Duty Cycle Based Energy Management Tool for Wireless Sensor Networks

Gustavo A. Núñez Segura, Cintia B. Margi

Abstract— **Wireless Sensor Networks are a valuable technology to support many applications, including smart cities and Internet of Things. Energy is a key resource for these devices, since they are usually battery operated. We propose a duty cycle based energy management tool that enables a better control of the duty cycle period according to the node energy availability, thus increasing the network lifetime. We present three different scenarios with three different configurations to show in what conditions the tool gets the best results. The tool was programmed for ContikiOS and the simulations were conducted with Cooja Simulator.**

Keywords— **wireless sensor networks; duty cycle; energy management.**

I. INTRODUCTION

Wireless Sensor Networks (WSN) have been used to support several different applications [5], such as detection and tracking, environmental and industrial monitoring, health monitoring and support. Furthermore, WSN plays an important role in the Internet of Things (IoT) as well.

Nodes in a WSN are typically battery-powered and resource constrained (in terms of memory, processing and communication), and they take part in a multihop ad hoc network [5]. Communication patterns in a WSN include node-to-sink, sink-to-node and node-to-node. The most common is related to the goal of a WSN to transmit sensed information from the environment to a central base station [6].

Each node in a WSN runs four main tasks: sensing, processing, receiving and transmitting information. The limited energy supply is a key challenge faced by WSNs [4]. Due to the lack of a vast energy source and, in several cases, the difficulties that carry on changing all the batteries in a WSN, the reduction of energy consumption is a major challenge [10].

From the tasks mentioned before, the energy consumption can be grouped into three categories: sensing, processing and communication. From these three, over the 80% of the consumption is from the wireless communication module [4]. The consumption of the wireless communication module is distributed in transmission, reception, idle and sleep. The largest energy dissipation is on reception state and the smallest is on sleep. Reception and idle (listening for new data) consumption are quite equal [19].

TelosB mote energy consumption measurement shows that the CPU consumption for an instruction at 4 MHz is 2,33 mA, while the Transmitter (TX) + CPU at 4 MHz and 0 dBm drains 21,4 mA. That is a difference of over 8 times and it can increase when the mote turn into a low power mode or decrease the clock frequency [15]. Notice that these numbers alone reinforce the need to make a smart use of communication and the radio module in order to increase the network lifetime.

WSN characteristics impose several constraints and requirements on its protocols, operating systems and programs running on the devices. One of the main approaches to decrease the energy consumption is to make use of duty cycles, thus alternating periods of activity (processing, sensing and communication) and sleep. The IEEE 802.15.4 [20] standard is widely used as the Medium Access Layer for Low Power and Lossy Networks (LLN), and it includes provision for radio duty cycling in its specification.

Both TinyOS [21] and Contiki [2], two well known operating systems for WSN nodes, provide mechanisms to implement radio duty cycling on top of IEEE 802.15.4. TinyOS has a Low Power Listening mechanism implemented, while Contiki used X-MAC at first and ContikiMAC nowadays. ContikiMAC radio duty cycle (RDC) protocol [1] uses a wake up routine with a very low duty cycle, keeping the radio off as much as possible. For both cases, duty cycle period is static and predefined before deployment.

We propose a duty cycle based energy management tool that enables a better control of the duty cycle period according to the node energy availability. Our hypothesis is that a dynamic duty cycle based on the amount of energy available on the node could increase the network lifetime.

In order to evaluate the proposed duty cycle based energy management tool, we selected Contiki [2] as the operating system given its wide use in WSN and IoT platforms. The tool is built upon ContikiMAC RDC [1], adding a variable control signal with a lower frequency than the original ContikiMAC RDC. For the application, the duty cycle depends on the battery level, which is configured in the code as a parameter. The tool uses the Energest [3] library to monitor the battery level.

The remainder of this paper is organized as follows: Section II reviews duty cycle related work. Section III presents the energy management algorithm proposed for the tool. The evaluation of the algorithm is discussed in Section IV. Finally, conclusions and future work are presented in Section V.

II. RELATED WORK

Different approaches have been explored to reduce the energy consumption in WSNs and duty cycle-based algorithms for energy management can be pointed as one of them. Duty cycle for energy management has different focuses, but most

G.A.N. Segura, C.B. Margi¸ Department of Computer Engineering and Digital Systems, Universidade de São Paulo, São Paulo, Brazil, E-mails: gnunez@larc.usp.br, cintia@usp.br.

of them are implemented in order to minimize the time spent on radio-on state of each node from different characteristics or behaviors of the node.

Zhang *et al.* [18] derive a relation between the duty cycle and the distance of the node with the sink to decrease power consumption. The paper is supported by the fact of convergecast communication in WSN, where the packet traffic in nodes near the sink is higher. Authors present experiments with results of packet delivery ratio, energy consumption and average latency for two different receiver protocols with duty cycle adaptation. The experiment was simulated with Opnet.

A similar approach is proposed by Youssef *et al.* [17], where it is presented a spatial configuration for the radio duty cycle. The duty cycle can be updated by two different mechanisms: the first one is by a broadcast sent by the sink, which updates the threshold of all nodes; and in the second one the threshold is updated as a function of the battery capacity level. To evaluate the results, authors employed ContikiOS, Cooja Simulator and MSPSIM emulator.

Lopez *et al.* [12] introduce a duty cycle to enable the radio for short time and after that turn the node into an idle state. Authors also propose a technique to minimize electromagnetic pollution. The experiment was performed using SunSPOT sensors.

The work presented by Qui *et al.* [16] focused on increasing the lifetime in a WSN for underground pipeline. Authors present an algorithm to adjust the duty cycle according to the real time wireless channel condition.

Pereira *et al.* [14] present an algorithm for deciding which tasks to run based on the battery energy level. To do this, the user can define states and the threshold for them. When the mote boots with full energy capacity, it begins in state zero and can perform any task, but when the battery level starts to decrease, the mote change states, limiting the quantity of tasks that it can run. Their implementation was designed for TinyOS.

We followed the tool approach [14] adding the dynamic duty cycle scheme. Unlike previous work, we present different communication scenarios to show how the duty cycle based tool behaviour depends on the node role (client, router, server).

III. The Proposed Tool

As mentioned before, ContikiMAC operation wakes up the node periodically to check if a neighbor has sent a message. If that is the case, the radio remains awake to receive it, and when it is done, the node goes back to the periodically wake up cycle. Figure 1 presents how ContikiMAC works. This operation actually has a good power efficiency, but some applications need as much lifetime as possible.

A. The algorithm

The algorithm proposed controls the ContikiMAC operation with a dynamic duty cycle low frequency signal (control signal). This signal can be considered as a flag with two possible states: high and low. Thus, when the control signal is high, the ContikiMAC operates normally, and when the control signal is low, the node turns off the radio and disables the

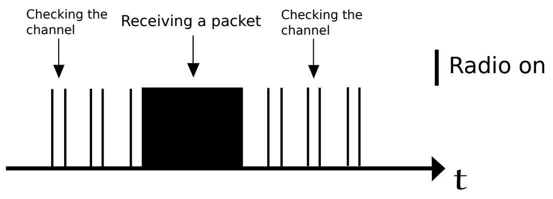


Fig. 1: ContikiMAC normal operation

ContikiMAC control. The node remains in that state until the flag turns high again (Figure 2).

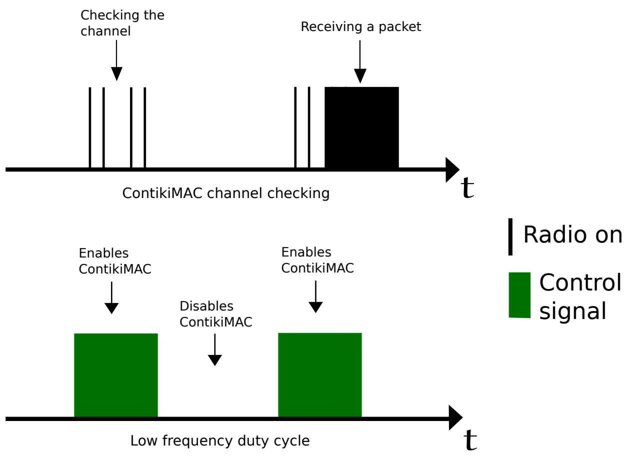


Fig. 2: ContikiMAC working together with the tool proposed

To change the duty cycle, the tool defines n states and each state has a threshold, and when the battery level reaches this threshold, the tool moves to the next state and the duty cycle changes too. To avoid problems while the message is being received, the control signal is disabled when the reception starts, and it is enabled again when the node turns idle again.

With this control, the energy consumption when in idle will be lower than working just with the ContikiMAC control. This tool allows to save the energy spent by the node when it awakes to check activity in the channel. Knowing the behavior of the network, the duty cycle of the tool could be adjusted to avoid messages losses and save energy.

B. Contiki implementation

The tool was developed in ContikiOS given all the advantages in energy consumption that it provides. Figure 3 illustrates the flowchart defined for this implementation.

First, the program sets the parameters. In this part, the user can define how many states he wants, the threshold of each one, the fully charged battery energy amount, and the current state timers dependency. Then the program initializes four timers. Two of them are to set control signal frequency and duty cycle, the third one sets the frequency for the *CalculateConsumption* function, and the fourth one sets the frequency for the *CalculateState* function. These functions are explained next:

- *CalculateConsumption*: this function uses the Contiki Energest library to measure the time spent in four pro-

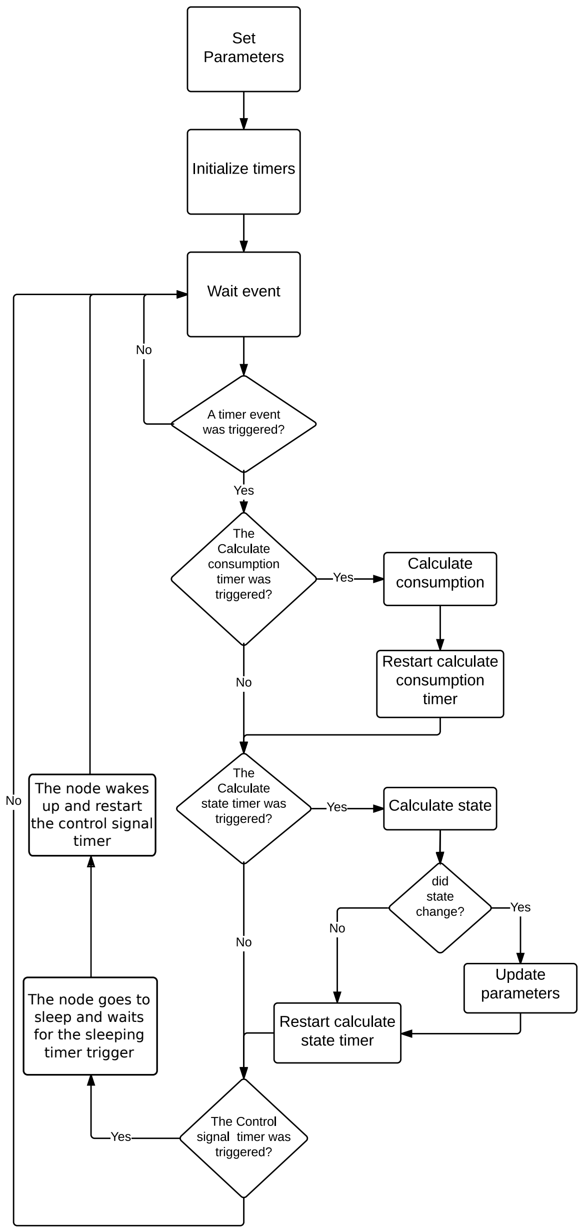

Set Parameters
Initialize timers
Wait event
No
A timer event was triggered?
Yes
The Calculate consumption timer was triggered?
Yes
Calculate consumption
Restart calculate consumption timer
No
The Calculate state timer was triggered?
Yes
Calculate state
did state change?
Yes
No
Update parameters
Restart calculate state timer
No
The node wakes up and restart the control signal timer
No
The node goes to sleep and waits for the sleeping timer trigger
The Control signal timer was triggered?
Yes

Fig. 3: Tool algorithm flowchart

cesses: CPU, Low power mode (LPM), listening and transmitting. Then, the function calculates the equivalent energy for those intervals and substracts the total from the battery level. It is important to previously know the current consumption of each task, since Energest returns only the time the task was running. The formula used to calculate the consumption is shown in equation (1), where E is the energy consumption, t_{task} is the time the task spend working, V_s is the voltage from the source, and I_{task} is the current consumption for the task.

$$E = t_{task} * V_s * I_{task} \tag{1}$$

- *CalculateState*: this function checks the battery level and compares it with the next state threshold. If the current battery level is lower than the next state threshold, the state changes to the next one. When the state changes, the timers period changes too. This dependency may be set by the user.
- *ControlSignal*: this function checks whether the mote has a task in the queue, if so, the mote waits until all the tasks are finished. When the mote is finally idle, the function deactivate the ContikiMAC and sends the mote to sleep. The checking frequency and the sleeping period depends on the timers explained before.

IV. Simulation

The objective of this analysis is to measure the tool performance in different scenarios. The performance is based on the energy consumption reduction when the code has the tool working on it. The experiment was performed with Cooja Simulator [8], a network simulator/node emulator part of in ContikiOS.

The experiment has three different scenarios and three different configurations for each one. Using Figure 4 as reference, the three scenarios are described below.

- First scenario: Node number 2 is broadcasting a message every 60 seconds, and nodes 1 and 3 are listening and receiving it. Besides the message, node 1 sends his MAC address to inform the other nodes who is broadcasting. The simulation employed the rime stack of ContikiOS for a lightweight communication. It is important to mention that ContikiMAC does not wait for a confirmation message when broadcasting. To ensure the nodes receive the message, the server sends the message several times during the wake-up interval [1].
- Second scenario: Node 1 is a client, node 2 is a router and node 3 is a server. The client sends a message every 60 seconds, the router receives the message and then forwards it to the server.
- Third scenario: As the second scenario, node 1 is a client, node 2 is a router and node 3 is server. The client sends a message every 60 seconds, the router receives the message and sends it to the server. In this arrangement the communication uses the Routing Protocol for LLN (RPL - RFC 6550).

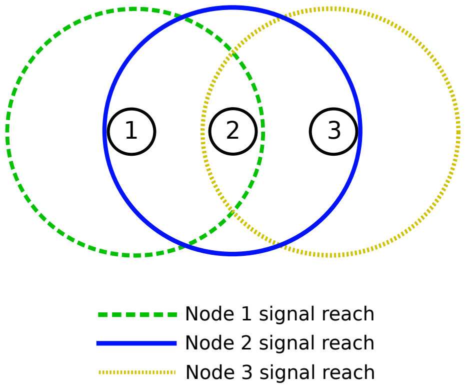
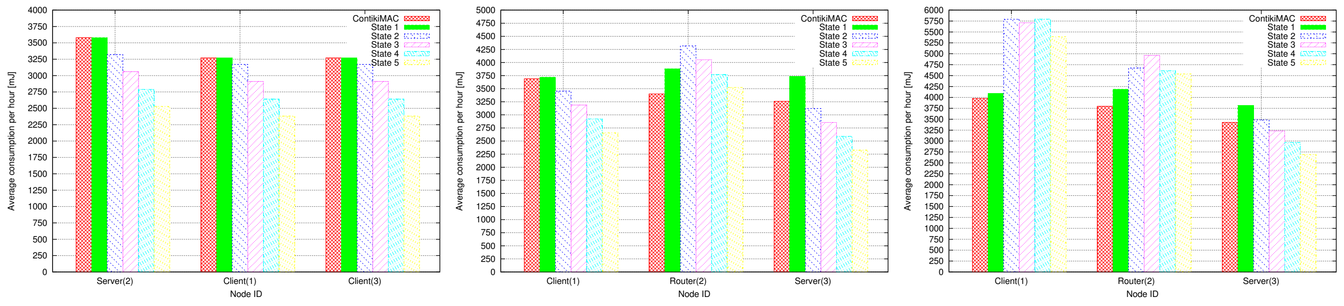


Fig. 4: Nodes configuration for simulations

To evaluate the tool performance, all scenarios were tested in three different configurations: first, with the ContikiMAC



(a) Energy consumption for Scenario 1 with ContikiMAC and the tool proposed (b) Energy consumption for Scenario 2 with ContikiMAC and the tool proposed (c) Energy consumption for Scenario 3 with ContikiMAC and the tool proposed

Fig. 5: Scenarios

control off, second running with the ContikiMAC control, finally with the tool and ContikiMAC control running at the same time in all nodes.

In the last configuration, the tool was tested for five states with different duty cycles, without changing the messages transmission rate. The tool control signal starts with a 100% duty cycle and it decreases 20% for each next states. For example, for the state number 2 the duty cycle will be 80%, for the number 3 will be 60% and so on. For the tests the program uses the TelosB mote consumption model [15] [13] [7]. Table I shows a summary of all tests and their configuration.

TABLE I: Summary of simulation tests

Scenario	MAC configuration	Routing	Duty cycle
Scenario 1	Without ContikiMAC	Static	-
	ContikiMAC		0,3%
	Tool and ContikiMAC		Dynamic
Scenario 2	Without ContikiMAC	Static	-
	ContikiMAC		0,3%
	Tool and ContikiMAC		Dynamic
Scenario 3	Without ContikiMAC	RPL	-
	ContikiMAC		0,3%
	Tool and ContikiMAC		Dynamic

The metric used to compare the performance is the average energy consumption per hour in millijoules. To obtain the results the tool measures the energy remaining in the battery of the node 10 times per hour during 10 hours (experiment duration). The process was repeated in all scenarios. The data collected during this time allows to get a good approximation of the node typical consumption, and with the average energy consumption per hour it is possible to see how the tool affects the consumption of each node.

A. *Results*

In the first experiment, scenario 1, 2 and 3 were running for the first two configurations, to compare the energy consumption performance when the node has not a system to manage energy consumption, and when the node uses the ContikiMAC control. Table II shows the average consumption per hour for each node for all the scenarios described before. The consumption in all nodes when ContikiMAC is working is under 2% compared with the configuration without a MAC control.

TABLE II: Energy consumption for first and second configuration

Node	Configuration 1 [mJ]	Configuration 2 [mJ]
Scenario 1		
1	237870	3270
2	237910	3580
3	237870	3270
Scenario 2		
1	237860	3690
2	237860	3400
3	237860	3260
Scenario 3		
1	237998,2	3978,8
2	238029,2	3797,8
3	237981,4	3428

Figures 5a, 5b and 5c show the average energy consumption per hour for all the scenarios when nodes are working just with the ContikiMAC control, and when they have the ContikiMAC and the tool proposed working together. In Figure 5a it is possible to see that the tool helps to reduce the energy consumption in all nodes. Working in the State 5, node 2 saves 29,32% of energy per hour, and node 1 and 3 saves 27,21% The packet delivery ratio (PDR) was not affected by the tool implementation because all nodes were running with the same duty cycle on each test. That means that the sleeping routines coincided and when the server sends the message, the clients were always awake.

Figure 5b shows that node 1 and 3 save energy when they reach State 2, and for State 5, both nodes reduce the consumption in 28,54% and 28,65% respectively. For node 2 (router) the consumption is always higher when using the tool than the consumption when using just ContikiMAC. Analizing the time measured for transmitting and listening for the router node when using ContikiMAC and when using the tool in State 2 (state with maximum consumption), the average transmitting time per hour increases from 910 ms to 6940 ms, and listening time increases from 21090 ms to 26060 ms. On the other hand, for the other two nodes, listening time decreases from 22690 ms to 18590 ms per hour for the client, and from 20170 ms to 18060 ms for the server. Just like in the Scenario 1, the PDR was not affected by the tool implementation.

Finally, the results for Scenario 3 are shown in Figure 5c. For this scenario the tool can not reduce the energy consumption for the client and router. For the server it is possible to reduce the consumption but until the node reaches State 3. For this scenario the average transmitting time per hour increases from 6030 ms to 31990 ms, and listening time increases from 21090 ms to 26060 ms for the client when the tool runs over State 3 (state with maximum consumption for this node). For the router the average transmitting time per hour increases from 3660 ms to 8620 ms, and listening time increases from 21920 ms to 24500 ms when the tool runs over the State 2 (state with maximum consumption for this node). Moreover, for the server, listening time decreases from 20630 ms to 666,75 ms per hour. The main difference between this scenario and the two others is that the routing protocol for the third one is dynamic (RPL) and this implies different control messages among the nodes [9]. Due to this, sleeping times do not coincided, causing packet losses and forcing ContikiMAC to do retransmissions, which increase the transmitting time. Finally, in the server the transmitting time has not changed over all states and the PDR is irregular and not over 0,18 for states 2, 3, 4 and 5.

V. Conclusions

We have proposed a duty cycle based energy management tool. The objective of this tool is to reduce the time the radio module remains on, by adding a low frequency control signal that changes his duty cycle depending on the battery energy level.

The results after measuring the energy consumption in three scenarios with three different configurations show that it is possible to save energy while using the ContikiMAC and the tool proposed by adding a duty cycle for the radio module. On the other hand, this combination can drive an increase in energy consumption by increasing the time the node remains transmitting a message. The cause of this problem is that nodes are sending out messages when the receiver is sleeping because of the lack of information of their neighbors' sleeping routines.

As future work, we intend to increase the number of nodes and to use more complex scenarios. Also we consider evaluating a mechanism to share the sleeping routines among the nodes, in order to decrease the over emitting time and improve the tool performance.

Acknowledgments

This work is partially funded by FAPESP research grant #2013/15417-4. C. B. Margi is supported by CNPq research fellowship #307304/2015-9. G.A.N. Segura is supported by a scholarship from CAPES-PROEX #0212083, and by the University of Costa Rica.

References

[1] Dunkels, A. 2011. The ContikiMAC Radio Duty Cycling Protocol - Core. SICS Technical Report T2011:13.

[2] Dunkels, A., Grönvall, B., Voigt, T. (2004, November). Contiki-a lightweight and flexible operating system for tiny networked sensors. In Local Computer Networks, 2004. 29th Annual IEEE International Conference on (pp. 455-462). IEEE.

[3] Dunkels, A., Osterlind, F., Tsiftes, N., He, Z. (2007, June). Software-based on-line energy estimation for sensor nodes. In Proceedings of the 4th workshop on Embedded networked sensors (pp. 28-32). ACM.

[4] Chen, F., Guo, L., Chen, C. (2012). A Survey on Energy Management in the Wireless Sensor Networks. IERI Procedia, 3, 60-66.

[5] David Culler, Deborah Estrin, and Mani Srivastava. 2004. Guest Editors' Introduction: Overview of Sensor Networks. Computer 37, 8 (August 2004), 41-49.

[6] de Almeida Oliveira, T., Godoy, E. P. ZigBee Wireless Dynamic Sensor Networks: Feasibility Analysis and Implementation Guide.

[7] Datasheet, T. Available Online: `http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf` (accessed on 5 april 2016).

[8] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. Proceedings. 2006 31st IEEE Conference on Local Computer Networks, Tampa, FL, 2006, pp. 641-648. doi: 10.1109/LCN.2006.322172

[9] Gaddour, O., Koubâa, A. (2012). RPL in a nutshell: A survey. Computer Networks, 56(14), 3163-3178.

[10] Islam, K. (2010). Energy aware techniques for certain problems in Wireless Sensor Networks (Doctoral dissertation, Queen's University).

[11] Kovatsch, M., Duquennoy, S., Dunkels, A. (2011, October). A low-power CoAP for Contiki. In Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on (pp. 855-860). IEEE.

[12] Lopez, M., Sabater, J. M., Daemitabalvandani, M., Gomez, J. M., Carmona, M., Herms, A. (2013, July). Software management of power consumption in WSN based on duty cycle algorithms. In EUROCON, 2013 IEEE (pp. 399-406). IEEE.

[13] Margi, C. B., De Oliveira, B. T., De Sousa, G. T., Simplicio Jr, M. A., Barreto, P. S., Carvalho, T. C., Gold, R. (2010, August). Impact of operating systems on wireless sensor networks (security) applications and testbeds. In Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on (pp. 1-6). IEEE.

[14] Pereira, A. H., Margi, C. B. (2012, November). Energy management for wireless sensor networks. In Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (pp. 329-330). ACM.

[15] Prayati, A., Antonopoulos, C., Stoyanova, T., Koulamas, C., Papadopoulos, G. (2010). A modeling approach on the TelosB WSN platform power consumption. Journal of Systems and Software, 83(8), 1355-1363.

[16] Qiu, L., I-Kai, K,. Salcic, Z. (December 2015). Dynamic Duty Cycle-Based Wireless Sensor Network for Underground Pipeline Monitoring. In Conference on Sensing Tecnhology, 2015 (pp 116-121). IEEE.

[17] Youssef, M. F., Elsayed, K. M., Zahran, A. H. (2014, March). Adaptive radio duty cycling in ContikiMAC: Proposal and analysis. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 491-495). IEEE.

[18] Zhang, Y., Feng, C. H., Demirkol, I., Heinzelman, W. B. (2010, December). Energy-efficient duty cycle assignment for receiver-based convergecast in wireless sensor networks. In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE (pp. 1-5). IEEE.

[19] Zhe, J., Shunjie, X., Guoqiang, W., Li, H. (2009, September). Cluster reconstruction strategy based on energy prediction mechanism in wireless sensor networks. In Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on (pp. 1-4). IEEE.

[20] 802.15.4-2011 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). 2011.

[21] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K. (2000, November). System architecture directions for networked sensors. In ACM SIGOPS operating systems review (Vol. 34, No. 5, pp. 93-104). ACM.