Tee: Traffic-based Energy Estimators for duty cycled Wireless Sensor Networks

Rémy Léone*† Jérémie Leguay‡ Paolo Medagliani‡ Claude Chaudet†
* Thales Communications & Security – Gennevilliers, France
† Institut Mines-Télécom / Télécom ParisTech / CNRS LTCI UMR 5141 – Paris, France
‡ Mathematical and Algorithmic Sciences Lab, France Research Center, Huawei Technologies Co. Ltd.

***Abstract*—Energy is classically considered as a critical resource in Wireless Sensor Networks (WSNs). These networks are composed of tiny devices that auto-organize around one or few *gateways*, which may have various roles from simple reference or traffic sinks to full network orchestrator.**

Such a gateway could influence the network behavior, for instance by decreasing activity when energy becomes scarce. It however needs to be able to estimate the nodes remaining energy. Indeed, this gateway is on the path of all traffic going in or out the WSN. This traffic sample could be used to acquire a coarse estimate of individual nodes energy consumption. The accuracy of this estimation can then be improved by explicit signaling if needed. This paper presents Tee, a set of such *Traffic-based energy estimators* that operates at the WSN gateway.

We evaluate, by simulation, the accuracy of two such estimators in IEEE 802.15.4 networks running RPL and ContikiMAC, a duty cycled MAC layer. Results show that such silent estimators benefit from information already available at the gateway, such as the routing topology. However, they still underestimate the consumption due to the routing control messages, to the packets strobing, or to contention and collisions and can easily be complemented by lightweight explicit calibrations.

Keywords*—*Wireless Sensors Networks, Internet of Things.

I. Introduction

A Wireless Sensor Network (WSN) is a network formed by tiny devices, with limited computation, storage and communication capabilities, whose role is to report measurements. In classical surveillance and monitoring scenarios [1], these devices self-organize to form a multi-hop wireless network around a central, more powerful node, that either plays the role of a *gateway* towards a more classical network, or acts as a data collection point. As wireless sensor nodes are generally deployed for long-lasting operations, dynamically adapting nodes parameters to satisfy lifetime objectives is a key feature.

In most of research works, the WSN gateway has been seen as the source or destination of most application traffic. It plays the role of a reference node for several protocols (e.g., Routing Protocol for Low-power and lossy links, RPL, [2] or slotted 802.15.4 [3] MAC protocol). More recently, the gateway has been considered as a full network orchestrator that sets routes and organizes medium access, like in the 6TiSCH [4] group at IETF, where the aim is to manage label-switched networks based on multichannel 802.15.4e MAC layer. Indeed, the gateway is, in many WSN deployments, the node with the most detailed knowledge of the network: it forwards most of the application traffic that goes in or out of the sensor network, and plays a central role for several routing protocols. It can thus acquire a good vision of the network in terms of (i) topology, (ii) data services, (iii) radio resource allocation, and (iv) status of nodes. With such knowledge, the gateway can estimate without explicit communications nodes status (e.g., remaining node lifetime).

This paper presents *Tee*, a set of *Traffic-based energy estimators* that can be implemented on sensor network gateways. Basically, Tee estimates the energy spent by each node by monitoring a minimum set of parameters such as the traffic received or forwarded by the gateway. Our primary objective is not to increase the network lifetime, but rather to estimate it, letting the gateway play on the various network parameters when necessary. For instance, when the network energy becomes scarce, the gateway could increase nodes sleeping periods, filter unnecessary data acquisition requests, divert traffic by influencing the routing protocol, etc.

In this paper, we study the accuracy of two Tee estimators that opportunistically make use of an incremental amount of available information: the first estimator only exploits information about the source and destination of packets, whereas the second relies also on the knowledge of the routing paths. As Tee works from the gateway, it may not capture all the energy consumption induced for instance by control traffic and MAC layer mechanisms. For this reason, we introduce a rescaling factor which takes into account duty cycles and a feedback loop via periodic recalibrations so that the estimators can learn and correct their errors.

This paper considers a typical monitoring deployment where nodes are running ContikiMAC [5], a duty cycled MAC protocol, and the RPL [2] routing protocol. Through COOJA [6] simulations, we evaluate the different strategies and compare them against an estimated energy consumption reported by the simulator. Results show that the knowledge of the routes significantly improves estimations. Using recalibration, we also demonstrate how the gateway can learn about the residual consumption induced by the routing protocol and the radio conditions (contention, collisions). Then it evaluates the errors committed and corrects the estimations accordingly.

This paper is structured as follows. Sec. II reviews related work. Sec. III describes network models that serve as a basis

Jérémie Leguay & Paolo Medagliani were working for Thales Communications & Security during the redaction of this paper.

for our estimators. Sec. IV introduces our estimators and Sec. V presents their evaluation.

II. Related Work

A few contributions examined the general problem of monitoring a wireless sensor network in an energy-efficient way. For instance, [7] and [8] consider the problem of selecting a subset of sensors (pollers) in charge of actively monitoring the other sensors (pollees) liveliness. The pollers are in charge of raising alarms to the gateway when they detect a fault. [7] proposes a distributed approximation algorithm to select a minimum number of pollers and studies the effect on the false alarm rate. [8] proposes to reduce the polling overhead by using routing control packets to select pollers and by embedding monitoring reports in the control messages of the routing protocol. However, these approaches require explicit information acquired from sensor nodes, while we expect this information to be optional.

In [9], the authors propose Livenet, a semi-passive monitoring architecture that relies on packet sniffers located within the network. Using aggregated traces transmitted to the gateway, Livenet is able to reconstruct network topology and determine various network performance metrics. This work explicitly targets energy monitoring, but could be adapted to other performance indicators. However, it requires the transmission and the process of packet logs, so it is not fully relevant for our goal.

In [1], the authors introduce a distributed method to create an energy map of a WSN. Sensors report their residual energy level to a neighbor node, in charge of aggregating and compressing this information and transmitting only incremental updates (digests) to the gateway. Following this idea, [10] lets each node estimate the evolution of its energy and transmit this prediction to the network monitor. In addition, [10] compares a probabilistic method, based on Markov chains, and a statistical method, based on an auto-regressive model, with a simple, explicit reporting method. In [11], the authors extend this idea by modeling the nodes energy with a hidden Markov model whose coefficients are tuned with explicit measurements. In [12], authors also build such an energy map over clusters and change the monitoring structure regularly to redistribute the energy cost more fairly across the network. If the idea of building a network energy map relates closely to our first goal, all aforementioned methods strongly rely on explicit energy reporting from the nodes. These reports cannot be totally avoided, as our experiments demonstrate. However, we believe that a slight part of the information can be directly extracted from existing control packets without any additional overhead.

III. Network model

In most of today WSNs, the radio interface is the major source of energy consumption. The micro-controller and the system bus both operate at low frequencies and only flash memory writing operations need comparable power. In particular, transmit and reception powers are generally similar. Indeed, while sending data requires generating a radio signal, the reception of data frames requires activating complex electronics to filter out signal from noise [13]. This paper thus considers cases where the consumption of nodes is only driven by their radio interface. This section details the networks protocols that come into play at each layer and drive the energy consumption of nodes.

A. MAC layer

The MAC layer is in charge of organizing how nodes access the shared wireless channel. In WSNs, it is also responsible for defining the radio duty cycling, i.e., when the nodes turn on or off their radio interfaces. In our work, we consider ContikiMAC [5], a duty cycling mechanism built above the IEEE 802.15.4 MAC and physical layers. In ContikiMAC, nodes sleep most of the time and wake-up periodically to check channel activity. As an emitter and a receiver may lose synchronization, an emitter willing to send a packet will make several successive attempts (packet *strobing*). Sending node repeats packet transmission until it receives an acknowledgment, or until the maximum number of attempts is exceeded. A waking-up node senses the channel and, if it detects energy on the channel, it remains awake until it successfully receives the whole frame. If this node is the intended receiver, it will acknowledge the sender that will stop packet transmission. ContikiMAC uses the *phase-lock* mechanism to learn the wake-up period of neighbors and dynamically reduce this strobing effort.

As the gateway has no energy constraints, we consider that the gateway node has no duty cycling and continuously listens to the channel when it is not emitting. In this paper, we consider the beaconless version of IEEE 802.15.4 that implements Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).

B. Network Layer

At the network level, the goal of a routing protocol is to dynamically establish and maintain routes. Routing protocols generate overhead traffic that could be significant, especially in dynamic scenarios.

In the rest of this article, we consider a static network deployment with RPL [2]. In RPL, the gateway periodically broadcasts control messages, referred to as DODAG Information Object (DIO), to build a Destination-Oriented Directed Acyclic Graph (DODAG). A DODAG is similar to a tree, but a node can have multiple parents in the structure, as the graph is directed towards the root of the routing tree, i.e. the gateway. DIO messages contain a rank value that is initialized by the gateway at a value of 0 and that is incremented by every forwarder. This rank is a discrete representation of the network metric exploited by the nodes. Possible examples of RPL metrics are the number of traversed hops, the delay, the remaining energy, etc. A node receives DIO messages from all its neighbors, selects the one(s) announcing the smallest parent(s) rank in the DODAG, increases the rank value, and broadcasts an updated DIO to all its neighbors.

DIO messages only allow building upwards routes towards the root. Downwards routes are an optional feature. They are built by letting each node send a Destination Advertisement Object (DAO) message to the root.

Since DIO control messages are broadcasted, this operation can be costly for the sender if receiving nodes use low-power listening mechanisms. However, RPL uses a Trickle algorithm [14] to dynamically adapt the emission interval of control messages. DIO are sent by every node to maintain the upwards routes, but their emission rate decreases exponentially up to a maximum when no topology changes occur. Consequently, we can expect a large amount of RPL traffic when the network starts up and a quick decrease after.

IV. Tee: Design and Implementation

Our goal with Tee is to study how much the gateway can estimate nodes energy by looking at the data packets it receives or forwards. As the wireless interface is the primal source of energy consumption, the gateway tries to estimate how much energy nodes spend for emitting and receiving each data packet.

A. Transmission and reception cost at each hop

We first need to estimate how much energy is spent at each hop for a frame m of size $s(m)$-byte sent in unicast mode and with acknowledgment.

In IEEE 802.15.4, the time to transmit a message m (e.g., data packet, strobes, ACK) can be expressed as:

$$T_p(m) = \left(\frac{8s(m)}{R} + \left\lceil \frac{s(m)}{L} \right\rceil h \right)$$

where $s(m)$ be the size of 802.15.4 frame (expressed in bytes) containing m, $R = 250$ kbit/s is the transmission rate of 802.15.4, L the maximum payload of a 802.15.4 frame (127 bytes) and h is the time required to transmit the header of a frame. The 802.15.4 standard gives us a value of $h = 992\mu s$ for the headers, which we confirmed by simulation. Since headers are sent for every frame, we also take into account the overhead of sending several headers in case of a packet fragmentation.

Let P_{TX} and P_{RX} be the power required to emit and receive data respectively. If we multiply $T_p(m)$ by P_{TX} (respectively P_{RX}) we obtain the energy cost of transmitting (respectively receiving) m. Missing parameters can be found in datasheets. For instance, the Chipcon CC2420 radio chip [15] that implements IEEE 802.15.4 operates with a voltage $V_{\text{DD}} = 3V$, the reception current consumption is $I_{\text{RX}} = 19.7\,mA$ and the emission current consumption at 0 dBm is $I_{\text{TX}} = 17.4\,mA$ [16].

We can thus estimate $S_{\text{cost}}(m)$ and $R_{\text{cost}}(m)$, the energy necessary to respectively send and receive a frame:

$$S_{\text{cost}}(m) = P_{\text{TX}} N_{\text{sender}}(m) T_p(m) + P_{\text{RX}} t(ACK)$$

$$R_{\text{cost}}(m) = P_{\text{RX}} N_{\text{receiver}}(m) T_p(m) + P_{\text{TX}} t(ACK)$$

where $N_{\text{sender}}(m)$ and $N_{\text{receiver}}(m)$ are the number of attempts processed by the sender and the receiver, respectively. Indeed, with ContikiMAC, the sender has to transmit *strobe* frames before receiving an acknowledgment. Even with the phase-lock mechanism of ContikiMAC, clock drifts can make the sender perform multiple attempts for a single frame.

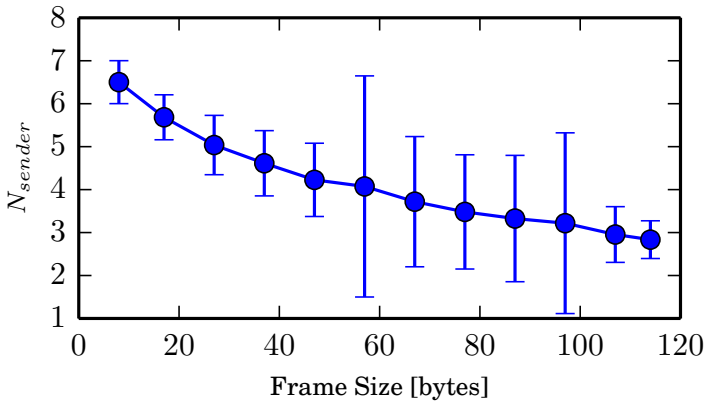


Fig. 1: Average number of strobing attempts depending on frame size.

We study this effect, in simulations, by measuring how many attempts are necessary on average using a two-nodes topology. Since there is only a sender and a receiver, the channel does not suffer from contention and we can focus on the duty-cycling impact. Fig. 1 represents the average number of transmissions $N_{\text{sender}}(m)$ necessary to send a $s(m)$ bytes frame successfully between two nodes. We can notice that the number of attempts decreases as the frame size increases, which is natural as frames occupy the medium during a larger time and wake up the receiver more often. Hence, fewer attempts are necessary than for shorter frames for a comparable time offset.

On the receiver's side, the scenario is easier to model. The receiver wakes up on average in the middle of a transmission and waits until the next attempt is performed to fully receive the frame and to send back the acknowledgment. The number of frame receptions can therefore be estimated by $N_{\text{receiver}} = 1.5$. For nodes in direct range to the gateway, the transmission is with ContikiMAC but the reception is without sleeping cycles on the gateway, which naturally leads to $N_{\text{sender}}(m) = 1$.

B. Traffic-based estimators

Using an estimation of the cost for exchanging a frame between two nodes, Tee is able to predict, at gateway level, the energy impact of data packet transmissions on all the network nodes. Then, the gateway uses of one of those two estimators according to the levels of knowledge on the network available.

1) Noinfo estimator: The simplest estimator, named *Noinfo*, only estimates packets impact at the source and the destination and can be used anywhere. In the scenario presented in Fig. 2 in which a node E sends a frame to the gateway G, Noinfo only takes out energy from nodes E and G and leaves the energy of all other nodes unchanged for a given packet.

Let $\mathcal{D}_i$ be the messages that originates from i and $\mathcal{A}_i$ be the messages that ends at i. We obtain the following estimated energy:

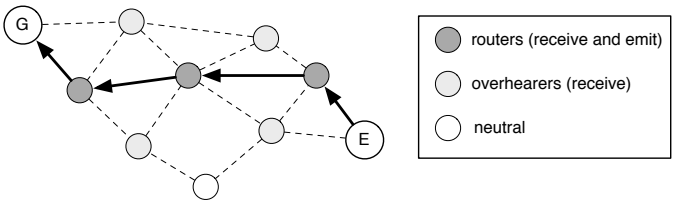


Fig. 2: Nodes potentially impacted by the forwarding of a frame from node E to gateway G.

$$\widehat{E}_i(t) = \sum_{m \in \mathcal{D}_i(t)} S_{\text{cost}}(m) + \sum_{m \in \mathcal{A}_i(t)} R_{\text{cost}}(m).$$

This estimator requires no knowledge and under-estimates the effect of a packet on the network unless all nodes are in-range of the gateway. For instance, it does not capture transmissions induced by forwarding actions at intermediary nodes.

2) Route estimator: In multi-hop topologies, the gateway will likely be aware of the network active routes by using network mapping tools or through the routing protocol.

The second estimator we examine, referred to as *Route*, uses this information and subtracts from the remaining energy of each node on the route (dark gray nodes in Fig. 2) the cost of a reception and an emission of the frame.

Let $\mathcal{F}_i$ be the message set forwarded by the node i while it's neither the destination nor the source of the message.

$$\widehat{E}_i(t) = \sum_{m \in \mathcal{D}_i(t) \cup \mathcal{F}_i(t)} S_{\text{cost}}(m) + \sum_{m \in \mathcal{A}_i(t) \cup \mathcal{F}_i(t)} R_{\text{cost}}(m)$$

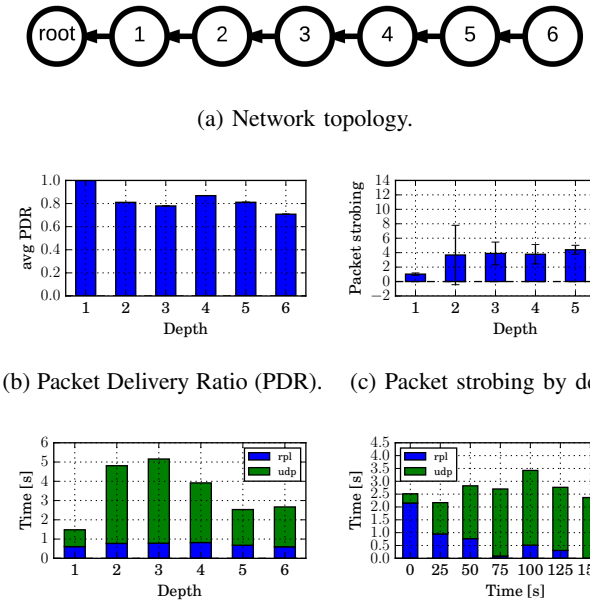
V. Experimental results

We evaluated the accuracy of the different estimators described above using the COOJA simulator with its Powertracker extension, which measures and reports the time spent by each node in reception and transmission state with $1\mu s$ resolution. All nodes run the latest development branch of Contiki and use RPL for routing and ContikiMAC over IEEE 802.15.4.

A. Chain topology

Let us consider a simple 7-nodes chain topology, as represented on Fig. 3a. In this scenario, each node sends to the root a 10-bytes UDP message every second, which results in 69 bytes frames at the link layer. This rate of traffic is higher than classical surveillance radio traffic but has been considered for practical reasons, as it makes simulations shorter because traffic is generated more quickly. We thus use in our estimators an average strobing of 3.76 packets, taking this value from our calibration experiment in Sec. IV. Nodes closer to the root have to forward traffic coming from downstream nodes in addition to their own traffic. Simulations run for 200 seconds.

Fig. 3b represents the packet delivery ratio at each node as a function of its distance to the root. We can notice that



(a) Network topology.

(b) Packet Delivery Ratio (PDR).

(c) Packet strobing by depth.

(d) Protocol repartition by depth.

(e) Protocol repartition evolution.

Fig. 3: Network and traffic characteristics of the chain experiment.

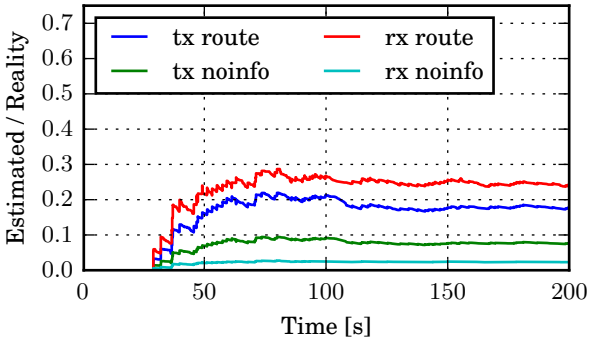
the delivery ratio is not uniform across the network and that nodes that are not directly connected to the root suffer from congestion or collisions. This figure gives us an idea of the proportion of the traffic that the gateway can actually see.

Fig. 3c represents the average number of strobing packets, i.e. the number of attempts each node needs to send a frame to its upwards neighbor. This metric quantifies the cost of sending a single frame over each hop. We can notice here a better correlation between the distance to the root and this figure.

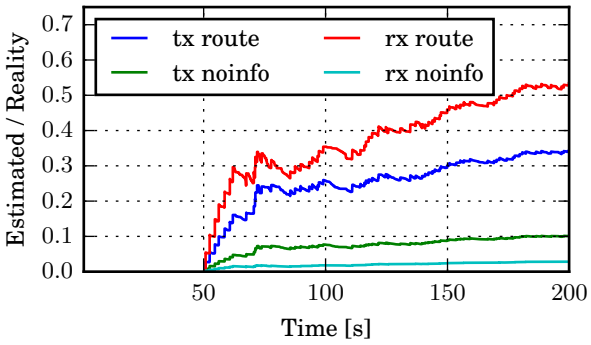
Fig. 3d represents the distribution of traffic types at each hop (UDP and routing messages). This measurement is important, as it shows the amount of control traffic that the gateway may not measure and shall infer. We can notice that, at this stage, the traffic is almost identical for the different nodes and hence could be modeled by a constant rate. However, Fig. 3e shows the evolution over time. As expected, we can notice that many routing packets are required at the beginning to build the RPL tree. Once this phase is over, thanks to Trickle, the majority of the traffic is due to the application. This confirms that the chaotic construction (and reparation) of the routing tree generates a large volume of traffic that is hard to predict. It however motivates the relevance of our applicative traffic-based estimators when the network is in stationary state.

Let us now compare estimations and real values. Fig. 4 represents the ratio between the estimated and the real time spent in transmission and reception states for nodes 3 and 4.

We can see that the *Noinfo* estimator largely underestimates the nodes activity. This phenomenon is expected since *Noinfo* ignores the cost at intermediary nodes. The costs are only



(a) Node 3.



(b) Node 4.

Fig. 4: Relative error in the chain scenario.

related to the traffic generated by each node and integrate a corrective factor for the strobing induced by duty cycles. Route produces a better estimation by taking also into account the impact at intermediary nodes, even though it is still below the real values. The initial under-estimation can be explained by the bootstrap phase illustrated by Fig. 3d, as routing packets are invisible to the gateway. The UDP traffic starts on each node when RPL has found a route to the gateway. However, the difference remains large even after the routing tree is established, which means that a sole blind estimation is insufficient; explicit correction mechanisms are necessary.

B. Explicit dynamic recalibration

We consider periodic recalibrations where each node send explicitly the time spent in emission and reception to the gateway, which updates the estimators accordingly. This information requires a specific packet transmission or can be piggybacked in routing or data packets. As these messages seem unavoidable, their frequency should be as low as possible.

When the gateway receives an explicit recalibration at time t, it rescales its estimator for the source node, $\widehat{E}(t)$ as follows:

$$\widehat{E}(t) = E(t_{\mathrm{r}}) + R_i(t) + S_i(t) + \epsilon(t_{\mathrm{r}})\frac{(t - t_{\mathrm{r}})}{T}\,,$$

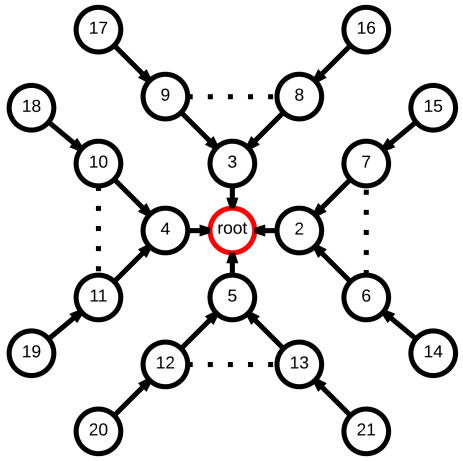


Fig. 5: Tree topology. Dashed lines represent the radio overhearing between nodes pair.

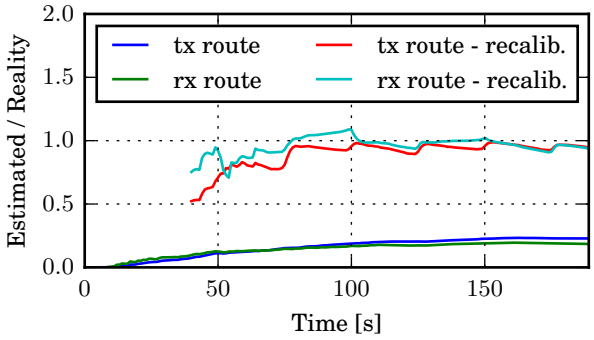


Fig. 6: Relative error for the tree topology with and without dynamic recalibration (every 25 seconds).

$$\epsilon(t_r) = \alpha.(E(t_{\mathrm{r}}) - \widehat{E}(t_{\mathrm{r}})) + (1 - \alpha).\epsilon(t_{\mathrm{r}-1})\,,$$

where t_{r} and $t_{\mathrm{r}-1}$ are the times of the two last recalibrations. S_i and R_i are respectively the estimated sending and receiving costs induced by applicative traffic since the last recalibration and $E(t_r)$ is the real energy level of the node at date t_r. $\epsilon(t_{\mathrm{r}})$ is the estimation error learnt from previous recalibration. It should integrate all the consumption that our base estimator misses, such as the background RPL traffic or the amount of retransmissions due to collisions. It can be calculated using an exponential moving average, as shown in the second equation. At time t, this error is taken proportionally to the time from the last recalibration (occurring every T). Note that $\epsilon(0) = 0$.

We evaluated this recalibration process considering the tree-like scenario represented on Fig. 5. The topology is composed of 21 client nodes sending packets to the root sending packets every second. We took a value of $\alpha = 0.25$ in order to not overreact too quickly in case of RPL bursts.

As we can see on Fig. 6, the periodic recalibration provides a much better estimation. Despite that the estimator diverges

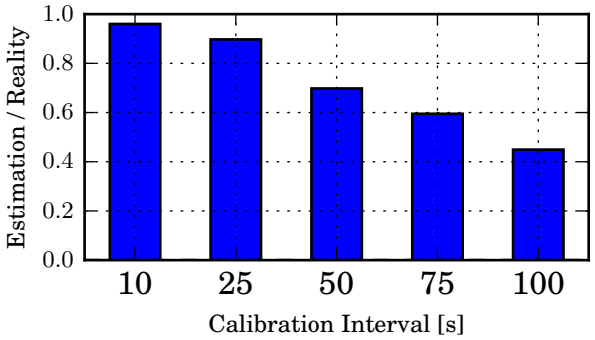


Fig. 7: Average relative error for different recalibration intervals.

at the beginning, due to network bootstrap as explained in Sec. V-A, it quickly provides accurate predictions of the energy depletion. When the recalibration happens, the estimator integrates the average error from previous recalibrations and converge quickly to accurate estimation.

Recalibration therefore improves estimation accuracy, but it has a linear cost with the recalibration interval and should be as infrequent as possible. Fig. 7 represents the average ratio achieved by the *Route* estimator over the 200 seconds simulation as a function of the recalibration events interval. As we have a regular traffic pattern, this figure accounts for the length of the transient period until the estimators converge. In order to reduce overhead, the pace of recalibrations could decrease over time, as the network stabilizes.

VI. Conclusions and perspectives

In this paper, we introduced *Tee*, a set of traffic-based estimators running on the gateway of a wireless sensor network in order to let it evaluate individual nodes energy consumption based on the applicative traffic.

We show, through simulations performed with COOJA, that using information already available at the gateway, such as the network topology, improves the performance of traffic-based estimators. However, only looking at the traffic passing through the gateway is not enough to have an accurate estimation. A rescaling mechanism is necessary to account for the effect of duty cycling at the MAC layer, since the sleeping interval introduces a desynchronization between emitters and receivers, which results in multiple transmission attempts for each frame.

Even in the case of a constant bit rate traffic, nodes need to send explicit energy reports, that we called recalibrations, to the gateway, at least in the network initialization phase. This is necessary to take into account the routing packets exchanged to build the initial tree. In case of a static network and a regular traffic pattern, such recalibrations rapidly become unnecessary. In presence of nodes mobility, the control traffic due to the reconstruction of the routing tree will most likely make such explicit recalibrations necessary throughout the network life. In presence of an irregular traffic pattern, we believe that the dynamic can be captured by the gateway, as it is generally the source or the destination of traffic.

Future work will first consist in validating these results in a real environment. To this extent, we dispose of large-scale experimental platforms with different sensors architectures supported by Contiki, which will permit measuring the real energy consumed. Besides, we also need to confront both the estimators and the protocol suite to a real, error-prone, wireless medium. Then, we plan on studying the effect of network dynamics by testing irregular traffic patterns and nodes arrivals before focusing on characterizing the trade-off between in-network mobility and the recalibration frequency required to reach a predefined accuracy level. Finally, we will introduce an improved estimator that leverages on the knowledge of radio interference between nodes to improve estimation accuracy.

VII. Acknowledgment

This work is supported by the French National Research Agency (ANR) under grant reference IRIS ANR-11-INFR-0016. A part of this work was realized at the LINCS laboratory.

References

[1] Jerry Zhao, Ramesh Govindan, and Deborah Estrin, "Residual energy scans for monitoring wireless sensor networks," in *IEEE WCNC*, 2002.

[2] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF RFC 6550, Mar. 2012.

[3] Tae Rim Park, Tae Hyun Kim, Jae Young Choi, Sunghyun Choi, and Wook Hyun Kwon, "Throughput and energy consumption analysis of ieee 802.15. 4 slotted csma/ca," *Electronics Letters*, vol. 41, no. 18, pp. 1017–1019, 2005.

[4] 6tisch, "IPv6 over the TSCH mode of IEEE 802.15.4e," http://datatracker.ietf.org/wg/6tisch/.

[5] Adam Dunkels, "The contikimac radio duty cycling protocol," Tech. Rep. T2011:13, Swedish Institute of Computer Science, 2011.

[6] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt, "Cross-level sensor network simulation with cooja.," in *LCN*, 2006.

[7] Changlei Liu and Guohong Cao, "Distributed monitoring and aggregation in wireless sensor networks," in *IEEE INFOCOM*, San Diego, CA, USA, 2010.

[8] Abdelkader Lahmadi, Alexandre Boeglin, and Olivier Festor, "Efficient distributed monitoring in 6lowpan networks," in *CNSM*, Zurich, Switzerland, 2013.

[9] Bor-rong Chen, Geoffrey Peterson, Geoff Mainland, and Matt Welsh, "Livenet: Using passive monitoring to reconstruct sensor network dynamics," in *IEEE/ACM DCOSS*, 2008.

[10] Raquel A.F. Mini, Antonio A.F. Loureiro, and Badri Nath, "The distinctive design characteristic of a wireless sensor network: the energy map," *Computer Communications*, vol. 27, 2004.

[11] Peng Hu, Zude Zhou, Quan Liu, and Fangmin Li, "The hmm-based modeling for the energy level prediction in wireless sensor networks," in *IEEE ICIEA*, Harbin, China, 2007.

[12] Edward Chan and Song Han, "Energy efficient residual energy monitoring in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 5, no. 6, 2009.

[13] Philipp Hurni, Benjamin Nyffenegger, Torsten Braun, and Anton Hergenroeder, "On the accuracy of software-based energy estimation techniques," in *Wireless Sensor Networks*, pp. 49–64. Springer, 2011.

[14] P. Levis, T. Clausen, J. Hui, O. Gnawali, and K. J., "The Trickle Algorithm," IETF RFC 6206, 2011.

[15] Chipcon, *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*.

[16] Moteiv Corp., "Ultra low power IEEE 802.15.4 compliant wireless sensor module," .