HAL
open science

Experiments with ODYSSE: Opportunistic Duty cYcle based routing for wirelesS Sensor nEtworks

Ichrak Amdouni, Cedric Adjih, Nadjib Aitsaadi, Paul Mühlethaler

To cite this version:

Ichrak Amdouni, Cedric Adjih, Nadjib Aitsaadi, Paul Mühlethaler. Experiments with ODYSSE: Opportunistic Duty cYcle based routing for wirelesS Sensor nEtworks. IEEE LCN 2016 - 41st IEEE Conference on Local Computer Networks, Nov 2016, Dubai, United Arab Emirates. 10.1109/LCN.2016.50 . hal-01407525

HAL Id: hal-01407525

https://inria.hal.science/hal-01407525v1

Submitted on 2 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Experiments with ODYSSE: Opportunistic Duty cYcle based routing for wirelesS Sensor nEtworks

Ichrak Amdouni*, Cedric Adjih†, Nadjib AitSaadi*, Paul Muhlethaler‡
* University Paris-Est, LiSSi, UPEC, Vitry sur Seine, France, Email: firstname.name@u-pec.fr
† INRIA Saclay, France, Email: cedric.adjih@inria.fr
‡ INRIA Paris, France, Email: paul.muhlethaler@inria.fr

***Abstract*—In this paper, we propose, design and experiment an energy efficient protocol for Wireless Sensor Networks (WSNs) named Opportunistic Duty cYcle based routing protocol for wirelesS Sensor nEtworks (ODYSSE). The main key innovation of ODYSSE is that it judiciously makes use of three mechanisms. The first one is *duty cycling* which consists in randomly switching on/off transceivers to save energy. The second one is *opportunistic routing* in which the next hop is not rigidly fixed: any node closer to the destination might become a relay. The third one, is *source coding* using LDPC, Low-Density Parity-Check codes. With asynchronous duty cycling as a starting point, the above techniques fit perfectly, yielding a robust low complexity protocol for highly constrained nodes. ODYSSE is implemented and installed in an experimental testbed composed of 45 Arduino nodes communicating with IEEE 802.15.4 (XBee) modules deployed in the large-scale platform FIT IoT-LAB. Results show that the performance obtained is very satisfying in both following scenarios: high load (images) and light load (reporting of infrequent event).**

Keywords: WSN, opportunistic routing, low duty-cycle, experimentations, Arduino, LDPC.

I. Introduction

In this paper, we focus on a distributed and adaptive routing protocol for WSNs, which can address efficient transmission of both low and large volumes of data, in particular multimedia traffic as in Wireless Multimedia Sensor Networks (WMSNs). The main objective is to efficiently manage the energy consumption while minimizing the end-to-end delay of packets from the source to the sink. To this end, we propose a routing protocol denoted by ODYSSE. It is based on opportunistic routing and low duty cycle of transceivers. The main key innovation of ODYSSE is that it combines opportunistic routing, duty cycle and source coding. The first two features are explored in this paper while the third one is included in [13] with more experimentations.

Unlike deterministic routing which builds specific routes for each source-destination pair, opportunistic routing [4][6] can instantly ensure route diversity to balance load by adaptively selecting forwarders at each intermediate hop in the path. A side effect is that energy consumption may be spread more evenly over multiple paths and intermediate forwarders with low battery levels can be avoided. Moreover, latency is minimized and fault tolerance is maximized. The main critical question in opportunistic routing is which neighboring node will forward a given packet. Solutions vary from selecting the first awake node, as in EFFORT [3], or conducting a more or less complex selection process like RI-MAC [5] and ORW [2]. For instance, ORW protocol selects the potential forwarders according to the expected number of wakeups until a packet has reached its destination (EDC metric). However, to update this metric, the exchange of duty cycle information is required. In [13], we have proved via a mathematical model that, under some conditions, selecting the first available forwarder is a delay efficient strategy.

Combined with opportunistic routing, the duty cycle mechanism has proved to be energy efficient as it minimizes the energy wasting states like idle listening, collisions and traffic overhead. For instance, ASSORT [4] and B-MAC [6] are sender-initiated: sender nodes announce they have data to send either by sending beacons in ASSORT, or a "wakeup signal", called a preamble which lasts longer than the receiver sleep interval in B-MAC. In contrary, RI-MAC [5] is a receiver-initiated protocol. In this protocol, when any node turns on its radio, it transmits a short beacon frame to announce that it is ready to receive data.

To improve wireless transmission reliability, conventional approaches such as full data replication or on-demand retransmission are too expensive or even not possible due to very strict energy constraints and asymmetric wireless channels. Indeed, either some feedback is implemented (which is still complex in duty cycling opportunistic routing), or when operating in open loop, a good design choice is forward error-correction methods. To reduce the complexity, ODYSSE adopts the last open-loop approach based on LDPC coding [7].

ODYSSE is responsible for routing data from data sources to the gateway while **optimizing the lifetime of the network**. The duty cycle of routers is adapted to fit 1) bulk data transfer, and 2) infrequent data reporting. ODYSSE aims at **keeping low data transmission delays and overhead** since it deals with small capacity devices. To the best of our knowledge, our work is the first to make experiments of three features: i) opportunistic routing, ii) duty-cycle and iii) coding for a multimedia application. Experimentation is done with a testbed of 45 based Arduino nodes deployed in the large-scale FIT IoT-LAB platform [8][1].

The rest of this paper is organized as follows. The next section describes the ODYSSE protocol. Afterwards, Section III details the large testbed and the experimental results obtained. Finally, Section IV concludes the paper.

[1]https://www.iot-lab.info/

II. ODYSSE Protocol

A. Overview of the ODYSSE Protocol

To save energy, ODYSSE protocol makes **router nodes duty cycled**. Routers are put in the sleep state asynchronously and randomly. This design avoids the synchronization overhead and the degradation of the system performance in case of clock drift. As a consequence, the wireless topology is dynamic and unpredictable. Thus, a good alternative for unsuitable classical routing is the **opportunistic routing**.

To keep low complexity of forwarder search method, in ODYSSE, the forwarder is the neighbor offering a best compromise in terms of: **i) residual energy, and ii) relative distance to the gateway and iii) quality of wireless links.** To ensure routing progress towards the gateway and to avoid routing loops at each hop, a DAG (Directed Acyclic Graph) rooted at the gateway is implicitly constructed and maintained by each node as its **'distance'** relative to this gateway. This distance is essentially the **number of hops** separating the node from the gateway but taking into account the **link quality** as a metric. The source node generates, in addition to the original data packets, **repair (redundant) packets** which are linear combinations of original ones. This data redundancy enables the final destination to decode the received packets in order to retrieve the lost packets. ODYSSE is implemented and experimented on a **real testbed of Arduino** devices.

B. Distance Computation

Due to space limitations, we briefly describe how distances are computed, details are in [13]. As indicated above, a distance is essentially computed as a hop count. Our objectives are 1) to avoid pure *shortest path* routing, which would limit the width of the DAG (hence the alternate routes) and 2) to optionally avoid spurious retransmissions in the spirit of minimizing energy consumption. Hence, longest links are eliminated. Our heuristic for characterizing long links (in terms of distance) is based on **Received Signal Strength Indicator (RSSI)**. Indeed, if the RSSI of a received packet is above a predefined threshold, the link metric is considered to be 1; otherwise it is $1 + \gamma$ (where γ is a constant > 0).

After these steps, nodes form a logical tree (actually DAG) defining their distances towards the gateway taking into account the RSSI values.

C. Forwarders Search and Selection

Our proposal is based on a **greedy approach**. In ODYSSE, forwarder selection is **sender-initiated**. When one node u has a data packet to transmit (either the source node or any router node), it broadcasts `Beacon` messages and then, awake neighbors answer by sending `Reply` messages.

More specifically, the `Beacon` message from u includes its distance to the gateway (named `gateway-distance`). Any awake node v receiving this beacon will proceed as follows:

1) If v is closer to the gateway than u and `RSSI` < `RSSI_THRESHOLD` then v sends a `Reply` message.
2) Otherwise, the node v stays silent.

The initiator u will repeatedly send `Beacon` messages until the forwarder search phase expires:

- After a predefined period of time `BEACON_PERIOD`,
- Or, when a predefined number of `Reply`, `MAX_NB_REPLY`, messages is received.

When the node u receives a `Reply` message from any node v, it has access to the following parameters:

1) The residual energy of v.
2) The distance of v relative to the gateway.
3) The RSSI of the `Reply` message received from node v.

Depending on the **policy** (e.g., emphasis on energy, or lower latency, ...), u will select the most appropriate forwarder. Focusing on latency and motivated by the result in [13], the heuristic that we adopt in practice is to select the first available forwarder having an acceptable metric based on the distance of v coupled with the RSSI. Indeed, in [13], we have modelled the average waiting time of forwarders assuming exponential and uniform wake-up distributions. Results show that a good heuristic for latency minimization consists in selecting the first available forwarder.

Once the forwarder is selected, the node u transmits its packet. In ODYSSE, whenever no forwarder is found after `BEACON_PERIOD`, this period is extended, unlike ASSORT [4] for instance where nodes would return to the sleep state. Our approach tends to minimize delays. We believe that it is more efficient for dense networks.

The advantages of this design are the low storage capacity required and a light-weight selection procedure. Furthermore, ODYSSE, unlike for instance ORW [2] is loop free and guarantees packet unicity in the network. The design of the forwarder selection allows for a large spectrum of forwarding policies.

D. Duty Cycle

1) Principles: In ODYSSE, source nodes and gateway are always active while routers are duty cycled. The maximum duration of the active mode is fixed, while the sleep period follows a random uniform distribution. Ultimately, the duty cycling depends on the network traffic and indirectly on the density of the network. The detailed functioning of a router node, starting from the active state is as follows:

- If the node is idle (has no data packet to send), it waits for `Beacon` messages:
  - If after an active mode duration of `ACTIVE_PERIOD` no `Beacon` is received, it returns to the sleep mode.
  - Otherwise, if the node actually replied to a `Beacon`, it waits for a `Data` message, and:
  - Then if the node does not receive a `Data` message after `WAIT_DATA_PERIOD`, it returns to the sleep state,
  - Otherwise, it forwards the `Data` message as follows:
- If the node has data to send: it sends periodically `Beacon` messages, collect replies as described in section II-C, until a forwarder has been successfully selected and then the `Data` packet has been successfully sent as unicast (with use of MAC layer acknowledgements). After that, the node has no data packet to send and returns to sleep mode.

Remark 1: Note that routers are not necessarily active for the whole `ACTIVE_PERIOD`. Indeed, a router having forwarded a packet within this period, returns to the sleep state.

2) Sleep Duration: The duty cycle of nodes is critical because it directly impacts data transmissions, routing overhead and energy efficiency. In ODYSSE, nodes are unsynchronized, they randomly select a random sleep period in an interval:

$$[\texttt{MIN_SLEEP_PERIOD}, \alpha \times \texttt{ACTIVE_PERIOD}] \quad (1)$$

Where the ACTIVE_PERIOD is the maximum active period of routers, and α is a constant.

However, ODYSSE design carefully tunes this expression taking into account two application scenarios: infrequent events monitoring and multimedia application.

• **Infrequent events**: This classical scenario is characterized by low volumes of data generated by the source either periodic, regular, rare or exceptional. With this assumption, router nodes are allowed to keep their initial duty cycle by applying formula (1). This is because in this scenario, a current packet is rarely a predictor for an additional incoming packet. This algorithm will be denoted **INFR**.

• **Multimedia application**: This scenario considers a bulk data transfer of still images from the source. For this scenario, we define two algorithms:

1) **MED_ADAP**: (MultimEDia ADAPtive) Router nodes adapt their duty cycle by shortening their duty cycle. Consequently, the applied algorithm is as follows. When any router node transmits a packet, instead of turning back to the sleep state with a period given by formula (1), it rather sleeps for the minimum period MIN_SLEEP_PERIOD for a predefined number of times SHORT_SLEEP_COUNT. That is, the node sleeps for the minimum possible period of time between each retransmission – until it reaches a fixed limit. After that, it returns to the normal behavior. With this adaptive strategy, the network will face the congestion effect resulting from the bulk transfer.
2) **MED_N_ADAP**: (MultimEDia Non ADAPtive) In this case, nodes do not adapt their duty cycle, they just apply formula (1).

E. Packet Erasure Codes in ODYSSE

To combat wireless transmissions losses, ODYSSE integrates Erasure Codes (EC) [9], which are based on **data redundancy**. Indeed, instead of sending k **original** packets, the **encoder** adds m **redundant** packets. When the number of the received packets is sufficiently high, the **decoder** decodes them and retrieves the lost packets. Different coding methods exist. In our work, we use the LDPC ("Low-density parity-check") codes [10]. LDPC code is a block code defined by a sparse parity-check matrix (that is, the majority of entries are zero) and is known to provide excellent decoding performances [10]. LDPC is implemented in ODYSSE in a way that fits the extremely limited memory of the devices. Because of space limitations, we do not include results about this code in this paper.

III. Testbed Deployment and Experimentations

A. Testbed Hardware

45 Arduino nodes are deployed in an area of 65×10 meters. Each node has two components: the ATmega2560, a 8-bit microcontroller (256 KB of flash memory and 8 KB of SRAM) and the radio module XBee (IEEE 802.15.4). The transmission power is set to 2 dBm. Source nodes are in addition equipped with TTL serial JPEG camera. The testbed is integrated in the existing large testbed FIT IoT-LAB, as described in [11][13].

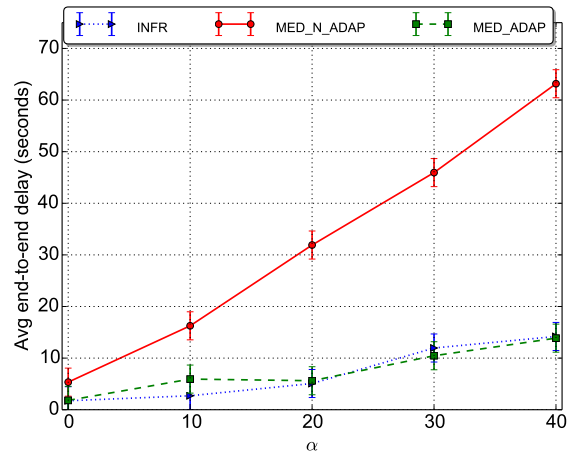

Fig. 1. Average end to end delays per data packet.

B. Experiment Settings

The source node generates one photo every 30 seconds. We vary the value of α ($\alpha = 0$: no duty cycle is applied). For the INFR scenario, the source generates data packets with a random inter-packet delay ranging from 5 to 10 seconds. The other experiment parameters are summarized in TABLE I. Results presented in this section are the average of 2 to 8 images.

TABLE I
EXPERIMENT PARAMETERS

ACTIVE_PERIOD	0.2 s	WAIT_DATA_RPERIOD	3 s
MIN_SLEEP_PERIOD	0.05 s	RSSI_THRESHOLD	−83 dBm
SHORT_SLEEP_COUNT	3	BEACON_PERIOD	3 s
MAX_NB_REPLY	1	m	30

Our result analysis originates from detailed logs collected through the serial ports of Arduino devices. Collected logs represent a volume of 26 GBytes.

C. Average End to End Delays

Fig. 1 depicts the average end-to-end delay per data packet for each image as a function of α.

As depicted in Fig. 1, the end-to-end delays vary linearly with α. The scenarios MED_ADAP and INFR have low end-to-end delays that slightly increase with α. Hence, in these scenarios there is a good trade-off between the energy saving and the delay reduction. In MED_N_ADAP, we see that the delays are rapidly increasing with α. This is because routers are less available due to the duty cycle and the network may reach congestion states because of the bulk transfer. These facts are alleviated by the duty cycle adaptation (in MED_ADAP) and large packet interarrivals (in INFR). While packet interarrivals are dependent on the application (low in

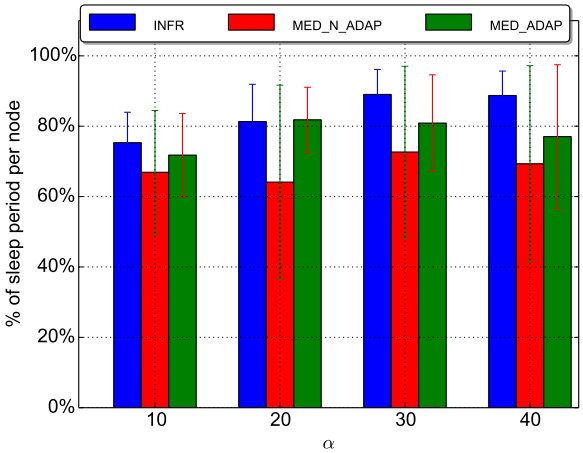

Fig. 2. Average duty cycle per node.

multimedia scenarios in particular), the duty cycle adaptation is an effective method to reduce delays. Also, as we will see hereafter, this adaptability has a smaller impact on the energy consumption. Delays are due to packet losses and to forwarders search time.

D. Duty Cycle

Fig. 2 depicts the average duty cycle per node, defined as the percentage of the average sleep time of any node during its lifetime. Notice that when router nodes are in the active state, they are either (1) waiting for beacons, (2) searching forwarders, (3) or waiting for data packets after having transmitted a `Reply` message. All these states have a maximum duration after which nodes return to the sleep state, except after a forwarder search failure is extended.

As explained in Section II-D, the sleep duration is given by a random value in a predefined interval: $[\texttt{MIN_SLEEP_PERIOD}, \alpha \times \texttt{ACTIVE_PERIOD}]$. Given the parameters setting, in a given cycle, for $\alpha = 10$ for instance, any node would sleep an average period approximated by $\text{random}(50, 200 * 10) \simeq 1025$ ms, in `MED_N_ADAP`. Which leads to a sleep ratio equal to: $\frac{1025*100}{1225} = 83\%$. Experiments yield an average value $\simeq$ 70%. A duty cycle of 70% means an energy gain of the same value. Notice also that many nodes reach a duty cycle of 90%.

Another observation form Fig. 2, is that, as expected, `INFR` scenario ensures the highest values of the duty cycle. The average duty cycle increases with α for `INFR` scenario, while it is almost insensitive to α for `MED_N_ADAP` and `MED_ADAP`.

$\alpha = 30$ is globally ensuring higher duty cycle than $\alpha = 10$. However, increasing α at any node, means that less neighbors of this node will be awake (no activity sensed). Thus, such node spends longer time searching for forwarders, which decreases its duty cycle and increases its energy consumption. This is what we see while evaluating the individual duty cycle of nodes (see Fig. 16 in [13]). In `MED_N_ADAP` scenario for instance, $\alpha = 10$ exceeds $\alpha = 30$ for some nodes. In contrast, in `INFR` and `MED_ADAP` scenarios, the duty cycle increases with α almost for all nodes. This means that regulating traffic injection (`INFR` scenario) and tuning the duty cycle (`MED_ADAP` scenario) allow the network to take advantage from setting long sleep durations.

IV. Conclusion

In this paper, we proposed ODYSSE routing protocol, which combines three main mechanisms: duty cycling, opportunistic routing and source coding as a reliability enhancement method. ODYSSE is also characterized by a duty cycle setting which is adaptable to different application scenarios: bulk data transfer with/without adaptive duty cycle and infrequent data transfer. Experimental results show that adapting the duty cycle to network traffic conditions is essential, even more important than the sleep period itself. Results highlight the importance of wireless protocols parameters as well: these parameters should dynamically adapt to the environment, in particular the data generation pattern.

ACKNOWLEDGEMENT

This work has been supported by the Celtic plus project TILAS [12].

References

[1] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*, John Wiley & Sons, 2005

[2] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, *Low power, low delay: opportunistic routing meets duty cycling*, in Proceedings of the 11th international conference on Information Processing in Sensor Networks, IPSN'12. New York, NY, USA.

[3] M. Chien-Chen Hung, K. Ching-Ju Lin, C.F Chou, and C.C. Hsu, *EFFORT: Energy-efficient opportunistic routing technology in wireless sensor networks*, Wireless communications and mobile computing, June 2013.

[4] C.C. Hsu, M.S. Kuo, S.C Wang, C.F. Chou, *Joint Design of Asynchronous Sleep-Wake Scheduling and Opportunistic Routing in Wireless Sensor Networks*, in Computers, IEEE Transactions, vol.63, no.7, pp.1840-1846, July 2014.

[5] Y. Sun, O. Gurewitz, and D. B. Johnson, *RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks*, in 6th ACM Conference on Embedded Network Sensor Systems. New York, NY, USA: ACM, 2008, pp. 1-14.

[6] J. Polastre, J. Hill, and D. Culler, *Versatile Low Power Media Access for Wireless Sensor Networks*, in 2nd International Conference on Embedded Networked Sensor Systems. ACM, 2004, pp. 95-107.

[7] N. Javaid,, O. Rehman, N. Alrajeh, Z. A. Khan, B. Manzoor, S. Ahmed, *AID: An Energy Efficient Decoding Scheme for LDPC Codes in Wireless Body Area Sensor Networks* , 2013 International Workshop on Communications and Sensor Networks (ComSense-2013).

[8] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, T. Watteyne, *FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed - A valuable tool for IoT deployment in Smart Factories*, IEEE ComSoc Multimedia Technical Committee E-Letter, Special Issue on "IoT and Smart Factory", 2015.

[9] M. Y. Naderi, H.R. Rabiee, M. Khansari, M. Salehi, *Error control for multimedia communications in wireless sensor networks: A comparative performance analysis*, Ad Hoc Networks. (2012).

[10] M. Cunche, V. Roca, *Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme*, Research Report 2008, pp.19. https://hal.inria.fr/inria-00263682v1/document

[11] ODYSSE: http://odysse-upec.github.io

[12] TILAS project: http://www.tilas.eu/

[13] I. Amdouni, C. Adjih, N. AitSaadi & P. Muhlethaler, *ODYSSE: A Routing Protocol for Wireless Sensor Networks*, Research Report RR-8873, Inria, https://hal.inria.fr/hal-01292479/.