Low Packet Latency Using New Radio Duty-Cycle Scheduling Method

Onny Setyawati*1, Abdur R. Muhammad2, Achmad Basuki3

1,2,3 Brawjaya University, Malang
1osetyawati@ub.ac.id, 2abdurrm@ub.ac.id, 3abazh@ub.ac.id
*Corresponding Author

Received 18 August 2023; accepted 13 December 2023

Abstract. As energy conservation in Wireless Sensor Networks is crucial, the scheduling methods are required to ensure that sensor nodes operate for a longer period. Duty cycle scheduling can be accomplished using synchronous, asynchronous, or on-demand methods with additional radio channels. This study has integrated the on-demand method with an asynchronous scheduling mechanism for the wake-up radio. Model simulation results with random placements showed that the Wake-up Asynchronous On-demand Radio Duty-Cycle (WAORDC) method was able to achieve the latency value of 3.3 seconds, better than the CXMAC asynchronous method, and the total energy required was approximately 4% only, to activate the sensor node.

Keywords: duty cycle, on-demand, asynchronous, latency, WAORDC

1 Introduction

An application of the Internet of Things (IoT) within the Personal Area Network is the utilization of wireless communication which offers convenient in design and installation of communication systems. Wireless Sensor Network (WSN) is an example of its practical application, i.e. a network composed of sensors that are capable of exchanging data with each other. A unit of sensors that can communicate is called a sensor node [1], [2]. Sensor nodes implemented in the monitoring system for planting Porang (Amorphophallus muelleri Blume) which was performed with Porang Research Center (PRC) [3], worked only with limited energy sources such as batteries, due to the difficulty for maintenance access, lack of sunlight or wind as an energy source. Porang tuber had a high glucomannan content and high diversity in the East Java region. The system requirements for cultivating Porang plants included environmental parameters such as temperature, humidity, soil moisture, and light intensity, serving as the basis for analyzing the plant's environment and its impact. To enhance energy efficiency some improvements have been made in hardware and software of the sensor nodes used in WSN [4], [5]. These enhancements involved the evaluation of batteries with extended lifetime, the design of antennas and microcontrollers, and the low power transmission systems [6], [7].

An alternative method for saving energy resources includes the implementation of scheduled sleep mode on the sensor nodes [8], [9]. Using the sleep/awake mechanism, also known as the duty cycle scheduling, is a viable approach that can reduce energy consumption during idle-states by operating the radio on sensor nodes in sleep mode. The mechanisms for scheduling are categorized into three distinct types i.e. scheduled rendezvous [10], [11], asynchronous [12], [13], and
on-demand. The scheduled rendezvous is done by synchronizing to determine the communication time span. However, the latency might be increase as the traffic load increases. Alternatively, in the on-demand method, the sensor nodes that require packet transmission would awake the destination nodes within the transmission area.

The usage of an additional radio [14] in the on-demand method generally resulted in a low latency. However, an idle listening on the Wake-up Receive (WRx) radio was required [15]. The WRx remained active while waiting for a beacon or a tone from the communicating node. Sensor nodes must activate additional radios on different channels while awaiting communication from other sensors. Therefore, knowledge pertaining to scheduling these radios is required to minimize the energy usage. Knowing the network latency would not only improve energy efficiency, but also significant for providing optimal routing, ensuring timely data delivery, maintaining QoS requirements, detecting faults and anomalies in the network, and enhance user experiences [16], [17].

This study combined the two previously established methods to improve energy usage by performing on-demand scheduling such as the WaCo method [14] and adding an asynchronous mechanism to the wake-up radio. Energy efficiency in WSN using on-demand duty cycle mechanism required a radio trigger, or wake-up radio with a low power wake-up signal receiver. WRx still consumed a significant amount of energy since it was constantly active even in the absence of network activity. To address these issues, this study focused on saving energy by adding a trigger radio scheduling or a Wake-up Radio (WuR) to the on-demand method.

The paper is organized as follows. After this introduction part, the second section explained duty cycle mechanism, network topology, and ContikiOS with Coja. The third section described the results and discussion of the proposed scheduling effects on network. The last section concluded this research finding.

2 Research Method

In this study, the tests were conducted through model simulations using Cooja for ContikiOS and Skymote as a sensor node. ContikiOS as a low power operating system could provide three network stacks, i.e. IPv4, IPv6, and Rime.

ContikiOS allowed network-layer to work on top of several kinds of MAC layer protocols divided into three sub-layers; MAC, RDC, and Framers. The framer followed the 802.15.4 standard for easy reading of communication logs and used the cc2420 radio driver and the msp430 radio wake-up module. ContikiOS used the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) to detect collisions having the following mechanism:

- Checking the medium before sending.
- Cancelling delivery if there’s another sender.
- Delivery time following the schedule made by RDC layer.
- The number of retransmissions following the configuration at the network layer.

The objective of these simulation tests was to evaluate latency in WSN, following the energy efficiency investigation [18], using the on-demand duty cycle method on WSN, accomplished by employing a radio trigger or WuR scheduling mechanism.

Figure 1 illustrates that energy savings can be by achieved by various means e.g. scheduling and setting on dynamic transmission distance at the media access (MAC) layer [19], [20], or other approaches at the network, transport, or application layer, or optimizing on hardware components at the physical layer [21].
Table 1 shows the comparison of scheduling methods [18], [21]. Positive (+) and negative sign (-) described an advantage, and a shortcoming of the method, respectively. Easy to be implemented factor could determine the cost and hardware selection. For instance, the energy saving was better using T-MAC than S-MAC; and CXMAC performed much better than those two, however it was harder to be applied.

2.1 Tree Topology

The tree topology of the wireless sensor network was used in this study. Figure 2 shows that a node has two layers of coverage, transmission-range, and interference-range. The transmission-range was the maximum distance that a node could successfully transmit data, while the interference-range was the area where the sending node might interfere with the transmission of a third-party node that transmitted data. The placement of nodes must intersect on the transmission-range; thus, the communication data can be received properly.

Energy consumption at a node can be described in Table 2. WSN communication is a low power and lossy network communication, thus a node would be in an idle condition more often than sending the data. Energy savings were made by reducing the active time of radio data and MCU (\(t_M\)), and the active time of radio wake-up (\(t_W\)) by sacrificing the success and speed of the packet delivery.
receiving time by receiver, covering sending, and receiving time at radio layer, and delay time of resending due to collision.

2.2 WAORDC work principle

The proposed module fixed the shortcomings of the on-demand WaCo method. Scheduling was performed on the continuously active WRx module using the asynchronous CXMAC technique as shown in Fig. 3. Therefore, the proposed new radio-duty-cycle is displayed in a time diagram as shown in Fig. 4. Changes in receive time on additional radios were illustrated as WRx. Previously in the WaCo method the module was always active, in our proposed method, called WAORDC (Wake-up Asynchronous on-demand radio duty-cycle), it was only active at certain time intervals using an asynchronous mechanism.

<table>
<thead>
<tr>
<th>No</th>
<th>Duty cycle</th>
<th>Category</th>
<th>Latency</th>
<th>Energy</th>
<th>Easy app</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SMAC</td>
<td>Synchronous</td>
<td>-</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>2</td>
<td>TMAC</td>
<td>Synchronous</td>
<td>-</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>3</td>
<td>LPL</td>
<td>Asynchronous</td>
<td>-</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>4</td>
<td>WiseMAC</td>
<td>Asynchronous</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>CXMAC</td>
<td>Asynchronous</td>
<td>+</td>
<td>++</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>STEM-B</td>
<td>On-demand</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>7</td>
<td>STEM-T</td>
<td>On-demand</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>8</td>
<td>WACO</td>
<td>On-demand</td>
<td>++</td>
<td>++</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 3. Preamble checking system in CXMAC

A test mechanism for sending a packet was made to measure the length of radio’s active time, when the process was idle, sending beacons, transmitting and receiving data. The test was performed using unicast delivery with several nodes as senders, relays, and sinks. The test used the Rime protocol stack supporting the CSMA/CA mechanism to ensure the success of packet delivery by avoiding
collisions. With this test, energy savings were achieved at each node.

Figure 4. Preamble checking system in WAORDC

3 Research Method
3.1 Communication between nodes
As shown in Fig. 4 communication is initiated by the sensor-node. The results of measuring environmental parameters using devices on the sensor-nodes were sent to the sink-nodes at the base-station. The sensor-nodes could send data directly to the sink-node or must go through other sensors working as relays. The process occurred on the network had the following steps:

a) Initial process
   • All sensors worked as receivers in stand-by mode. Periodically, the node activated the receiver on the Wake-up Radio to check for activity on the network. The distance between checking was \( t_i \).
   • If a node sent a packet, the sender must send a preamble-tone, a simple signal containing destination information using the transceiver on the Wake-up Radio. The length of the preamble time was \( t_p \).
   • In order for the preamble-tone of the sender to be read by the receiver, the condition of \( t_i < t_p \) must be met.

b) After getting the data, sensor-node started the communication process.

c) The sensor-node acted as a sender activating the preamble-tone and extending the periodic checking time.

d) The preamble-tone activated the adjacent node based on RPL routing.

e) The node next to it received a wake-up and then checked the address.

f) The node then activated the main radio.

g) The node receiving the wake-up signal replied as a sign that the node was ready to receive data from the main radio.

h) After the sender received a confirmation, the actual data was sent via the main radio.

i) The receiver sent an ACK via the main radio.

j) After successful data transmission, the sending node returned to stand-by mode.

k) Receiver checked the address on the package. Were the packets matched, they would be forwarded as in process numbers 3 to 10.

l) If the address in the packet matched the address of the receiver, the packet was stored or forwarded to the data center for further processing.
After the communication model was made, an asynchronous duty-cycle module was created managing the Wake-up Radio scheduling.

### 3.2 Making WAORDC module

In the proposed method, improvements were made to solve the shortcomings in the On-demand method using the additional Wake-up Radio that has been done [14], [15] by adding a Wake-up Radio module that sleeps periodically with the asynchronous STEM-T sub-method. The new module was created by modifying some of the ContikiOS source codes. ContikiOS allowed developers to add new modules for each netstack layer.

a) **Powertracker for Additional Radios**

To facilitate the calculation of the duty-cycle percentage of each node, a Powertracker program was needed. Cooja only provided Powertracker for recording every event occurring on ContikiOS. The events represented the duration of a sensor turning on the radio module, sensor, or working in a sleep mode. Meanwhile, additional radios (WuR) were required in the WAORDC module, then displayed in a WuRPowertracker observation window by adding additional radio variables to the document tools/cooja/apps/wur_powertracker/java/WurPowerTracker.java.

b) **Registering a New Protocol on Netstack ContikiOS**

At first the WAORDC module should be registered in the ContikiOS protocol-stack. WAORDC worked on the RDC layer, which was between the physical layer and the media access layer. RDC must be able to work with the RADIO_cc2420 protocol, and be able to carry out commands sent by protocols from the media access layer such as CSMA. Protocol configuration registered in the netstack.h module was represented as follows.

```c
RADIO_cc2420.init();
RDC_waordc.init();
MAC_csma.init();
LLSEC_nullsec.init();
MAC_csma.init();
NETWORK_rime.init();
FRAMER_802154.init();
```

The configuration would be added to file core/net/netstack.h. RDC_waordc.init() function would call new module header i.e. waordc.h.

c) **Adding the WAORDC Library**

The WAORDC module allowed packet delivery to be initiated by activating the preamble signal i.e. by activating WTx via the set_wuR_tx() function. This function sent a preamble containing the destination address according to the instructions ordered by the CSMA module from the media access layer. The module worked according to the following pseudocode.

```c
Init(Send_packet)
    Set rdc driver
    input()
    if packetlen == acklen, ignore
    if not FRAMER.parse, error parsing
    if address not match, ignore
    detect duplicate
    read packet, forward MAC driver
    send ACK
    Send_packet(), callback(input)
```
This module had three main functions, the function to start the operating system, the input function to read the response sent by the radio module at the radio/physical layer, and the send_packet() function which was executed when the timer found the time of sending sensor data, or when the node received a packet that must be forwarded to the next sensor-node. This module was stored in a WAORDC library file at the location core/net/mac/waordc.c.

d) Adding the WAORDC Driver

A new driver was created with the addition of a sleep function for one third of the cycle-time with the following pseudocode, therefore the WAORDC module could provide instructions to additional radios using the ultralow power MCU msp430.

```
Set dc_timer cycle_time/3
Wur_init()
    Set radio driver
    Configure
        If sensor active
            Set interrupt pin Rx
            Select pin Rx
            Enable interrupt
        Select pin Rx
        Read pin Rx
            If address match, wait for td_timer
            Clear pin Rx
    Wur_set_tx(), set Tx pin
    Wur_clear_tx(), clear Tx pin
    Wur_off(), Set sensor off, wait for ti_timer
    Wur_on(), Set sensor on
```

When WRx detected a signal with an address that matched the sensor-node address, the WuR module would be active longer until the longest packet sending time (t_{d}), but if during the active time it did not get any wake-up signal, the WuR module would return to sleep during the interval (t_{i}).

e) Sensor_collect Application

To collect monitoring data results from each sensor-node, a sensor_collect.c program was created. This program ran on sensor-nodes running the ContikiOS operating system with the addition of the previous modules. The functions in this program were registered as a subprocess (thread). The sensor_collect() function would run continuously as long as the sensor-node was active, and ran the recv() function according to the following pseudocode.
AUTOSTART_PROCESSES (sensor_collect);
recv(linkaddress, originator, seqno, hops)
read packet message
if not sink_node, run sensor_collect(message)
print message
PROCESS_THREAD sensor_collect();
listen, callback(recv)
loop
if interval_timer expired
activate sensors
build message
deactivate sensors
send data
endif
endloop
ENDPROCESS_THREAD

The recv() function was executed as a callback. If the sensor-node was a sink-node, the packet would be received and then written into the log file. However, if the address was unmatched, the packet would be forwarded to the sensor-node gateway node. In this paper, we focused on evaluating the latency. The higher the latency value, the higher the risk of communication failure.

4 Discussions

The Disk Graph Radio Medium (UDGM) unit in the Cooja simulator abstracted radio transmission as a transmission range circle. The UDGM used in the simulation was UDGM Distance Loss. In the simulation, the number of nodes of 3 and 10 represented a loose sensor network, while 25 and 50 number of nodes represented a high-density mesh network condition. Transmission range and energy efficiency in sparse networks had been discussed in [22], [23].

In this study the packet-rate value was adjusted to the PAN results test, where it ran in the simulation of collecting data from the sensor node to the sink-node every 30 seconds [18].

4.1 Energy Efficiency

Energy results showed in Table 3 was following Equation (1).

\[ E = \sum_{i=1}^{n} \text{(active percentage)} \times \frac{20}{60} \times \text{(wattage module)}_i \]  

(1)

Table 3 shows the energy per-node calculated 20 minutes. The energy used by the M_{Rx} and M_{Rx} modules was examined by adding up the MR power when idle, with the power used when transmitting data or receiving data. While the energy consumption of the Wtx and Wrx modules was calculated by adding up the WuR power when idle, with the power used when sending preamble packets or when activating the probe. In the NullRDC and CXMAC methods, energy consumption was not carried out on the WuR, Wtx and Wrx modules since they only used one radio (MR) to communicate. WAORDC’s energy usage was reduced to 68.4 mWh, better than the WaCo method i.e. 90.2 mWh, from the average energy of 1890.5 mWh for 20 minutes without scheduling [18].
Table 2. Energy Requirements in WSN with a duty-cycle mechanism

<table>
<thead>
<tr>
<th>No</th>
<th>State</th>
<th>Power (W)</th>
<th>Active period</th>
<th>E (Wh)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Idle</td>
<td>PM</td>
<td>tM = (t_cycle*t_sleep_m)/t_cycle %</td>
<td>EM</td>
</tr>
<tr>
<td>2</td>
<td>Transmit data</td>
<td>PMtx</td>
<td>tMtx = (t_cycle*t_sleep_mrx)/t_cycle %</td>
<td>EMtx</td>
</tr>
<tr>
<td>3</td>
<td>Receive data</td>
<td>PMRx</td>
<td>tRx = (t_cycle*t_sleep_mrx)/t_cycle %</td>
<td>EMRx</td>
</tr>
<tr>
<td>4</td>
<td>Idle beacon</td>
<td>PW</td>
<td>tW = (t_cycle*t_sleep_w)/t_cycle %</td>
<td>EW</td>
</tr>
<tr>
<td>5</td>
<td>Sending beacon</td>
<td>PWtx</td>
<td>tWtx = (t_cycle*t_sleep_wtx)/t_cycle %</td>
<td>EWtx</td>
</tr>
<tr>
<td>6</td>
<td>Waiting beacon</td>
<td>PWrx</td>
<td>tWrx = (t_cycle*t_sleep_wrx)/t_cycle %</td>
<td>EWrx</td>
</tr>
</tbody>
</table>

Total E_total

Table 3. Details of Energy Requirements per node in 20 minutes for 3 nodes

<table>
<thead>
<tr>
<th>Module</th>
<th>Power per-module (Watt)</th>
<th>Duty-cycle (%)</th>
<th>Energy (Watt hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NullRDC</td>
<td>CXMAC</td>
<td>WaCo</td>
</tr>
<tr>
<td>MR</td>
<td>56.400</td>
<td>99.94</td>
<td>6.79</td>
</tr>
<tr>
<td>Mtx</td>
<td>52.200</td>
<td>0.02</td>
<td>0.11</td>
</tr>
<tr>
<td>Mrx</td>
<td>56.400</td>
<td>0.04</td>
<td>0.09</td>
</tr>
<tr>
<td>WuR</td>
<td>1.944</td>
<td>-</td>
<td>99.95</td>
</tr>
<tr>
<td>Wtx</td>
<td>8.380</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Wrx</td>
<td>1.440</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 5. Packet delivery latency with low node density

4.2 Latency

In latency test, the difference in delivery time was obtained by writing logs and time on the simulator on the Cooja when a node started the delivery process, then logs and time on the simulator were returned when the packet was being read by the destination node. In the first test, it was carried out on a low-density network by recording the latency of each packet sent by each node, then the average of packet delivery latency value with the same packet order. For the first 10 packets, the results
are shown in Table 4.

The CXMAC method utilized a preamble having a long lag time, when a node started to send a packet, while there was no nearby active node, the sending node would have to wait for long enough, hence, it had an impact on latency as shown in Fig. 3 and 4. The WAORDC solved this by lowering preamble-time by short pauses.

The NullRDC was used as a reference method because at low density it had the smallest latency value. Figure 5 shows the average of all latencies crossed all packet sequences with a density of 3 motes per 100x100 m^2. The WAORDC performed better than the scheduling technique using asynchronous methods such as CXMAC.

Figure 6 shows comparison of time averages for a packet being sent by a sensor node until being received by a sink node, plotting for 3 to 50 nodes. Therefore, the increasing of latency in the WAORDC was higher than the WaCo, however it was faster than the CXMAC.

Through a comparison between the NullRDC method and the other three techniques, Table 5 showed that the CXMAC and the WaCo scheduling approaches required 8.97% (or 169.6 mWh) and 4.76% (or 90.2 mWh) less energy, respectively. The WAORDC presented even less than 4%, or approximately 68.4 mWh, of the total energy required which was 1890.5 mWh, to activate the sensor node within a 20 minute-period. The CXMAC method revealed a reduction of the success rate of data transmission by 90.88%, with transmission delay time of around 4.7 seconds, while the WaCo method showed 86.08% reduction with a delay time of 2.3 seconds. The WAORDC method exhibited the lowest one, i.e. 79.2% with a latency of 3.3 seconds. These findings suggested that by combining the asynchronous and on-demand methods, as seen in the WAORDC approach, could effectively save energy consumption.
<table>
<thead>
<tr>
<th>Sequence number</th>
<th>Null RDC</th>
<th>CX MAC</th>
<th>WaCo</th>
<th>WAORDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.026</td>
<td>0.373</td>
<td>0.036</td>
<td>0.255</td>
</tr>
<tr>
<td>5</td>
<td>0.026</td>
<td>0.517</td>
<td>0.033</td>
<td>0.368</td>
</tr>
<tr>
<td>6</td>
<td>0.026</td>
<td>0.274</td>
<td>0.033</td>
<td>0.179</td>
</tr>
<tr>
<td>7</td>
<td>0.026</td>
<td>0.460</td>
<td>0.036</td>
<td>0.329</td>
</tr>
<tr>
<td>8</td>
<td>0.026</td>
<td>0.210</td>
<td>0.033</td>
<td>0.115</td>
</tr>
<tr>
<td>9</td>
<td>0.026</td>
<td>0.333</td>
<td>0.036</td>
<td>0.216</td>
</tr>
</tbody>
</table>

Table 5. Influence of Scheduling Method on Network Quality

<table>
<thead>
<tr>
<th>Method</th>
<th>Energy (mWh)</th>
<th>PDR (%)</th>
<th>Latency (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CXMAC</td>
<td>169.61</td>
<td>90.88</td>
<td>4.685</td>
</tr>
<tr>
<td>WaCo</td>
<td>90.16</td>
<td>86.08</td>
<td>2.330</td>
</tr>
<tr>
<td>WAORDC</td>
<td>68.39</td>
<td>79.20</td>
<td>3.336</td>
</tr>
</tbody>
</table>

5 Conclusion
The WAORDC method implemented the radio trigger scheduling method (WuR) to obtain energy efficiency in WSN. During the 20-minute period test, the average energy requirement of each sensor node decreased to 68.4 mWh or only approximately 4%, from the total energy of approximately 1890.5 mWh without scheduling. These results outperformed the on-demand WaCo (90.15 mWh), and the asynchronous CXMAC mechanism (169.6 mWh). Although the packet delivery ratio decreased by 79.2%, with a delay time of 3.3 seconds for this proposed WAORDC, it was relatively small compared to that of the CXMAC method.

Acknowledgments. Authors thank the Porang Research Center for the material support.

References


