Article

Energy-Efficient Method for Wireless Sensor Networks Low-Power Radio Operation in Internet of Things

Mehdi Amirinasab [1], Shahaboddin Shamshirband [2,3,*], Anthony Theodore Chronopoulos [4,5], Amir Mosavi [6,7] and Narjes Nabipour [8,*]

[1] Internet of Things Laboratory, Hamedan 6514867953, Iran; Mehdi.amiri.n@gmail.com
[2] Department for Management of Science and Technology Development, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[3] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[4] Department of Computer Science, University of Texas, San Antonio, TX 78249, USA; antony.tc@gmail.com
[5] Department of Computer Engineering & Informatics, University of Patras, Patras, 26500 Rio, Greece
[6] Institute of Structural Mechanics, Bauhaus-Universität Weimar, D-99423 Weimar, Germany; amir.mosavi@uni-weimar.de
[7] Faculty of Health, Queensland University of Technology, 130 Victoria Park Road, Kelvin Grove QLD 4059, Australia
[8] Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
[*] Correspondence: shahaboddin.shamshirband@tdtu.edu.vn (S.S.); narjesnabipour@duytan.edu.vn (N.N.)

Abstract: The radio operation in wireless sensor networks (WSN) in Internet of Things (IoT) applications is the most common source for power consumption. Consequently, recognizing and controlling the factors affecting radio operation can be valuable for managing the node power consumption. Among essential factors affecting radio operation, the time spent for checking the radio is of utmost importance for monitoring power consumption. It can lead to false WakeUp or idle listening in radio duty cycles and ContikiMAC. ContikiMAC is a low-power radio duty-cycle protocol in Contiki OS used in WakeUp mode, as a clear channel assessment (CCA) for checking radio status periodically. This paper presents a detailed analysis of radio WakeUp time factors of ContikiMAC. Furthermore, we propose a lightweight CCA (LW-CCA) as an extension to ContikiMAC to reduce the Radio Duty-Cycles in false WakeUps and idle listening though using dynamic received signal strength indicator (RSSI) status check time. The simulation results in the Cooja simulator show that LW-CCA reduces about 8% energy consumption in nodes while maintaining up to 99% of the packet delivery rate (PDR).

Keywords: Internet of Things; IoT; wireless sensor networks; ContikiMAC; energy efficiency; duty-cycles; clear channel assessments; fog computing; smart sensors; signal processing; received signal strength indicator (RSSI)

1. Introduction

The advancement of hardware systems for IoT is an essential research domain of large-scale wireless sensor networks (WSN). In this realm, the development of low-power wireless communication is of utmost importance. Since these types of devices are usually powered by low capacity batteries to provide both sensing and actuation capabilities, managing power consumption is one of the major challenges in designing their hardware and software. The radio is the greatest source of power consumption in a sensor node. Although the sensor network improves and enhances the performance of other technologies, it also presents challenges due to some of their inherent

characteristics and location in a variety of environments. The nodes comprising a wireless sensor network are expected to be small, reliable, low cost, and low power. In some situations, the nodes are in hard-to-reach areas, so their power supply is usually battery or harvest energy from the environment. In both models of power supply the management of power consumption in the nodes is mandatory, in the former case to avoid battery replacement and in latter case to reduce the cost of the energy harvesting system [1–3].

Contiki OS is a well-known lightweight operating system that can be used to manage low-power wireless platforms based on wireless internet communication. In wireless networks, in many cases, it is necessary to relay data between nodes to reach the destination. The radio duty cycle (RDC) protocols available on Contiki OS allow end nodes and even relay nodes to sleep and save energy between each message sending or relay, so the network lifetime is increased. ContikiMAC is a radio duty cycling protocol available in Contiki OS based on the low power listening (LPL) mechanism [4]. It uses the periodic WakeUps method to monitor the communication medium for ongoing transmissions from neighbor nodes. ContikiMAC tries to reduce the amount of energy consumed by radio activities by using pairs of clear channel assessments (CCA) in every sleep and wake up sequence and providing a model of radio management. CCA identifies to recognize the difference interference by measuring the received signal strength indicator (RSSI) thresholds [5,6].

The major power consumption in ContikiMAC is spent on the WakeUps of nodes. The WakeUp is known as a single radio check with a chance to detect the activity of a radio. The WakeUps can be considered in three categories, positive WakeUps, which results in the receipt of the packet, false WakeUps, which results in noise or interferences, and idle listening [7]. Idle listening occurs when the communication medium monitoring does not detect any activity on the radio. Implementing the CCA model on WSN hardware to recognize the different interference is extremely challenging due to the limitations of the low power radio, such as TI CC2420. The power consumption over radio listening mode (Rx) is unavoidable. Nodes should periodically listen to the radio channel to reduce the communication latency in-network, even if data traffic is slow [8]. In this paper, we carefully investigate the time factors associated with a radio WakeUp on the ContikiMAC and to reduce the radio power consumption by dynamic radio check time in WakeUps on the Rx mode. RSSI status check is an essential timing parameter during radio check-in nodes. Consequently, in this research particular attention has been paid to RSSI status check to enable an efficient manipulation of radio check within the limits during the Rx. The maximum timing of radio check is used when RSSI is either new or identified as a positive WakeUp. Otherwise, the false WakeUp or idle listening are associated with minimum timing of radio check. In the proposed method the radio check is essentially based on RSSI where the use of CCA is minimized. CCA is used only for the purpose of RSSI validation and classification.

The rest of the paper is structured as follows. Section 2 provides an overview of related works. Section 3 identifies the hardware and software tools, as well as the scenario used in this paper. Section 4 highlights the challenge of ContikiMAC in WakeUps as a problem statement. Section 5 proposes a lightweight clear channel assessment (LW-CCA) method to reduce power consumption in WakeUp mode. Section 6 contains simulations to compare the proposed method to the ContikiMAC base method. Section 7 presents concluding remarks and future work.

2. Related Works

One of the most important challenges in implementing IoT applications is the power management of hardware platforms. Nowadays, researchers are using many new technologies to minimize radio power consumption so that they can pave the way for IoT. The communication to sending and receiving data on the network is typically the most energy-consuming task in IoT applications [9]. Since the communication of the nodes is in the radio WakeUp state, so a lot of effort has been devoted to designing energy-efficient radio WakeUp models in the last decades. Different methods based on hardware and software have been suggested to control the radio WakeUp mode. Magno et al. [10] have considered the combination of energy harvesting WakeUp receiver and LoRa radio technology to design a new IoT node for long and short-range networking. In another method

in [11], BLE technology and WakeUp radio are integrated with energy harvesting. The design of both proposed methods is hardware-based so that a dual-radio mechanism using separate components is used in Node radio structure on a single chip that is expensive for IoT devices. When implementing WSN on a large scale, small hardware size and low cost are very important factors. CTP-WUR [12], Guo et al. [13], and WUR-MAC [14] have introduced other methods that are more responsive to channel changes using protocol-based design. Recently, cognitive radio has been used as a solution to the opportunistic spectrum access in WSN [15]. ZIPPY [16] reduces latency and power consumption as a synchronization method using WakeUp radio and the MAC synchronization protocol.

Node mobility is one of the important challenges in designing a wake-up schedule (e.g., [17]). Failure to consider node mobility, where node mobility is likely, leads to excessive overhead and as well as this results in poor schedule performance. In addition, the design of WakeUP-based radio systems should also address the challenges associated with WakeUp radio signal propagation in the forest, industrial, or indoor body environments. One of the important issues in low-power radios under WakeUp states is the coexistence with unrelated radio. For example, in-home automation systems and medical technologies, 802.15.4 radios are commonly used but the interference of these types of radios with waves of radio like Wi-Fi leads to a decrease in network performance. Another issue of discussion among researchers is the coexistence between 802.15.4 radios and unrelated radios that lead to the classification, detection, and reduction methods [7]. Airshark [18] and WiFiNet [19] get spectrum information using powerful Wi-Fi hardware and detect nonWi-Fi interference. Another method based on interference information is DOF [20] that provides the local wireless information plane. In the other method in [21], 16 ZigBee channel is scanned to classify spectrum characteristics. The authors, in [22], design a framework to scan the 2.4 GHz band. The beacons are periodically detected to be identified as Wi-Fi signals in ZiFi [23] and ZiFind. This method is dependent on long-term sampling. SoNIC [24] enables resource-limited sensor nodes to detect the type of interference they are exposed to and select an appropriate mitigation strategy. The key insight underlying SoNIC is that different interferers disrupt individual 802.15.4 packets in characteristic ways that can be detected by sensor nodes.

Tang et al. [25] proposed a CCA threshold adaptation method to reduce the impact of interference on packet loss in WSN. Under Wi-Fi interference, the proposed method results showed that increasing the CCA threshold has the effect of reducing the CCA collision and, consequently, improving the WSN packet delivery rate. In the proposed method, the CCA threshold is adjusted adaptively based on the transmit buffer overflows rate in the node. Since there is only one CCA check before transmitting in Zigbee, this approach is only Zigbee-based and is not the case for MAC protocols that are based on multiple CCA per transmission. AEDP [6] is an adaptive energy detection protocol for LPL, which dynamically adjust a node's CCA threshold to improve network reliability and duty cycle based on application-specified bounds. AEDP can effectively mitigate the impact of noise on radio duty cycles while maintaining satisfactory link reliability.

Tang et al. present interference aware adaptive clear channel assessment (IAACCA), which more proactively contends for channel access by replacing the standard CCA [26]. In this method, unlike the Zigbee that performs single CCA, the sequence of CCAs is performed until the channel is found to be clear. Under Wi-Fi interference, IAACCA reduces packet loss compared to standard CSMA mechanism. In IAACCA, a policy decision is adapted after collision with interference. In the other proposed method, ZiSense [27] reduces false WakeUps by using an active scanning technique in duty cycling MAC protocols. The authors in Zisense present one approach to realizing DCCA by RSSI sampling at high frequency, listening for timing and spectral characteristics indicative of 802.15.4.P-DCCA [25] proposed an optimized approach to differentiating clear channel assessment (DCCA) so that a variety of output power is considered in the transmission mode. A P-DCCA check indicates two states, when the transmission medium is occupied by another WSN node, and when the channel is occupied by external interference. This method is based on ContikiMAC, the radio duty cycle protocol.

The above methods are either hardware-based or in some case, they are based on WiFi radio. In some methods, they are based on a single CCA in every WakeUp as opposed to the ContikiMAC protocol. Nevertheless, some of them have high complexity in terms of code and also dependent on the sink coordination, routing control, and signal modulation. In the current paper, LW-CCA (light weight clear channel assessment) is presented as an extension of ContikiMAC. It is designed to reduce energy consumption caused by idle listenings and false WakeUps in the ContikiMAC concerning the high packet delivery rate (PDR) [28]. LW-CCA mechanism is a non-complex method, and it is independent to sink and to route mechanism by minimum complexity in programming. The LW-CCA, through focusing on the time factors in radio wake-up, offers a method to reduce the percentage of radio duty cycles in false WakeUps and idle listenings. Finally, LW-CCA is compared via simulation to the basic ContikiMAC. The proposed method can be used on low power IOT platforms based on 802.15.4 radio and Contiki OS, such as Z1, Tmote-SKY, and Micaz [29].

3. Software Platform and Simulation Tools

In the present study, we use the latest version of Contiki 3.0. As mentioned above, Contiki is an open-source operating system for the Internet of Things that supports tiny low-cost, low-power microcontrollers connected to the Internet. We focus on Contiki available features such as platforms, network protocols, and radio duty cycles to provide an optimized way to reduce energy consumption in IoT nodes [30]. In this research, we use the Cooja simulator available at Contiki Os to simulate the network scenario and run the proposed method on the nodes. The collect-view data gathering software is used to evaluate the status of network nodes in terms of different parameters. One of the strengths of Cooja is the ability to simulate radio medium activity, that the researchers can graphically analyze the radio states [31,32].

3.1. Platform

In the present study, we use the Tmote-Sky platform as the network nodes with MSP430 microcontroller for network simulation. Contiki OS considers the value of 32,768 Hz as tick per the second parameter in the RTIMER_SECOND variable based on the MSP430F1611 basic clock module [33]. The frequency of a real-time clock varies with the application. The frequency 32,768 Hz (32.768 kHz) is used because it is a power of 2 (2^{15}) value. Also, one can get a precise 1 s period (1 Hz frequency) by using a 15 stage binary counter. RTIMER_SECOND variable has a key role in calculating radio activity times [34].

The radio used in the Tmote-Sky Platform is CC2420. The CC2420 is an IEEE 802.15.4 compliant RF transceiver 2.4 GHz designed for low-power and low-voltage WSN applications. CC2420 is controlled via SPI port, and a series of digital output and input, as well as interrupts by MSP430. The CC2420 driver in Contiki OS provides two types of radio check, CCA and RSSI. CCA is based on the measured RSSI value and a programmable threshold [35]. The RSSI value is averaged over eight symbol periods (128 μs), in accordance with [26]. MCU SPI pin takes strobe commands registers and sends them to CSn Radio pin for calibration. There are types of registers, status bits, and pins for radio control in CC2420 driver in Contiki OS. Table 1 lists the ones which are used in three ON (), RSSI (), and OFF () functions in the course of an idle listening channel check by RSSI according to CC2420.c file available in Contiki OS.

Table 1. Participants in one idle listening based on CC2420 driver source code in Contiki OS

Parameter	Type	ON()	RSSI()	OFF()
SRXON	register			
SRFOFF	register			✓
RSSI	register		✓	
SNOP	register		✓	✓
SFLUSHRX	register			✓
RXFIFO	register			✓

RSSI_VALID	status bit		*	
T_X_ACTIVE	status bit			*
CSN	pin		*	
FIFOP	pin	*		

3.2. Energest Module

In implementing the proposed method we use the Enegest module feature to configure the application on nodes. Contiki uses the Energest software-based module to estimate the power consumption of the nodes. With this module, time spent for every sensor node is measured in some states such as CPU, LPM, Tx, and Rx in real-time. The energy estimation module is called when the component is turned on to produce a time stamp. The power consumption of nodes is calculated as

$$P = P\_CPU + P\_LPM + P\_Rx + P\_Tx \tag{1}$$

The average power consumption of each node (*P*) is sum of the average power consumption in *CPU* state (*P_CPU*) that is activated whenever the node is active (the real-time the *CPU* is active without using the radio transceiver is *CPU −Tx –Rx*), *LPM* state (*P_LPM*) that is activated when the sensor node goes to low power mode, *Rx* state (*P_Rx*) that node is active in the radio receive mode and *Tx* state (*P_Tx*) that radio is active in transmit mode. The battery voltage (Vcc) and current power consumption in the indicated state are set based on platform datasheet. In fact, the energy consumption in states is calculated based on the number of CPU ticks based on microcontroller, current power consumption in indicated state, and battery voltage [36,37]. The Tmote-Sky power parameters are considered according to Table 2 as follows.

Table 2. Power parameters in Tmote-Sky based on [38]

Variable	Power Current Consumption State	Value	Unit
VCC	Supply voltage	3	volt
PC_CPU	MCU on, Radio off	1.8	mW
PC_LPM	MCU idle, Radio off	0.0545	mW
PC_Tx	MCU on, Radio Tx	17.7	mW
PC_Rx	MCU on, Radio Rx	20	mW

3.3. Network Scenario

Since the purpose of this paper is not merely to provide a method for simple and one hop networks, the basis of this research is based on a multi-hop random network in order to make the proposed method as practical as possible. In the multi-hop random scenario, nodes are exposed to different numbers of neighbors, so interference has a greater impact on nodes. Nodes also have different amounts of RSSI. On the other hand, the network packets are more likely to be lost. In addition, each node experiences a different number of retries in sending packets [39]. Therefore, in the random scenario, the power consumption in the nodes is not monotonous. Figure 1 shows a scenario and network graph based on RPL routing protocol such that 20 sender nodes and one sink node or border router that they are scattered in the environment dimensions 100 square meters (100 × 100 m) randomly. In the current study, nodes are considered to be fix. The transmitter range at all nodes is equal to 50 m.
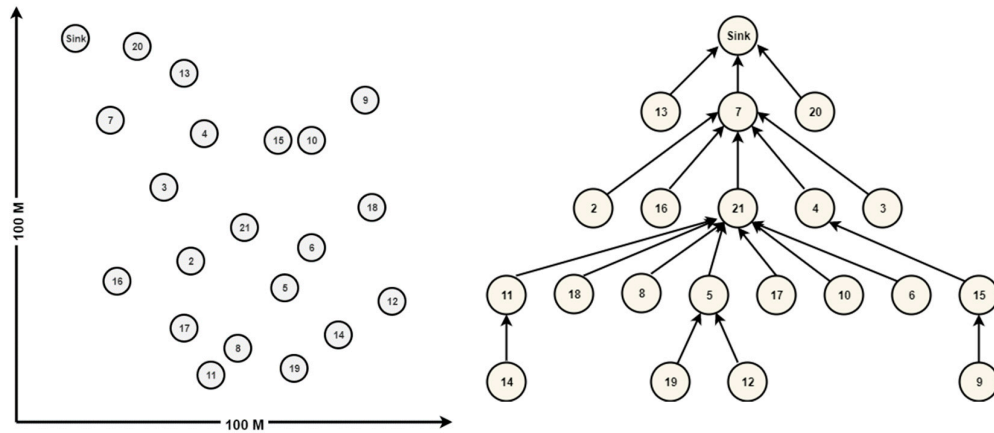
Figure 1. Simulation scenario and network graph based on the RPL routing protocol.

3.4. Network Protocols

The Network layer protocol's stack on the nodes list in Table 3 is among the most widely used in WSN. We use the feature of a collect-view application available in Contiki OS to allow data to be aggregated and categorized by a node in the network. It basically involves a sink node and one (or more) sender nodes. Data such as sensor, power, and network information is sent from the sender node to the sink node, and the collect-view (a java-based application available in Contiki OS Cooja) displays them in a graphical form [40]. In this work, the IPv6 Routing Protocol for LLNs (RPL) is set as a proactive routing protocol. RPL automatically forms a tree topology by exchanging ICMPv6 control packets to find a path to the root. 6LowPAN (i.e., IPv6 over Low -Power Wireless Personal Area Networks) is an adaptation layer protocol that allows the transport of IPv6 packets over 802.15.4 links. Carrier sense multiple access with collision avoidance (CSMA/CA) adopted for Mac layer as the IEEE 802.15.4 standard, and finally, ContikiMAC are used as the RDC layer for control 802.15.4 radio based on Low Power Listening (LPL). The RDC layer in the sink is set to Null_RDC, so the sink does not sleep during network life.

Table 3. Network layers structure used on the nodes [36]

Layer	Protocol	Standard
Application	Collect view	-
Transport	UDP	IETF RFC 768
Network	RPL/IPv6	IETF RFC 6550
Adaptation	6lowpan	IETF RFC 6282
Data link	IEEE 802.15.4 MAC (CSMA)	IEEE 802.15.4
Radio Duty Cycling	ContikiMAC	-
Physical	IEEE 802.15.4 PHY	IEEE 802.15.4

4. Problem Statement

From the moment a ContikiMAC node is turned on, the radio check is repeatedly adapted to send and receive data types, such as routing packets, medium control packets, or packets containing environment sensor information. Each node performs a large number of WakeUp Rx modes depending on the environment. Two conditions can be considered: 1) the node is ideally positioned over minimum interference with other unrelated radio or noise signal; 2) the node is in a state of coexistence with unrelated radios, noise, or interference. In the state 1, between each data reception (positive WakeUp) the node will have a large number of idle listening that plays a significant role in the activity of a radio. In the state 2, in addition to idle listening, the node also has many false WakeUp. Figure 2 shows samples of positive, false, and idle listening WakeUp on the Radio Timeline

in Cooja simulator. The *x*- and *y*-axes represent to node numbers and their activity of radio, respectively. In this simulated example, the power configuration and the network layers of the nodes are according to Tables 2 and 3, respectively. The red, blue, and green colors in the activity of radio line show interference, sending packet, and receiving packet respectively [41].

Figure 2. Positive, false, and idle listening WakeUp in Cooja timeline.

The ContikiMAC considers CCA_CHECK_TIME that is the time it takes to perform a CCA check. In fact, CCA_CHECK_TIME is a balancing time for other timing parameters in ContikiMAC, and it has no direct effect on CCA duration. The time spent on each false WakeUp is depending on MAX_NONACTIVITY_PERIODS. It is the maximum number of periods we allow the radio to be turned on without any packet being received. Each period counts as a sum of CCA_CHECK_TIME and CCA_SLEEP_TIME. The CCA_SLEEP_TIME is the time between two successive CCA checks. The time variables in ContikiMAC that affect both idle listening and false WakeUp times are shown in Table 4. The fast-sleep mechanism is responsible for the diagnosis of false WakeUps in ContikiMAC. Therefore, if the node fails to receive data after 11 periods (21.1 ms), it quickly returns to sleep mode.

Table 4. Amount of effect on idle listenings and false WakeUps according to ContikiMAC.c

Variable	CPU Ticks	Unit (ms)
CCA_CHECK_TIME	32768/8192	0.4
CCA_SLEEP_TIME	(32768/2000) + 1	1.7
MAX_NONACTIVITY_PERIODS	10 × (CCA_CHECK_TIME + CCA_SLEEP_TIME)	21

Table 5 shows the number of idle listenings and false WakeUps during the 10-min simulation in the 5, 7, 12 nodes according to Sections 3.3 and 3.4. For example, the time spent in idle listening and false WakeUp in Node7 is calculated as Equation (2). In fact, almost 1.6% of the network time is wasted in idle listening and false WakeUp in Node 7. This is, of course, a mere indication of the effect of the wake-up time loss on idle listenings and WakeUps, and it is not a definite value. Thus except for the positive WakeUps, the node tolerates inefficient energy consumption in idle listenings and false WakeUps. Because this value varies depending on the application and location of the nodes.

Table 5. Number of idle listenings and false WakeUps in nodes 5, 7, and 12

Node Number	Idle Listenings	False WakeUp
5	11,357	115
7	13,895	220
12	12,082	50

$$T_{node\ 7} = T_{idle} + T_{false}$$
$$T_{node\ 7} = (13895 \times 0.4) + (220 \times 21) = 5558 + 4620 = 10178 \text{ ms} \tag{2}$$

5. Lightweight Clear Channel Assessment (LW-CCA)

In this paper, we propose the LW-CCA, a lightweight, low-complexity programming method designed to minimize WakeUp power on ContikiMAC nodes. In this method, it attempts to reduce the time spent in Rx mode by maintaining a PDR rate similar to base ContikiMAC. By focusing on software and hardware RSSI status check time, LW-CCA reduces the percentage of duty cycles as much as possible to reduce radio power consumption. The minimum WakeUp time for a medium check would take at least one idle listening. In Section 5.1, a single idle listening is analyzed to identify the essential time factors. Furthermore, the evaluation of the minimum and maximum RSSI status check time on nodes is presented in Section 5.2. In Section 5.3, the LW-CCA node categories RSSI based on the CCA verification, whether they are predicted as a false WakeUp, idle listening, or positive WakeUp. Finally, in Section 5.4, the performance of LW-CCA for the classification and validation of RSSI and the dynamic radio check time are described. Worth mentioning that LW-CCA runs exclusively on the relay and sender node, not sink or server node. The proposed method is considered when the nodes are fixed in the network. The following sections describe the details of the proposed method.

5.1. Time Factors in a Single RSSI Radio Check

The basis of LW-CCA is the momentary check of RSSI. In this study, we examine the time parameters involved in an RSSI radio check. The relationship between ContikiMAC functions, the CC2420 driver, and the Energest module is shown in Figure 3. In general, each RSSI radio check consists of three phases:

- Phase 1: Checking permissions for radio driver access by RDC, registering radio hardware to Rx mode by the radio driver, and recording start time of Rx by Energest.
- Phase 2: Validating the RSSI and returning RSSI value from related radio register.
- Phase 3: Set the radio registers to 'off' state, Preparing the radio queue for the next stage of radio activity and also announce the end of Rx state to Energest module.

Since our goal is to manage WakeUp time based on radio potentials, so analysis and retrieval of time factors affecting WakeUp are inevitable. Figure 3 shows a single WakeUp diagram based on the RSSI check.
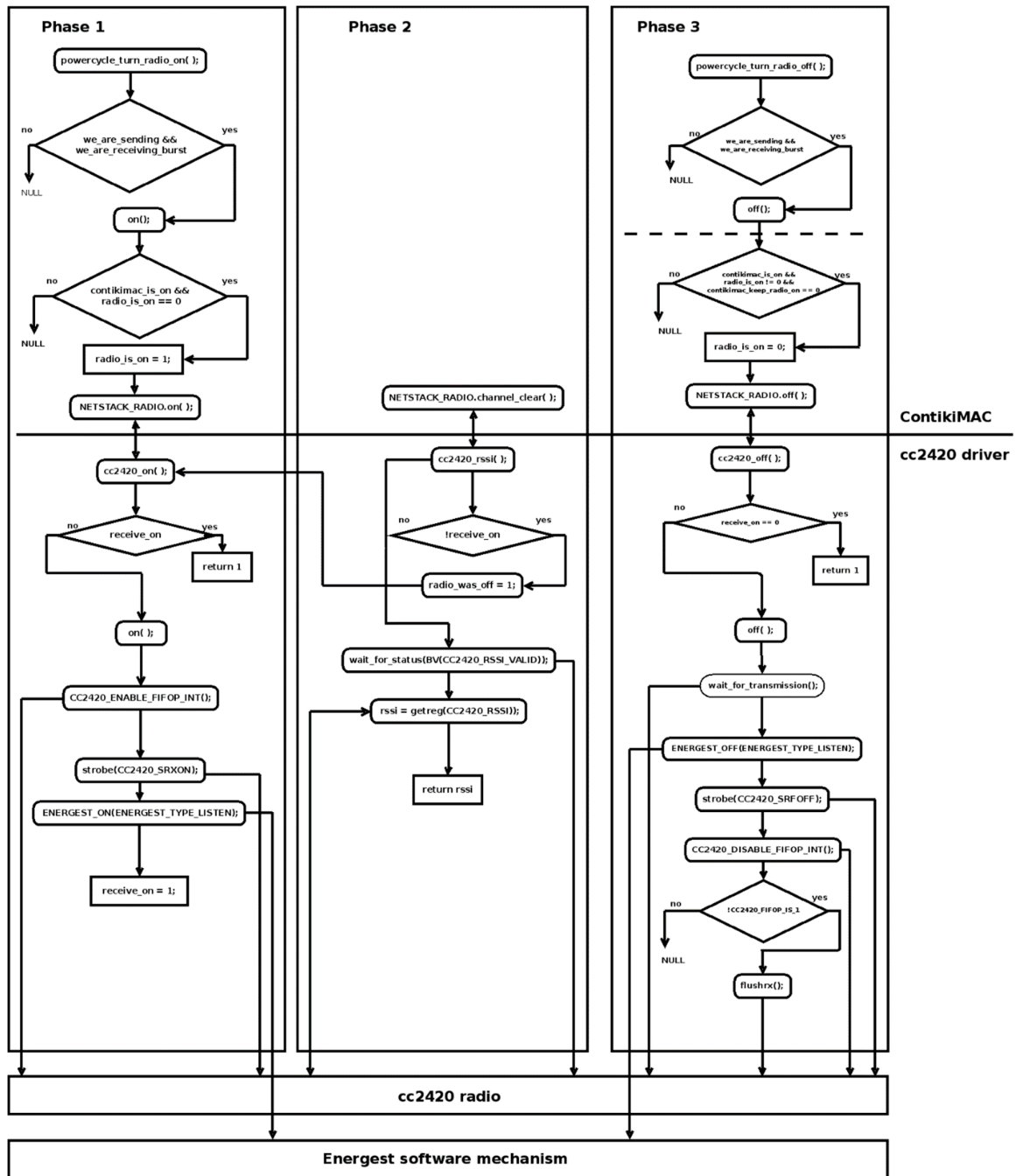
Figure 3. Diagram of the relationship between RDC, the radio driver, and the Energest module in performing a single RSSI.

Every radio WakeUp lasts between strobe CC2420_SRXON and strobe CC2420_SRFOFF. The number of CPU ticks in the radio check is saved in Energest. The wait_for_status (CC2420_RSSI_VALID) in phase 2 is known as an important time factor for a single RSSI check. It takes time equal with Equation (3).

$$T_\Delta = \frac{\text{RTIMER\_SECOND}}{10} \tag{3}$$

Table 6 shows time factors which are resulted from analyzing the relationship between RDC and radio hardware in an RSSI check. The table is extracted from the radio Timeline in Cooja.

Table 6. Effective time factors per idle listening based on RSSI under Cooja simulator timeline

$t\_wait\_for\_status$(ms)	t_RSSI (ms)	$total\_time$ (ms)
0.32	0.128	0.448

Every RSSI check involves the different action timings: t_wait_for_status is time spent for RSSI status check based on Equation (3). t_RSSI is time spent for the read of RSSI status check. Total_time is the sum of the time spent in an RSSI radio check. Actually in an RSSI check, the $t\_wait\_for\_status$ takes the longest time 0.32 milliseconds (20 symbols period according to [42]) to doing a reliable RSSI status check-in ContikiMAC. Therefore, it indicates that ContikiMAC devotes more time to RSSI status checking than the eight-symbol periods suggested in CC2420 datasheet. It should be noted that 20 Symbols has no role in RSSI reception. In fact, it is a BackOFF for reliable time to wake the radio.

5.2. RSSI Check Time Models in LW-CCA

This section examines the hardware and software RSSI status check time. Based on the phase 2 Section 5.1, LW-CCA divides the radio check time into T_Δ and t_RSSI time periods. The following function is responsible for doing the RSSI validation under a period of time t_RSSI = 0.128 ms (8 symbols) and T_Δ = 0.32 ms (20 symbols) in cc2420.c.

```
 static void
wait_for_status(uint8_t status_bit)
{
rtimer_clock_t t0;
t0 = RTIMER_NOW();
while(!(get_status() & status_bit)
&& RTIMER_CLOCK_LT(RTIMER_NOW(), t0 +
RTIMER_SECOND / 10);
}
```

The LW-CCA, with respect to radio check time partition, considers two models for a dynamic check in the Rx mode.

Model 1: This model considers RSSI check for measuring signal strength in Rx and uses default CCA check-in Tx mode. The RSSI status check time is by default method in CC2420 driver based on the sum of T_Δ and t_RSSI.

Model 2: In this model, considering $T_\Delta = 0$, the RSSI check time is considered to be equal with the eight-symbol period's case. In fact, the RSSI status check is performed over the intended time for the cc2420 radio chip. The rest of the conditions are the same as in Model 1. The following function is responsible for doing the RSSI validation under a period of time t_RSSI = 0.128 ms.

```
 static void
 wait_for_status(uint8_t status_bit)
 {
 rtimer_clock_t t0;
 t0 = RTIMER_NOW();
 while(!(get_status() & status_bit)
 && RTIMER_CLOCK_LT(RTIMER_NOW(),
 t0);
 }
```

Table 7 shows a comparison between indicated models in terms of percentage of the listening duty cycle according to the scenario in Sections 3.3 and 3.4 in a 40 min (High Rate). It shows that ignoring T_Δ in RSSI status check time for nodes based on model 2 results in about 11 percentage reduction in average of the listening duty cycle. The results in Table 7 are extracted based on the Collect-View output. The details of the average listen duty cycle calculation are available in Section 6.

Table 7. Comparison of models of 1 and 2 in terms of percentage of listen duty-cycle

Method	Rx | Tx	Listen Duty-Cycle (%)
Model 1	RSSI | CCA	1.451
Model 2	RSSI | CCA	1.282

Therefore, each RSSI status check can be divided into two states: 8 internal symbols set for CC2420 radio hardware and 20 symbols for performing a reliable radio WakeUp on Rtimer equal to Equation (3). In fact, 8 and 20 symbols can be considered as the minimum and maximum periods required to perform a minimum radio check time, respectively. Figure 4 illustrates the performance of the RSSI status check. The file Msp802154Radio.java in Cooja is responsible for simulating eight symbols for RSSI status check based on CC2420 radio datasheets.
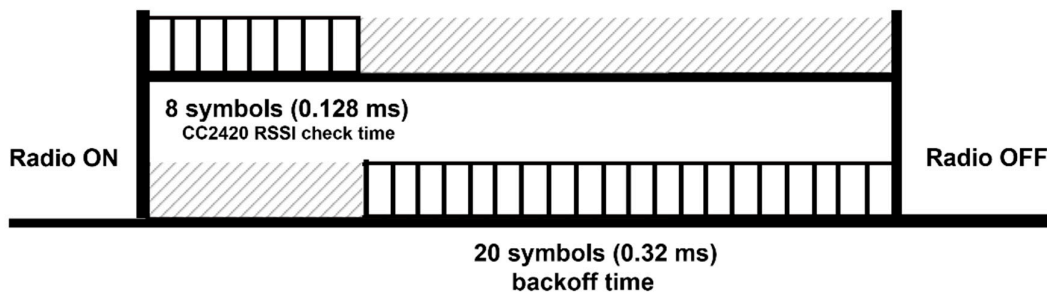
Figure 4. Performance of RSSI status checks based on radio hardware and ContikiMAC.

Figures 5 and 6 show changes in RSSI check time in nodes based on model 1 and 2 in Rx and Tx mode. In this simulated example, the power configuration and the network layers (except for the amount of RSSI check time in ContikiMAC) of the nodes are according to Tables 2 and 3, respectively. The difference between the radio checks is shown in Figure 6 with some zoom to make the difference clearer. In both models, the radio check is assumed to be based on default CCA in Tx mode.
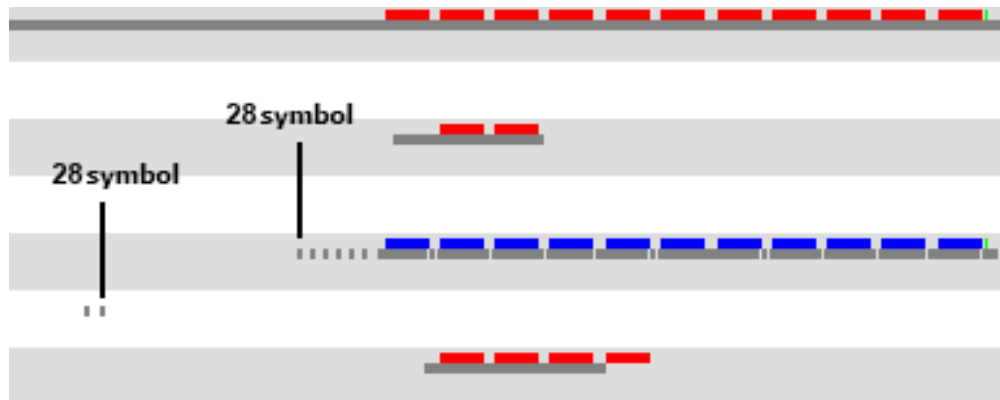
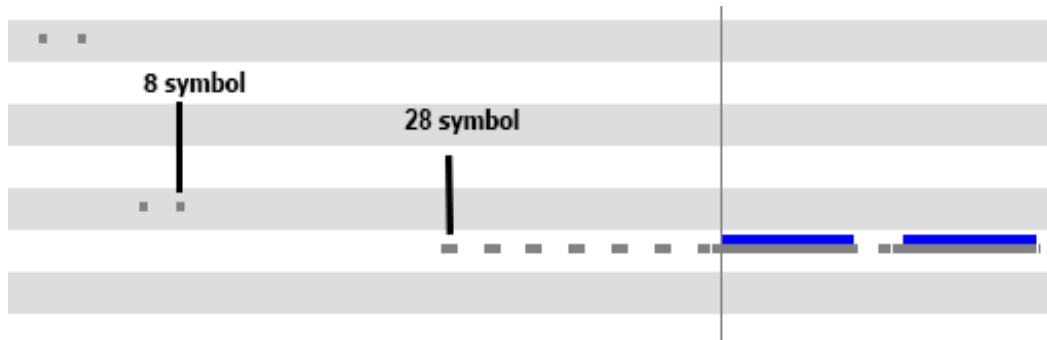**Figure 5.** Model 1: RSSI check-in Rx (28 symbols) and Tx(28 symbols) in Cooja timeline.

Figure 6. Model 2: RSSI check-in Rx (8 symbols) and Tx (28 symbols) in Cooja Timeline.

5.3. Categories of RSSIs in LW-CCA

The CC2420 in our application returns [−100,0] as the normal range of RSSI, where −100 is the minimum level of the noise floor. In performing each RSSI check, the values obtained for RSSI can be divided into three categories. Firstly, the values that are valid and determine the radio activity and ultimately, it results in the receipt of the data frame (positive WakeUP); Secondly, the values that are valid and determine the radio activity and ultimately, it results in the noise or interference (false WakeUp); and third the values that are invalid and do not specify any activity on the radio (idle listening). In the LW-CCA categories, false WakeUp and idle listening RSSIs are located in rssi_null. Figure 7 shows the RSSIs classified. Identification of idle listening and false WakeUp conducted via CC2420 driver's and ContikiMAC's side and the false WakeUp is reported to CC2420 driver through NETSTACK_RADIO [43].
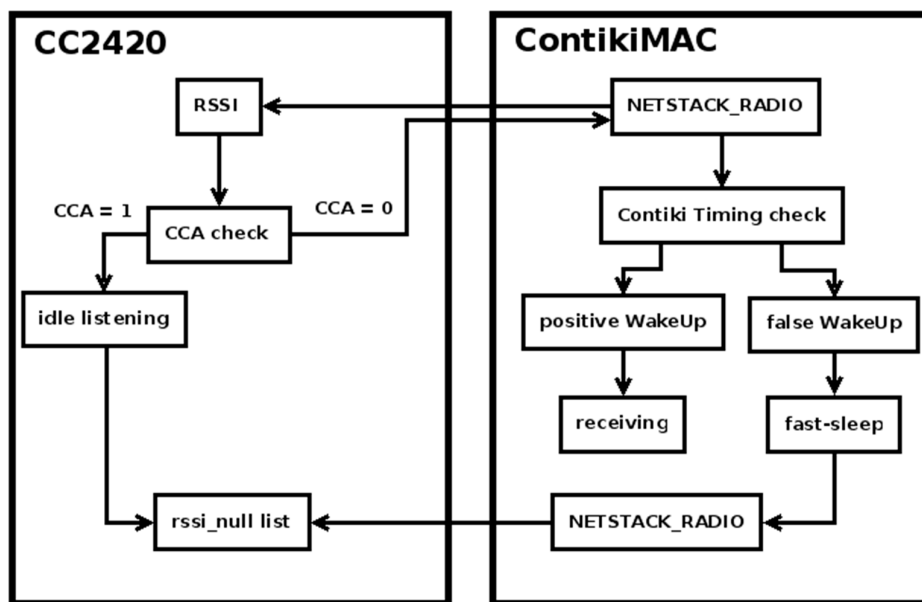

Figure 7. RSSIs classified as null.

It is important to note that the classification of RSSIs depends on the return value of CCA check, so that if CCA value is 0 so ContikiMAC checks false/positive WakeUp, otherwise it is an idle listening. The diagnosis of positive WakeUp and false WakeUp is based on a timing mechanism in ContikiMAC. It is worth mentioning that, each node when it receiving a new RSSI value, performs the validation through CCA.

5.4. Dynamic RSSI Check Time in LW-CCA

This section describes the performance of LW-CCA according to the description of the previous sections. As mentioned, LW-CCA performance is based on RSSI measurements. In this method, for a dynamic check, the radio check time works according to the models 1 and 2 (Section 5.2). Every node has two tasks in receiving RSSIs: i.e., receiving RSSI based on model 1, and re-evaluating RSSI based on the model 2 and its classification.

The proposed method is illustrated in Figure 8, where nodes perform radio checks in two modes: listen and transmit mode. In the LW-CCA method, a reduction in Rx radio check time is considered. Each ContikiMAC node periodically checks the radio with CCA pairs. In fact, each CCA recognizes the validity and invalidity of radio activity by comparing the RSSI threshold. Based on Section 5.1, in LW-CCA each radio check uses the RSSI directly. LW-CCA nodes first assign a value of eight symbols (Model 2) to the status check RSSI. The value obtained is evaluated based on the classification in Section 5.3. In this case, two states are considered:

1/ RSSI is a member of rssi_null list:

If the RSSI value is a member of the rssi_null list, the node returns 1 to ContikiMAC and quickly go to sleep. In fact, the node does just eight-symbols RSSI status check.

2/ RSSI is a member of the rssi_active List or it is a new RSSI:

If the RSSI value is outside the rssi_null list, it is either a new value or a member of rssi_active. In both cases, the node re-evaluates the radio based on Model 1. In this case, after the RSSI check, the CCA check will be considered for validation and classification of RSSI. If CCA returns a value of 1, the RSSI value is detected as an idle listening and stored as a worthless value in the rssi_null list. Otherwise, ContikiMAC applies its timing based on Section 4 to check for false Wake Up or positive WakeUp. If the radio activity in the node results in a false WakeUp the RSSI value is stored in the rssi_null list and the node goes to sleep quickly based on the Fast-Sleep mechanism. If the RSSI results in positive WakeUp the node stays on the receiving mode.
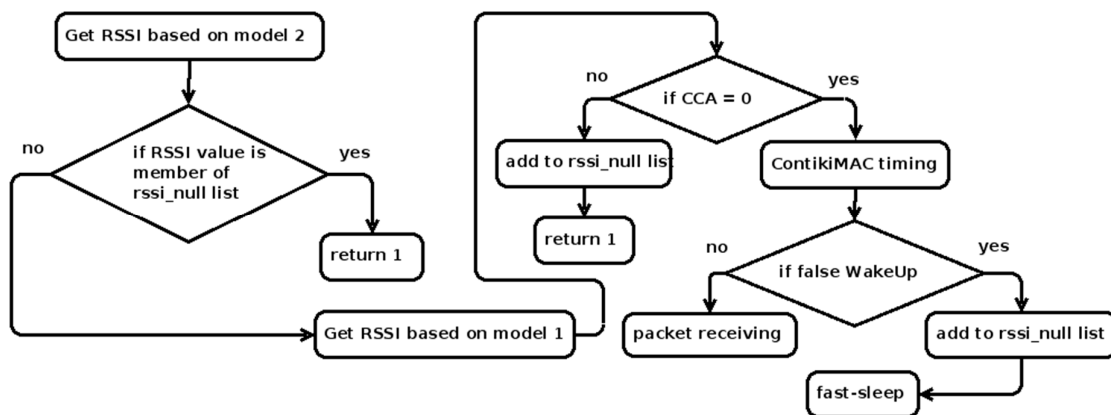

Figure 8. Diagram of the LW-CCA method.

The proposed method actually uses the maximum value of the RSSI status check time when the RSSI value is detected either new or active. Therefore, in other cases, at least eight symbols are considered. LW-CCA thus reduces the amount of radio check-in false WakeUps and idle listening as much as possible to reduces the inefficient power consumption caused by radio activity in Rx mode. Also, using RSSI validation based on the CCA value makes the node performance reliable in the real environment as well.

6. Comparison of LW-CCA with ContikiMAC

We evaluated the proposed method LW-CCA and compared it with ContikiMAC using the Cooja simulator. Tmote-Sky is used as a hardware platform for nodes on the network. The network protocol for all nodes is the same as in Table 3. Figure 1 illustrates the network scenario with respect to the RPL routing protocol graph. It is a random scenario with a variety of hops and neighbors consists of:

- 1 emulated node that is programmed as a sink that plays the role of the root node for RPL in the network graph. In fact, it is a UDP server that collects data from client nodes;
- 21 emulated duty cycle nodes as UDP clients in network graph that send data to sink;
- power consumption of the nodes being estimated by the Energest module available at Contiki.

The LW-CCA method is compared with the ContikiMAC in terms of the average CPU ticks in CPU, LPM, Rx and Tx, CPU Power, LPM Power, Rx and Tx Duty-Cycle(%), Rx and Tx power, total power consumption, and PDR. LW-CCA and ContikiMAC nodes are evaluated based on the two data transmission rates, once every 60 s (low rate) and once every 30 s (high rate). Simulation time for each of low rate and high rate scenario is 4 h. The node number "1" is considered as a sink and its radio is turned on during the network time, so the network status analysis is done based on the sender nodes from 2 to 21. All of the scenarios outputs are based on Collect-View in Cooja. The equations are based on embedded equations for power consumption in the Collect-View source code [31]. The power parameters for simulation are listed in Table 2.

6.1. Average of Ticks in CPU, LPM, Rx, and Tx States in the Network

Tick conversion simply refers to the ability to convert physical units to timer ticks [44]. On a system with a 2 MHz timer, for example, 1 ms is mapped to 2000 ticks. The calculation of the power consumption in different states for the nodes depends on how long the node stays in every state. The Energest considers the time spent in different states with the number of ticks in that state. Table 8 shows the average of ticks in CPU, LPM, Rx and Tx states in both low rate and high rate scenarios in LW-CCA and ContikiMAC nodes. Table 8 shows the average of ticks for CPU (numbers of ticks for CPU in active mode) in CPU_time, LPM(numbers of ticks in low power mode) in LPM_time, Rx (numbers of ticks in listening mode) in Rx_time and Tx(numbers of ticks in transition mode) in Tx_time. The LPM_time shows that LW-CCA by reducing RSSI status check time in false WakeUps and idle listenings, increase low power mode time in nodes. CPU_time simulation result shows that there is a small amount of computing overhead in LW-CCA. Rx_time also shows a significant decrease in listening time in LW-CCA, although there is little change in Tx_time. It should be noted that the parameters are averaged from all of the sender nodes.

Table 8. Average of CPU ticks in SKY nodes

RDC	CPU_time	LPM_time	Rx_time	Tx_time
ContikiMAC (Low Rate)	4498.73	46,354.15	511.15	145.71
ContikiMAC (High Rate)	4834.95	47,584.65	719.45	273.50
LW-CCA (Low Rate)	4537.25	47,478.90	425.40	147.52
LW-CCA (High Rate)	5066.35	48,036.60	601.05	281.30

6.1. Average of Percentage for Listen and Transmit Duty Cycle in the Network

This section analyzes the simulation results for a network (20 sender nodes) in terms of listen and transmit duty cycle that refers to the percentage of the duty cycle used to listen and transmit to the radio. Table 9 shows the percentage of listen and transmit duty cycles based on the low send rate and high send rate for LW-CCA and ContikiMAC nodes. The simulation outputs show that in both scenarios, percentage of listen duty cycle in LW-CCA nodes are significantly lower than ContikiMAC nodes. LW-CCA decreases average of listen duty cycle about 0.2 in both of low high send rate

scenarios that it results in a reduction of approximately 18% in the average total listen duty cycle. The average percentage of transmit duty cycle in LW-CCA does not make much difference to ContikiMAC. The experiment results show that the proposed LW-CCA method can reduce the time of listen duty cycle by considering a dynamic RSSI check time so that it tries to reduce listen time in false WakeUps and idle listenings. How to calculate the average percentage of listen and transmit duty cycle is according to Equations (4) and (5). Rx_time, Tx_time, CPU_time, and LPM_time are equal with numbers of ticks available in Table 8.

$$\text{Rx DutyCycle}(\%) = 100 * (\frac{\text{Rx\_time}}{\text{CPU\_time} + \text{LPM\_time}}) \qquad (4)$$

$$\text{Tx DutyCycle}(\%) = 100 * (\frac{\text{Tx\_time}}{\text{CPU\_time} + \text{LPM\_time}}) \qquad (5)$$

Table 9. Average of Rx and Tx duty cycles (%)

RDC	Rx Duty Cycle (%)	Tx Duty Cycle (%)
ContikiMAC (low rate)	1.005	0.286
ContikiMAC (high rate)	1.372	0.521
LW-CCA (low rate)	0.817	0.283
LW-CCA (high rate)	1.131	0.529

6.2. Network Power Consumption

The total power consumption of each node comprises the power consumption of the node in different states such as CPU, LPM, Rx, and Tx. In this section, the average power consumption on indicated states in LW-CCA nodes is compared with the ContikiMAC nodes in both low and high rate scenarios and evaluates the effect of the proposed method on the total power consumption and finally the packet delivery rate (PDR). The power and time parameters in the power consumption calculation are obtained from Tables 2 and 8. The average of power consumption in CPU, LPM, Rx, and Tx states in network is calculated by Equations (6)–(9) respectively. The average of total power consumption (P) is calculated according to Equation (1). For example, average of listen power for LW-CCA nodes in high rate is calculated in Equation (10).

$$\text{P\_CPU(mW)} = \frac{\text{CPU\_time} \times \text{VCC} \times \text{PC\_CPU}}{\text{CPU\_time} + \text{LPM\_time}} \qquad (6)$$

$$\text{P\_LPM(mW)} = \frac{\text{LPM\_time} \times \text{VCC} \times \text{PC\_LPM}}{\text{CPU\_time} + \text{LPM\_time}} \qquad (7)$$

$$\text{P\_Rx(mW)} = \frac{\text{Rx\_time} \times \text{VCC} \times \text{PC\_Rx}}{\text{CPU\_time} + \text{LPM\_time}} \qquad (8)$$

$$\text{P\_Tx(mW)} = \frac{\text{Tx\_time} \times \text{VCC} \times \text{PC\_Tx}}{\text{CPU\_time} + \text{LPM\_time}} \qquad (9)$$

$$\text{P\_Rx(LW} - \text{CCA, High\_Rate)} = \frac{\text{Rx\_time} \times \text{VCC} \times \text{PC\_Rx}}{\text{CPU\_Time} + \text{LPM\_time}} = \frac{601.5 \times 3 \times 20}{5066.35 + 48036.60} \qquad (10)$$
$$= 0.679(\text{mW})$$

Table 10 compares the network states in terms of average power consumption for LW-CCA and ContikiMAC in CPU, LPM, Rx, Tx states, and finally considers the average total power consumption and packet delivery rates. The outputs show that the average power consumption in the LPM mode for both methods is relatively similar. The average power consumption in the high-rate scenario in

the LW-CCA represents a relatively higher value in CPU state, which may be due to the processing overhead imposed on the CPU in the LW-CCA. However, the reduction in average power consumption in the listening state and finally, the overall average energy consumption in LW-CCA nodes justify this overhead. The LW-CCA is able to reduce average listening power by 0.113 and 0.144 mW in low and high rates, respectively. The average power consumed in the transmit state in both methods shows a relatively similar value. Finally, the overall average power consumption in LW-CCA nodes has decreased by 0.121 and 0.124 mW in low and high rates, respectively compared to ContikiMAC. Table 10 shows that LW-CCA is able to retain a 99% packet delivery rate similar to ContikiMAC by decreasing overall power consumption by 7.1% and 8.7% for high and low rate scenarios, respectively.

Table 10. Average of power consumption in CPU, LPM, Rx, Tx, total power consumption, and PDR

RDC	P_CPU (mW)	P_LPM (mW)	P_Rx (mW)	P_Tx (mW)	P (mW)	PDR (%)
ContikiMAC (low rate)	0.477	0.149	0.603	0.152	1.381	99
ContikiMAC (high rate)	0.498	0.148	0.823	0.277	1.746	99
LW-CCA (low rate)	0.471	0.149	0.490	0.150	1.260	99
LW-CCA (high rate)	0.515	0.147	0.679	0.281	1.622	99

As the outputs of the simulator show, the proposed method has acceptable performance at both the 60 s and 30-s transmission rates. On the other hand, the results show that the performance of the proposed method in saving power in high-throughput applications is reduced. Figure 9 compares average power consumption improvement in the LW-CCA method compared to the basic ContikiMAC method at 60, 50, 40, 30, 20, and 10-s transmission rates. As shown in Figure 9, as the sequence of data transmission in the nodes is shorter, the average energy consumption improvement is also lower. It is due to the higher rate of interaction between the nodes for data transfer. Because network interference is higher and the CPU overhead of nodes is increased to compare the signal intensity. Therefore, the proposed method is suitable for applications where the data transmission times of the nodes are not short.
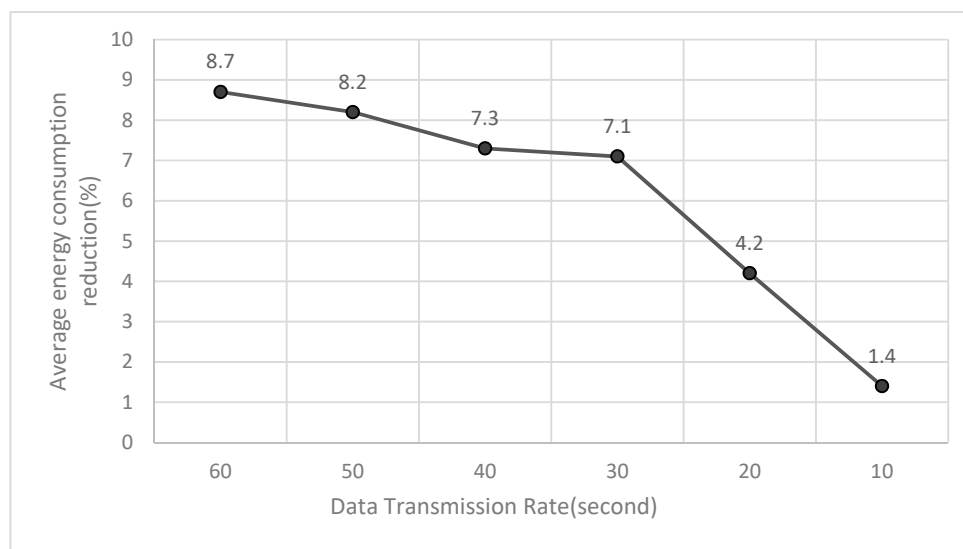

Figure 9. Average improvement of LW-CCA energy consumption at different data transmission rates.

7. Conclusions

The 802.15.4 radios are low power radios that are used in many applications of the IoT and wireless sensor networks on a variety of platforms. Some operating systems offer varieties of radio duty cycle (RDC) mechanism for controlling 802.15.4 radios. One of the challenges in RDC protocols is WakeUps in idle listening and falseWakeUps (Section 4). The proposed method LW-CCA aims to reduce the wakeup time caused by false WakeUp and idle listening nodes. For this purpose, the performance of ContikiMAC is evaluated according to the time factors affecting each WakeUp. RSSI status check time is an important time factor to the radio. ContikiMAC considers a fixed time of 20 symbols (0.32 ms) to perform an RSSI check time per WakeUp. The proposed LW-CCA method, unlike ContikiMAC, considers two values of minimum and maximum 8 symbols (0.128 ms) and 28 symbols for the radio check. (Section 5). The value of eight symbols is the minimum value based on the CC2420 Datasheet. The LW-CCA nodes classify RSSI values based on CCA validation. LW-CCA considers eight-symbol radio check time if RSSI is detected as idle listening or false WakeUp otherwise it considers a 28 symbol. LW-CCA nodes, based on the simulations of Section 6 have reduced 18% of duty cycle compared to ContikiMAC. The based on simulation outputs of Section 6, by considering the power parameters and indicated scenario in Section 3, the LW-CCA is able to reduce approximately 0.12 mW in average power consumed in Rx mode. Therefore, it is able to lower the overall power consumption level about 8% in the nodes is considered high and low data transmission rates.

Furthermore, the LW-CCA reduces energy consumption while it maintains packet delivery rate 99%, which is the same as ContikiMAC in sender nodes. The proposed method can be applied to all IoT platforms based on 802.15.4 radios such as Z1, Tmote-SKY, and Micaz. Since RSSI validation is based on CCA check, it can also be exploited in the real environment. The instantaneous categorization mechanism in the LW-CCA method can make this method flexible against interference caused by irrelevant radios or noise. According to Figure 9, the proposed method is not suitable for applications with high transmission data rates. In fact, the efficiency of the proposed method decreases with increasing interference in the network. The more balanced the node distribution, the better the proposed method will perform in terms of network lifetime.

Author Contributions: Conceptualization, modeling, data curation, data analysis, and analysis of results, M.A., S.S., A.T.C., A.M., and N.N.; IoT, machine learning, and soft computing expertise, M.A., S.S., A.T.C., A.M., and N.N.; Mathematics expertise, M.A., S.S., A.T.C., A.M., and N.N.; Management, database, writing, administration, and methodology M.A., S.S., A.T.C., A.M., and N.N.; Visualization, M.A., S.S., A.T.C., A.M., and N.N.; Supervision, resources, software, revision, and verification of the results, A.T.C., A.M., and S.S., funding, A.M. All authors have read and agreed to the published version of the manuscript.

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

Abbreviations

Abbreviation	Definition
CCA	Clear Channel Assessments
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access with Collision Avoidance
IoT	Internet of Things
LPL	Low Power Listening
LPM	Low Power Mode
LW-CCA	Light Weight CCA
MAC	Medium Access Control
MCU	Microcontroller Unit

PDR	Packet Delivery Rate
RDC	Radio Duty Cycle
RPL	Routing Protocol for LLNs
RSSI	Received Signal Strength Indicator
UDP	User Datagram Protocol
VCC	Voltage at the Common Collector
WSN	Wireless Sensor Network

References

1. Avallone, S.; Guadagno, S.; Emma, D.; Pescapè, A.; Ventre, G. D-ITG distributed internet traffic generator. *Proc.-First Int. Conf. Quant. Eval. Syst.* **2004**, *QEST 2004*, 316–317.

2. Dutta, P.; Dunkels, A. Operating systems and network protocols for wireless sensor networks. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **1958**, *370*, 68–84.

3. Homaei, M.H.; Salwana, E.; Shamshirband, S. An Enhanced Distributed Data Aggregation Method in the Internet of Things. *Sensors* **2019**, *19*, 3173.

4. Thang, V.C. A Comparative Study of Network Performance between ContikiMAC and XMAC Protocols in Data Collection Application with ContikiRPL. *Int. J. Comput. Netw. Inf. Secur.* **2019**, *11*, 32.

5. Bhar, J. A Mac Protocol Implementation for Wireless Sensor Network. *J. Comput. Networks Commun.* **2015**, *2015*, 1–12.

6. Sha, M.; Hackmann, G.; Lu, C. Energy-efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments. *Proc. 12th Int. Conf. Inf. Process. Sens. Networks* **2013**, *61*, 277–288.

7. Zheng, X.; Cao, Z.; Wang, J.; He, Y.; Liu, Y. Interference resilient duty cycling for sensor networks under co-existing environments. *IEEE Trans. Commun.* **2017**, *65*, 2971–2984.

8. Huang, Y.; XIANG, W.; WEN, S.; JIN, Y. The Study of Traffic-Aware ContikiMAC. *DEStech Trans. Comput. Sci. Eng.* **2016**, doi:10.12783/dtcse/wcne2016/5114.

9. Aoudia, F.A.; Magno, M.; Gautier, M.; Berder, O.; Benini, L. Wake-up receivers for energy efficient and low latency communication. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*; California, USA 2016; p. 53. Springer.

10. Magno, M.; Aoudia, F.A.; Gautier, M.; Berder, O.; Benini, L. WULoRa: An energy efficient IoT end-node for energy harvesting and heterogeneous communication. *Proc. 2017 Des. Autom. Test Eur. DATE* **2017**, *2017*, 1528–1533.

11. Khomami, M.M.D.; Rezvanian, A.; Meybodi, M.R. A new cellular learning automata-based algorithm for community detection in complex social networks, *J. Comput. Sci.* **2018**, *24*, 413–426.

12. Basagni, S. CTP-WUR : The Collection Tree Protocol in Wake-up Radio WSNs for Critical Applications. In *2016 International Conference on Computing, Networking and Communications (ICNC)*; Beijing, China 2016; pp. 0–5.

13. Guo, C.; Zhong, L.C.; Rabaey, J.M. Low power distributed MAC for ad hoc sensor radio networks. In *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270)*; Oslo, Norway, 2001; Volume 5, pp. 2944–2948.

14. Mahlknecht, S.; Durante, M.S. WUR-MAC: energy efficient wakeup receiver based MAC protocol. *IFAC Proc.* **2009**, *42*, 79–83.

15. Joshi, G.P.; Nam, S.Y.; Kim, S.W. Cognitive radio wireless sensor networks: applications, challenges and research trends. *Sensors* **2013**, *13*, 11196–11228.

16. Sutton, F.; Buchli, B.; Beutel, J.; Thiele, L. Zippy: On-demand network flooding. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*; 2015; pp. 45–58.

17. Homaei, M. H., Soleimani, F., Shamshirband, S., Mosavi, A., Nabipour, N., & Varkonyi-Koczy, A. R. An Enhanced Distributed Congestion Control Method for Classical 6LowPAN Protocols Using Fuzzy Decision System, *IEEE Access*, 8, **2020**, 20628-20645, doi: 10.1109/ACCESS.2020.2968524.

18. Rayanchu, S.; Patro, A.; Banerjee, S. Airshark: Detecting non-WiFi RF Devices Using Commodity WiFi Hardware. *Proc. 2011 ACM SIGCOMM Conf. Internet Meas.* Seattle, USA. **2011**, 137–154. Springer.

19. Rayanchu, S.; Patro, A.; Banerjee, S. Catching Whales and Minnows using Wifinet: Deconstructing Non-wifi Interference using Wifi Hardware. *USENIX Conf. Networked Syst. Des. Implement.* Milan, Italy, **2012**, 57–70. Elsevier.

20. Hong, S.; Katti, S. DOF: A Local Wireless Information Plane. *Sigcomm* **2013**, Hamburg, Germany, 230–241, Springer.

21. Chowdhury, K.R.; Akyildiz, I.F. Interferer classification, channel selection and transmission adaptation for wireless sensor networks. *IEEE Int. Conf. Commun.* **2009**, 11, 1–5.

22. Bloessl, B.; Joerer, S.; Mauroner, F.; Dressler, F.; Joerer, S. ACM MobiCom 2012 Poster : Low-Cost Interferer Detection and Classification using TelosB Sensor Motes. *Proc. 18th Annu. Int. Conf. Mob. Comput. Netw.* **2012**, *16*, 403–406.

23. Zhou, R.; Xiong, Y.; Xing, G.; Sun, L.; Ma, J. ZiFi: wireless LAN discovery via ZigBee interference signatures. In *MobiCom '10 Proceedings of the Sixteenth annual International Conference on Mobile Computing and Networking*; Madrid, Spain, 2010; 49–60, Springer.

24. Hermans, F.; Ngai, E. SoNIC : Classifying Interference in 802. 15. 4 Sensor Networks. In *ACM*, Tokyo, Japan, 2013; pp. 55–66, Elsevier.

25. King, A.; Roedig, U. Differentiating Clear Channel Assessment Using Transmit Power Variation. *ACM Trans. Sens. Networks* **2018**, *14*, 1–28.

26. Uwase, M.P.; Bezunartea, M.; Tiberghien, J.; Dricot, J.M.; Steenhaut, K. Experimental Comparison of Radio Duty Cycling Protocols for Wireless Sensor Networks. *IEEE Sens. J.* **2017**, *17*, 6474–6482.

27. Zheng, X.; Cao, Z.; Wang, J.; He, Y.; Liu, Y. ZiSense: Towards Interference Resilient Duty Cycling in Wireless Sensor Networks. *Sensys'* **2014**, *14*, 119–133.

28. Oller, J.; Demirkol, I.; Casademont, J.; Paradells, J.; Gamm, G.U.; Reindl, L. Has Time Come to Switch from Duty-Cycled MAC Protocols to Wake-Up Radio for Wireless Sensor Networks? *IEEE/ACM Trans. Netw.* **2016**, *24*, 674–687.

29. Ojo, M.O.; Member, S.; Giordano, S.; Member, S.; Procissi, G.; Seitanidis, I.N. A Review of Low-End , Middle-End , and High-End Iot Devices. *IEEE Access* **2019**, *6*, 70528–70554.

30. Joshi, M.; Kaur, B. Web Integrated Smart Home Infrastructure Using Internet of Things. *Int. J. Eng. Res. Gen. Sci.* **2015**, *3*, 153–158.

31. Velinov, A.; Mileva, A. Running and Testing Applications for Contiki OS Using Cooja Simulator. *Int. Conf. Inf. Technol. Dev. Educ.* 8, **2016**, 279–285.

32. Sadiki, S.; Ramadany, M.; Faccio, M.; Amegouz, D.; Boutahari, S. Running Smart Monitoring Maintenance Application Using Cooja Simulator, In *International Journal of Engineering Research in Africa*; Cape Town, South Africa, 2019; Volume 42, pp. 149–159. Springer.

33. Instruments, T.; Slas, I. MSP430F15x, MSP430F16x, MSP430F161x mixed signal microcontroller. *Internet of things*, 11, **2011, *2002*, 77**.

34. Nasseri, M.; Al-Olimat, H.; Alam, M.; Kim, J.; Green, R.; Cheng, W. Contiki Cooja Simulation for Time Bounded Localization in Wireless Sensor Network. *Proc. 18th Symp. Commun. Netw.* **2015**, 8, 1–7.

35. Mansoor, K.; Ghani, A.; Chaudhry, S.A. Securing IoT-Based RFID Systems: A Robust Authentication Protocol Using Symmetric Cryptography. Sensors **2019**, 19, 4752.

36. Schandy, J.; Steinfeld, L.; Silveira, F. Average power consumption breakdown of Wireless Sensor Network nodes using IPv6 over LLNs. In *2015 International Conference on Distributed Computing in Sensor Systems*; Madrid, Spain. 2015; pp. 242–247.

37. Son, N.M.; Thinh, T.N.; Thi, N.D.; Nhan, N.C.; Technology, I. An Approach of Low Power Wifi Sensor Mote for Internet of Things Applications Faculty of Computer Engineering, Faculty of Computer Science and Engineering, *J. Emerg. Trends Eng. Appl. Sci.* **2016**, *7*, 161–166.

38. SNM. *Tmote Sky Sensor Datasheet*; 2007; pp. 1–28.

39. Michel, M.; Quoitin, B. Technical Report : ContikiMAC vs X-MAC performance analysis. *Netw. Internet Archit.* **2015**, *5*, 1–28

40. Sitanayah, L.; Sreenan, C.J.; Fedor, S. A Cooja-based tool for maintaining sensor network coverage requirements in a building. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*; Seattle, USA 2013; p. 70. Elsevier.

41. Österlind, F.; Eriksson, J.; Dunkels, A. Cooja TimeLine: a power visualizer for sensor network simulation. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*; 2010; pp. 385–386. Athenes, Greece, Elsevier.

42. T.C.O.S.O. for the Internet of Things. Dunkels, Adam, The Contiki Open Source OS for the Internet of Things. Available online: https://github.com/contiki-os/contiki/blob/master/dev/cc2420/cc2420.c (accessed on 12.08.2019).

43. Quan, Y. *Topology-Based Device Self-identification in Wireless Mesh Networks*; School of Electrical Engineering and Computer Science: Stockholm, Sweden, 2019.
44. Baccelli, E.; Gündoğan, C.; Hahm, O.; Kietzmann, P.; Lenders, M.S.; Petersen, H.; Schleiser, K.; Schmidt, T.C.; Wählisch, M. RIOT: An open source operating system for low-end embedded devices in the IoT. *IEEE Internet Things J.* **2018**, *5*, 4428–4440.