizani, M.; Alhamed, A.A. An Advanced Energy Consumption Model for terrestrial Wireless Sensor Networks. In Proceedings of the 2016 International Wireless Communications and Mobile Computing Conference (IWCMC), Paphos, Cyprus, 5–9 September 2016; pp. 790–793. [CrossRef]
33. Bolch, G.; Greiner, S.; De Meer, H.; Trivedi, K.S. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
34. Abo-Zahhad, M.; Farrag, M.; Ali, A.; Amin, O. An energy consumption model for wireless sensor networks. In Proceedings of the 5th International Conference on Energy Aware Computing Systems Applications, Cairo, Egypt, 24–26 March 2015; pp. 1–4. [CrossRef]
35. Wang, Q.; Yang, W. Energy Consumption Model for Power Management in Wireless Sensor Networks. In Proceedings of the 2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, San Diego, CA, USA, 18–21 June 2007; pp. 142–151. [CrossRef]
36. Texas Instruments. *CC2420 2.4 GHz IEEE 802.15.4/ZigBee-Ready RF Transceiver Applications (Rev. C)*; Texas Instruments: Dallas, TX, USA, 2017.
37. Texas Instruments. *CC1120 High-Performance RF Transceiver for Narrowband Systems*; Texas Instruments: Dallas, TX, USA, 2015.
38. Microchip. *IEEE 802.15.4 2.4 GHz RF Transceiver Features*; Microchip: Chandler, AZ, USA, 2010.
39. Texas Instruments. *CC2520 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver Applications*; Texas Instruments: Dallas, TX, USA, 2007.
40. Dunkels, A.; Gronvall, B.; Voigt, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 16–18 November 2004; pp. 455–462.
41. Dunkels, A. *The Contikimac Radio Duty Cycling Protocol*; SICS Technical Report T2011:13; SICS: Singapore, 2011
42. Alves, R.C.A.; Oliveira, D.A.G.; Gustavo, N.S.; Margi, C.B. IT-SDN: Improved architecture for SDWSN. In Proceedings of the XXXV Brazilian Symposium on Computer Networks and Distributed Systems, Belem, Brazil, 15–19 May 2017.
43. Lin, K.; Zhao, H.; Yin, Z.; Luo, D. An Energy Prediction Clustering Hierarchy for Wireless Sensor Networks. In Proceedings of the 2007 2nd IEEE Conference on Industrial Electronics and Applications, Harbin, China, 23–25 May 2007; pp. 2268–2272. [CrossRef]
44. Osterlind, F.; Dunkels, A.; Eriksson, J.; Finne, N.; Voigt, T. Cross-Level Sensor Network Simulation with COOJA. In Proceedings of the 2006 31st IEEE Conference on Local Computer Networks, Tampa, FL, USA, 14–16 November 2006; pp. 641–648. [CrossRef]