EAI Endorsed Transactions
on Mobile Communications and Applications
Research Article ICST.ORG

History-based MAC Protocol for Low Duty-Cycle Wireless Sensor Networks: the SLACK-MAC Protocol

Affoua Thérèse Aby[1,2,*], Alexandre Guitton[1,2], Pascal Lafourcade[3,2], Michel Misson[3,2]

[1]Université Blaise Pascal, LIMOS, BP 10448, F-63000 Clermont-Ferrand, France
[2]CNRS, UMR 6158, LIMOS, F-63173 Aubière, France
[3]Université d'Auvergne, LIMOS, BP 10448, F-63000 Clermont-Ferrand, France

Abstract

Wireless sensor networks (WSNs) are increasingly used in environmental monitoring applications. They are designed to operate for several months by featuring low activity cycles in order to save energy. In this paper, we propose a Medium Access Control (MAC) protocol for such WSNs with very low duty-cycles of 1% and less. Nodes are activated randomly and use a history of previous successful frame exchanges to decide their next activation time. We study the choice of the history size, and we compare the performance of our protocol with other protocols from the literature. We show by simulations and real experiments that with a limited history size of only six entries, our protocol achieves better performance than other protocols from the literature, while keeping the advantages of fully asynchronous protocols.

Received on 30 September 2015; accepted on 22 December 2015; published on 20 June 2016

Keywords: WSN; adaptive; asynchronous MAC protocol; duty-cycle

doi:10.4108/eai.20-6-2016.151516

1. Introduction

Environmental monitoring applications, such as the monitoring of volcanoes [1], bird nests [2], fields [3], or bridges [4], are increasingly using Wireless Sensor Networks (WSNs). In such applications, wireless sensor nodes are deployed in the environment, where they perform periodic measurements and communicate the collected data to a sink in a multi-hop manner.

Energy-efficient protocols are designed to increase the lifetime of such WSNs. These protocols deactivate the radio module of nodes most of the time, as the radio module is the node hardware component having the largest energy consumption. The MAC protocol is responsible for allowing nodes to communicate in the rare periods when the radio module of both nodes is active.

In this paper, we present the SLACK-MAC (Self-adaptive Low Activity Cycle Knowledge-based MAC) protocol (extended version of [5]), which is an asynchronous protocol where nodes activate their radio modules randomly. The idea behind SLACK-MAC is inspired by the routing protocol proposed in [6], where authors proposed the SR3 protocol, which is an improvement over a biased random walk based on a reputation mechanism. In SLACK-MAC, nodes communicate opportunistically. Nodes record a history of previous successful communications with neighbors, and use this history to determine the instant of the next activation of their radio module. This history increases the probability to select a recent successful instant of activation among all possible instants during each cycle. Thus, nodes adapt their activation time depending on their neighborhood. In this paper, we show that this behavior improves the probability of successful communications, which in turn improves the performance in terms of delivery rate and delay.

This paper is an extended version of [5]. It improves the description of [5] on several aspects, including mainly: (i) an experimental evaluation of SLACK-MAC using TelosB motes, (ii) new simulation results comparing SLACK-MAC with additional asynchronous protocols from the literature [7, 8], and (iii) the energy consumption as a new metric.

The remainder of this paper is as follows. Section 2 presents the main existing MAC protocols for low duty-cycles. Section 3 describes the SLACK-MAC protocol, and justifies our choice of parameters. Section 4 compares the performance of existing protocols with the performance of SLACK-MAC, based on both simulation results and on experimental results. Finally, Section 5 concludes our work.

*Affoua Thérèse Aby. Email: aby@sancy.univ-bpclermont.fr

2. State of the art

Most energy-efficient MAC protocols for WSNs are based on sequences of active and inactive periods, called *duty-cycle*. As the radio module of a node is the component having the largest energy consumption, some energy can be saved by deactivating it periodically. MAC protocols based on duty-cycles can be classified depending on whether the activities of nodes are synchronized or not.

2.1. Synchronous MAC protocols

In synchronous duty-cycle MAC protocols, nodes share a common time (through synchronization) and agree on a common schedule for their activities and inactivities. Generally, all nodes are either simultaneously active or simultaneously inactive.

The IEEE 802.15.4 standard [9] in *beacon-enabled* mode is one of the most largely used synchronous MAC protocols. Full-function devices send periodic beacons, with period BI (for Beacon Interval). Reduced-function devices start their activities at the beacon reception, and are allowed to communicate during a period SD (for Superframe Duration). The communication is performed using the slotted CSMA/CA (Carrier-Sense Multiple Access with Collision Avoidance) mechanism, which is designed to consume little energy for channel sensing. After this period, nodes go back to sleep until the next beacon. The ratio SD/BI defines the duty-cycle of nodes.

Several other protocols have been proposed for the same purpose, such as D-MAC [10], DW-MAC [11], Speed-MAC [12], TreeMAC [13], MC-LMAC [14], SEA-MAC [15] and others [16–18]. As in IEEE 802.15.4, these protocols generally have three types of periods: a *synchronization period*, which ensures that all nodes share a common time, a *communication period*, where nodes can communicate efficiently, and an *inactive period*, where nodes save energy.

Synchronous duty-cycle MAC protocols have two main drawbacks. The first drawback is the overhead of the mandatory synchronization period. The second drawback is the high contention for the channel when all nodes are active simultaneously. In this paper, we focus on asynchronous duty-cycle MAC protocols.

2.2. Asynchronous MAC protocols

Asynchronous duty-cycle MAC protocols do not need to synchronize nodes. Notice that generally, asynchronous MAC protocols yield large delays, but have a low energy consumption. In the following, we describe the two main categories of asynchronous protocols: sender-initiated protocols and receiver-initiated protocols.

Sender-initiated protocols. The first asynchronous MAC protocol for WSNs was B-MAC [19], which is based on LPL (Low Power Listening). In B-MAC, the source sends a long preamble before each data frame, and receivers wakes up periodically to detect potential preambles. This technique has been the basis for sender-initiated protocols.

In WiseMAC [20], instead of sending a long preamble before each data frame (as in B-MAC [19]), the receiver nodes include in each ACK frame the instant of their next wake-up. In that way, the sender has knowledge of the wake-up time of its receiver. If the sender has other frames to send, it proceeds to send a small preamble and starts quickly the transmission of data frames.

X-MAC [21] is based on a similar approach. In X-MAC, nodes switch between active (20 ms) and inactive periods (500 ms), but instead of using long preambles, nodes send a series of small preambles to inform the receiver. The maximum duration of the series of short preambles is one inactivity period (500 ms). Once the receiver wakes up and receives a short preamble, it replies by an acknowledgment to inform the transmitter of its availability to receive data. When a node, having no packet to send, wakes up and hears a preamble for another node, it immediately returns to sleep. When a node having packets to send wakes up and hears another preamble, it stops sending its own preamble and waits to receive the acknowledgment for the other transmission before attempting to send its own preamble again. In X-MAC, some source nodes remain active much longer than other nodes: this causes an inequity in energy consumption which reduces network lifetime. Moreover, X-MAC generally achieves a low end-to-end delay, but increases the risk of collisions due to the fact that nodes can interpret the duration between two preambles as a free channel.

In [22], the authors proposed a distributed algorithm to control the sleep interval of nodes to achieve fairness of energy consumption in asynchronous duty-cycle WSNs. The mechanism can increase network lifetime, but has a significant impact on the delay.

In this paper, we decide not to use a sender-initiated approach, in order to avoid the overhead and energy consumption caused by preambles. Indeed, for very low duty-cycles, the average duration for a preamble is long.

Receiver-initiated protocols. In RI-MAC [7], when a sender has data to send, it wakes up and waits for a beacon. When the receiver wakes up, it sends a beacon to express its ability to receive data packets, and waits for a short interval to receive potential data from a sender. If the receiver does not receive data, it goes back to sleep. In this way, the main energy cost in RI-MAC is paid by the sender waiting for a receiver to wake up, rather than by the receivers. This approach is called the receiver-initiated approach. Under low traffic conditions, senders are expected to have few data to send, and thus the overhead of RI-MAC is lower than the overhead of sender-initiated protocols, for low traffic conditions. RI-MAC also reduces channel occupation (as it does not require nodes to send preambles), but introduces a wasted period as the sender has to wait for the reception of the beacon. The ABD protocol [23] adds a broadcast service to the RI-MAC protocol.

In PW-MAC [8], each node wakes up according to a pseudo-random schedule, rather than according to a fixed schedule. Each node stores the parameters of the pseudo-random generators of its neighbors (which are transmitted in beacons). When a sender has data to send to a given receiver, it predicts the wake-up time of the receiver (based on the parameters of the pseudo-random generator), wakes up before this estimated wake-up time, and waits. When the receiver wakes up (which happens periodically), it sends a beacon and waits for potential data. Upon receiving this beacon, the sender starts sending the data for this receiver. The drawbacks of PW-MAC come from the overhead of sending beacons before frame transmissions (in terms of energy, channel occupation and delay), from predictions errors (due to clock drift for instance), and from collisions (when several senders wakes up simultaneously for the same receiver).

In EM-MAC [24], nodes decide independently their wake-up schedule and channel using a pseudo-random generator. EM-MAC allows the sender to wake up just before the beacon of the receiver. However, EM-MAC requires an initial neighbor discovery phase and each node needs to maintain information about all its neighbors.

HKMAC [25] uses an hybrid approach, where time is divided into random activation periods (as in RI-MAC) and scheduled activation periods (which requires synchronization).

The MAC protocol proposed in [26] is based on a random wake-up mechanism. Each node knows the duration of the cycle, denoted by C, and the duration of its activity within each cycle, denoted by A. Each node activates its radio module during A time units every C time units. The beginning of the activation within each cycle is chosen uniformly at random in $[0; C - A[$. When a node is active, it uses unslotted CSMA/CA to access the medium (as in the non beacon-enabled mode of IEEE 802.15.4 [9]). With this mechanism, nodes are not synchronized, and nodes do not make assumptions about the activity time of the others. Moreover, there is a non-null probability that any two neighbors share a common activity at each cycle. Figure 1 depicts an example of the activities of three neighbor: n_1, n_2 and n_3. We notice that the cycles of nodes are not synchronized. During the first cycle of n_1, nodes n_1 and n_2 share a common activity, during which they can communicate. However, for n_1 to communicate with n_3, both nodes have to wait until the middle of the third cycle of n_1.

In this paper, we focus on a receiver-initiated protocol based on random node activities, as in [26]. Such protocols are generally more suitable to low duty-cycles of 1%.

3. Proposition of a MAC protocol for low duty-cycles

In this section, we describe our SLACK-MAC protocol and our methodology to choose its parameters.

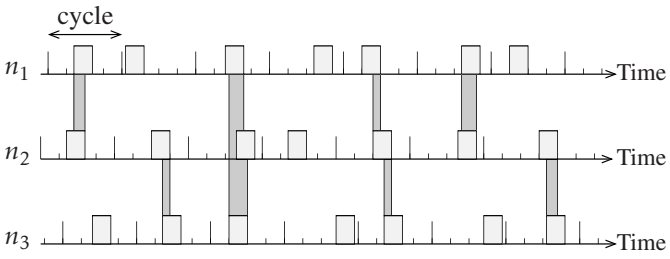

Figure 1. Example of the activities of three neighbor nodes with the protocol of [26], with a duty-cycle of 25% (this long duty-cycle is chosen for clarity).

3.1. SLACK-MAC protocol

SLACK-MAC was initially presented in [5]. The main idea of SLACK-MAC is to maintain a history of instants corresponding to successful communications with neighbors. In SLACK-MAC, nodes do not always choose their activation time uniformly at random. Instead, they have a high probability to choose instants when successful communications occurred in the recent past. Initially, all nodes choose their activation time uniformly at random. When a node chooses an instant that yields to successful communications (reception or transmission of a frame), it memorizes it and the probability to choose this instant increases, as it can be seen on Figure 2, on cycles 5 and 7 of n_1.

SLACK-MAC requires each node to maintain two lists E (Emission) and R (Reception) that contain wake-up instants in the cycle. Let us denote by t_i^{start} the start of activity in the current cycle i, and t_i^{end} the end of activity duration, as depicted on Figure 2. During this activity a node can send and receive one or several frames, and can add t_i^{start} in both lists, possibly several times. Each node uses these two lists to determine its next wake-up instant. A new time t_i^{start} is added to E when a node, having been awaken at t_i^{start}, performs at least one successful communication with another node located closer to the sink. Similarly, a new time t_i^{start} is added to R when a node wakes up in a given cycle and performs at least one successful communication with another node located further away from the sink.

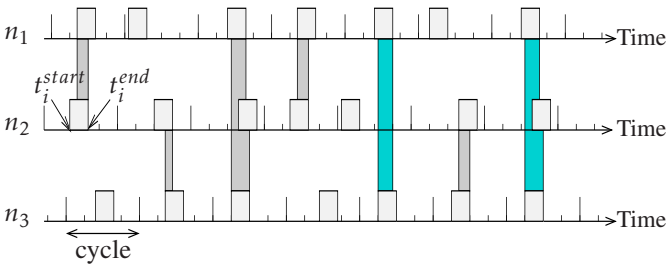

Figure 2. Example of the activities of three neighbor nodes with the SLACK-MAC protocol, with a duty-cycle of 25% (again, this long duty-cycle is chosen for clarity).

Figure 3 shows the evolution of lists E and R, at three different time steps. Step 1 shows the state of the lists when they are being filled (with one successful receptions t_1^R). When a list is full (see Step 2), the last entry t_1^R is removed to add the newest entry t_4^R to the front (using a first-in first-out mechanism, as shown on Step 3).

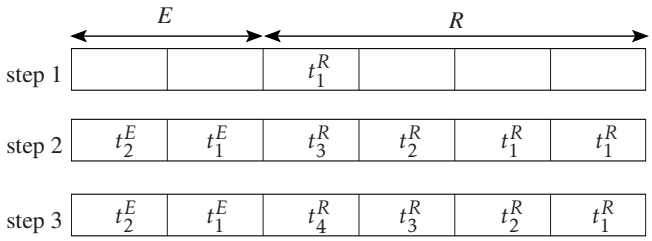

Figure 3. Example of the state of the E and R lists, at three different time.

Note that nodes consider that time is discrete (the granularity of time can be, for instance, 320 μs, as in IEEE 802.15.4, which results into 15,625 slots for a cycle C of 5 seconds).

Each node has a packet queue of fixed size for packets that have to be sent, denoted *SendQueue*. If *SendQueue* is full and a node receives a new packet, the node ignores this last packet.

In Algorithm 1, we give a pseudo code of the behavior of a node when it wakes up. During this time, a node can receive and send some data. These two operations change the content of *SendQueue* and the two lists E and R. Before going back to sleep, a node uses these lists to determine its next wake-up instant using the function Next-Wake-Up-Time presented in Algorithm 2. The function *Next-Wake-Up-Instant* draws the next wake-up instant according to the content of the two lists and following the distribution given in formula (1). In this algorithm, random(x) draws an integer uniformly at random within $[0; x-1]$, C denotes the duration of a cycle in time units, and A denotes the activity of a node in time units. Note that initially, when both lists are empty, we have $Pr[X = t] = \frac{1}{|D|}$, meaning that the next wake-up instant is chosen uniformly at random in D. If one of the two lists is empty, we select an element of the non empty list with a probability of 1/2, and uniformly at random in D otherwise. If none of the lists are empty, we select an element from list R with probability 1/3, from list E with probability 1/3, and uniformly at random in D otherwise.

In order to optimize our protocol according to the state of *SendQueue*, each node adapts its selection strategy for its next wake-up instant, according to the following rules:

- If *SendQueue* is empty ($state = 1$), a node has no packet to send, so it is useless to select instants that are in the E list. The next wake-up instant is chosen uniformly in R with probability 1/2, and uniformly at random in D (*i.e.*, all possible instants) otherwise.
- If *SendQueue* is full ($state = 2$), a node cannot accept any incoming packet, so it is useless to select instants that are in the R list. The next wake-up instant is chosen uniformly in E with probability 1/2, and uniformly at random in D otherwise.
- In all the other cases ($state = 3$), a node selects its next wake-up instant uniformly in E with probability of 1/3, uniformly in R with probability 1/3, and randomly in D otherwise.

Algorithm 1 Activity of a node.

node n wakes up at time t_i^{start} for duration A in a cycle i of C time units.
while node n (at distance d) is active **do**
  if n has received a frame from n_r (at distance d_r) during cycle i **then**
    if ($d < d_r$) and (t_i^{start} has not yet been added) **then**
      add(n_r, t_i^{start}) to R
      add frame to *SendQueue*
    end if
  end if
  if n has sent a frame to n_s (at distance d_s) during cycle i **then**
    if t_i^{start} has not yet been added **then**
      add(n_s, t_i^{start}) to E
      remove frame from *SendQueue*
    end if
  end if
end while
if *SendQueue* is empty **then**
  $t \leftarrow$ *Next-Wake-Up-Instant*(1);
else
  if *SendQueue* is full **then**
    $t \leftarrow$ *Next-Wake-Up-Instant*(2);
  else
    $t \leftarrow$ *Next-Wake-Up-Instant*(3);
  end if
end if
schedule next activity at time t of the next cycle

The probability that a node selects its next wake-up instant $t \in D$ (where D denotes all possible instants) is given by the following formula and follows exactly the mechanism given above:

$$Pr[X = t] = \frac{\mathbb{1}_{|R|\neq 0}\left(\frac{|R|_t}{|R|}\right) + \mathbb{1}_{|E|\neq 0}\left(\frac{|E|_t}{|E|}\right) + \frac{1}{|D|}}{\mathbb{1}_{|R|\neq 0} + \mathbb{1}_{|E|\neq 0} + 1} \qquad (1)$$

where $|L|$ denotes the number of elements in list L, $|L|_t$ denotes the number of occurrences of time t in L, and $\mathbb{1}_P$ is the indicator function (it is equal to 1 when the predicate P is true and to 0 otherwise).

Algorithm 2 Next wake-up instant in a cycle i.

```
Require: Next-Wake-Up-Instant(state);
  if state=1 then
    indicator ← random(2);
    if indicator = 0 then
      position ← random(sizeOf(R));
      t ← R[position];
    else
      t ← random(C − A);
    end if
  else
    if state=2 then
      indicator ← random(2);
      if indicator = 0 then
        position ← random(sizeOf(E));
        t ← E[position];
      else
        t ← random(C − A);
      end if
    else
      indicator ← random(3);
      if indicator = 0 then
        position ← random(sizeOf(E));
        t ← E[position];
      else
        if indicator = 1 then
          position ← random(sizeOf(R));
          t ← R[position];
        else
          t ← random(C − A);
        end if
      end if
    end if
  end if
  return t
```

3.2. Determination of the size of SLACK-MAC lists

In order to determine the size of both E and R lists, we need to specify the routing algorithm used in our experiments. We have taken a gradient-based routing protocol. Gradient-based routing protocols operate by estimating a distance, called the gradient, to the sink. When a node receives a frame to forward to the sink, the node sends the frame to any neighbor having a gradient smaller than its own gradient. The gradient is computed in the following way: initially, the sink has a gradient of 0; when a node has a gradient defined, it sends its gradient to its neighbors; when a node receives a gradient from a neighbor, it updates its own gradient if it detects that this neighbor is closer to the sink than itself.

We first notice that for a routing protocol based on gradient and for a random topology with one sink (located at one corner of the area), a node is likely to have more neighbors further away from the sink than closer to the sink. We estimate that this ratio is about two, which leads us to set for the size of R twice the maximum size of E. In order to determine the actual size of these lists, we performed 100 simulations over 10 random topologies (of average degree 8) for three different values of the traffic generation period P.

Figures 4 and 5 shows respectively the delivery ratio and the end-to-end delay as a function of the size of E, with $|R| = 2|E|$. In these two figures, we observe that regardless of the period P, the best size is two for E and four for R when combining the two criteria. These parameters are used in the following. Note that a history of six entries is realistic for the memory of sensor nodes.

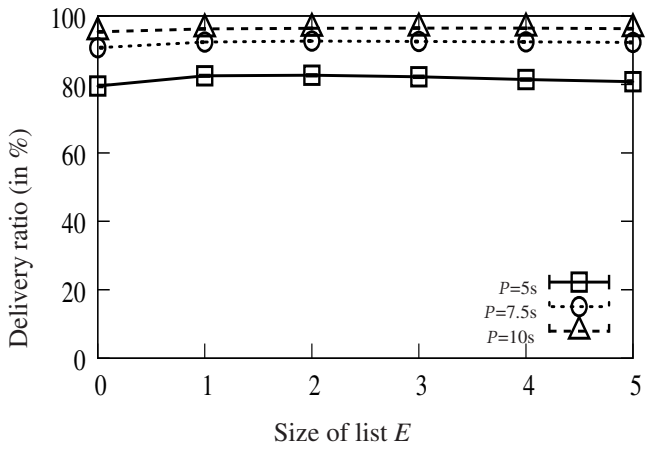

Figure 4. Impact of the size of list E of SLACK-MAC on the packet delivery ratio, with $|R| = 2|E|$ and a duty-cycle of 1%, where P is the traffic generation period.

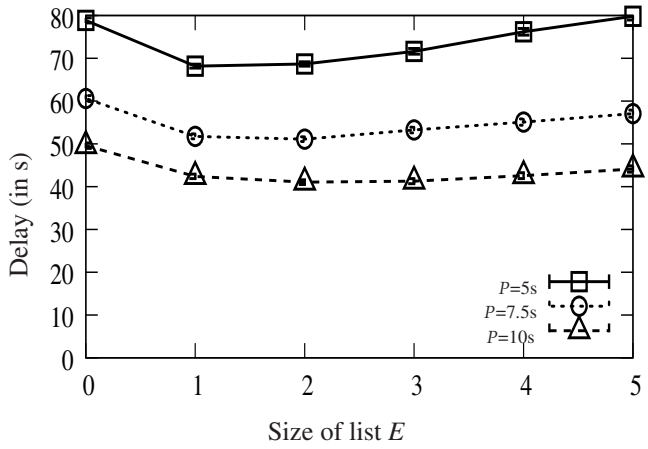

Figure 5. Impact of the size of list E of SLACK-MAC on the end-to-end delay, with $|R| = 2|E|$ and a duty-cycle of 1%, where P is the traffic generation period.

We also observed experimentally that on average, it takes about 12 cycles (60 seconds) for the nodes to fill E and about 50 cycles (250 seconds) for the nodes to fill R. This shows that

the convergence of the lists are fast and negligible compared to the life time of a node.

4. Results

In this section, we describe our simulation and experimentation results concerning SLACK-MAC.

4.1. Simulation results

In order to evaluate the performance of SLACK-MAC, we conducted several simulations to compare it with the synchronous MAC protocol of the standard IEEE 802.15.4 [9] and with the following asynchronous MAC protocols: X-MAC [21], RI-MAC [7], PW-MAC [8] and the protocol given in [26] (also called Basis protocol in the rest of the paper) on which SLACK-MAC is based.

Simulation parameters. Our simulations are performed using the network simulator NS-2 [27]. For all protocols, the transmission power is set to 0 dBm, and the selected propagation model is the shadowing model with a path loss of 2.74. In our settings, 30 sources perform periodic measurements and route data (in a multi-hop manner) to a single sink located at one corner of the network. Nodes have a duty-cycle of 1 % and the global cycle is 5 s (that is, nodes are active during A=50 ms every C=5 s), unless specified otherwise. For our simulations, we used 100 nodes randomly located on a topology of 170 m x 170 m with a transmission range of 30 m, which yields to a maximum number of hops of 7. All presented results are averaged over 100 repetitions per topology and each repetition lasted for 3600 seconds. In all our simulations for all tested MAC protocols (except for the standard IEEE 802.15.4 [9] that is based on ZigBee [28] and then uses a tree based routing protocol), we used the same gradient based routing protocol for routing data through hops until to reach the sink.

We first notice that synchronous MAC protocols are not adapted to deal with low duty-cycle. We compared SLACK-MAC with the synchronous MAC protocol of the standard IEEE 802.15.4 [9]. Then we compared our protocol with the following asynchronous MAC protocols: X-MAC [21], RI-MAC [7] and PW-MAC [8]. These protocols represent the main asynchronous duty-cycle MAC protocols. We also compared SLACK-MAC with the protocol presented in [26].

Discussion. Figure 6 and 7 show respectively the packet delivery ratio and the average delay of data packets as a function of the traffic generation period (from 5 seconds to 30 seconds), for the following protocols: X-MAC, RI-MAC, PW-MAC, the standard IEEE 802.15.4, the protocol of [26] and SLACK-MAC.

For the standard IEEE 802.15.4, BI and SD are set respectively to 5 seconds and 50 milliseconds. The delivery ratio increases from 12 % to 50 % and the average delay decreases from 163 s to 44 s. This low packet delivery ratio with the standard is due to the fact that the low duty-cycle generates a strong contention, as nodes are all synchronized.

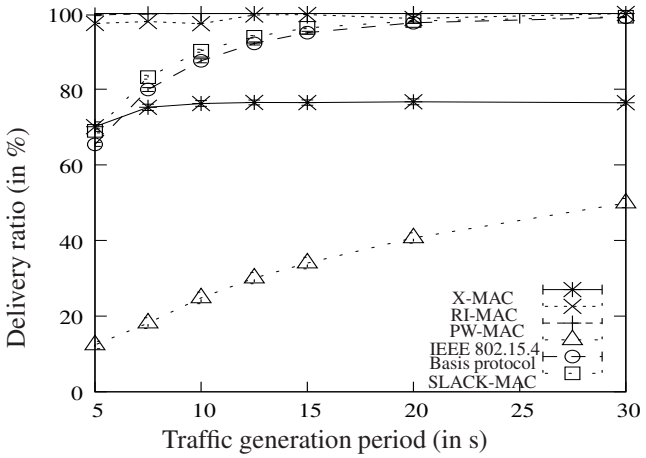

Figure 6. Delivery ratio as a function of the traffic generation period, for a duty-cycle of 1 %, and for several protocols.

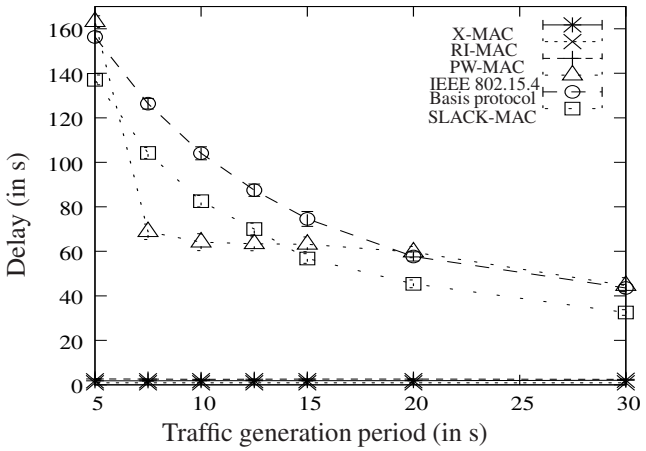

Figure 7. End-to-end delay of data packets as a function of the traffic generation period, for a duty-cycle of 1 %, and for several protocols.

This generates many collisions and causes overflows in nodes queues, causing a large packet loss ratio. It is important to note that in these results we do not count the cost of the synchronization because we assumed that all nodes are perfectly synchronized.

For the protocol X-MAC, the delivery ratio increases from 70 % to 76 % and the average delay is around 2 s, when the traffic generation goes from 5 s to 30 s. The loss of data mainly comes from the relatively high number of preamble frames and also from the sender has no knowledge of the successful reception of packets by the receiver. The low delay of 2 s is due to the fact that X-MAC does not set a fixed duty-cycle for each node: when a node has data to send, it stays active until it can send it.

For RI-MAC protocol, the delivery ratio increases from 97 % to 100 % and the average delay is around 1 s. These

results are due to the fact that RI-MAC does not set a fixed duty-cycle for each node. We also notice that RI-MAC reduces the occupation of the channel compared to X-MAC, which implies less collisions. Moreover, the senders in RI-MAC are informed by acknowledgment messages when a sent data is received.

For PW-MAC, the delivery ratio is always 100 % and the average delay decreases from 3 s to 2 s. As X-MAC and RI-MAC, PW-MAC does not set a fixed duty-cycle for each node. The delay with PW-MAC is larger than with RI-MAC because the senders do not remain active until the receiver wakes up. Indeed, the senders predict when the receivers wake up using a prediction mechanism. It allows this protocol to reduce its energy consumption, but induces a longer average delay.

For the protocol of [26], the delivery ratio increases from 65 % to 99 % and the average delay decreases from 156 s to 43 s. The delivery ratio is low when the traffic is high, because the duration of the common activities is not large enough to absorb the high number of messages in the network. However, this protocol has a high delivery ratio when the traffic is low despite a longer delay.

For SLACK-MAC, the delivery ratio increases from 68 % to 99 % and the average delay decreases from 137 s to 32 s. Globally, the delivery ratio is high and the delay is low compared to the protocol of [26]. Indeed, SLACK-MAC takes advantage of a mechanism similar to the protocol of [26], and allows more common activities between nodes thanks to the history mechanism. SLACK-MAC also remains completely dynamic with a probability of 1/3 of nodes choosing a random instant.

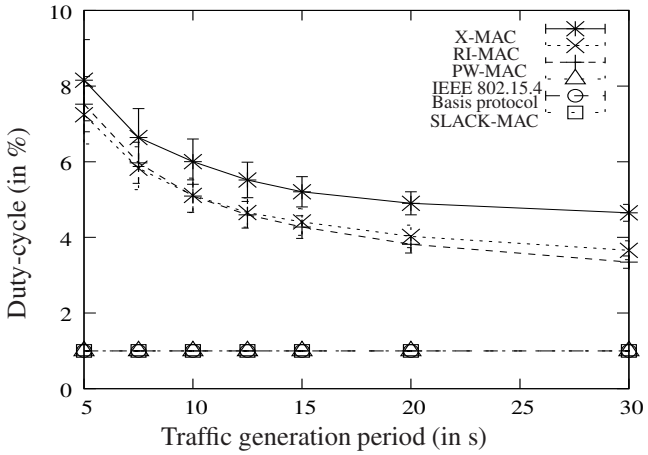

Figure 8. Duty-cycle as a function of the traffic generation period, for a duty-cycle of 1 %, and for several protocols.

Figure 8 shows the duty-cycle in percent for each protocol as a function of the traffic generation period (from 5 seconds to 30 seconds).

For ZigBee, the protocol of [26] and SLACK-MAC, the duty-cycle is set to 1 % and is fixed.

For X-MAC, the average duty cycle decreases from 8.15 % to 4.65 %, for RI-MAC from 7.24 % to 3.66 %, and for PW-MAC from 7.52 % to 3.34 % when the traffic generation period increases from 5 seconds to 30 seconds. Indeed, for X-MAC and RI-MAC the duty-cycle of node actually depends on the communication opportunities, as nodes having frames to send remain active until they can send their frames. PW-MAC implements a local synchronization, which causes many collisions when multiple transmitters simultaneously send their frames to the same receiver. So, the nodes can make a good prediction without being able to transmit their data. Thus, X-MAC, RI-MAC and PW-MAC yield larger duty-cycles than the other protocols.

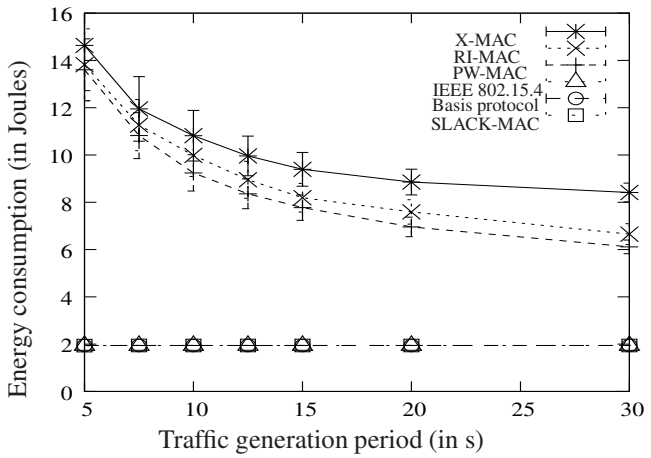

Figure 9. Energy consumption as a function of the traffic generation period, for a duty-cycle of 1 %, and for several protocols.

Figure 9 shows the average energy consumption per node in Joules for a period of 1 hour, as a function of the traffic generation period (from 5 seconds to 30 seconds) for the protocols X-MAC, RI-MAC, PW-MAC, the standard IEEE 802.15.4, the protocol of [26] and SLACK-MAC.

For the standard IEEE 802.15.4, the Basis protocol and SLACK-MAC, the average energy consumption is always below 2 J (because all nodes have a duty-cycle set to 1 %). For X-MAC, the energy consumption decreases from 15 J to 8 J. For RI-MAC, it decreases from 14 J to 7 J. For PW-MAC, it decreases form 14 J to 6 J. We notice that for X-MAC, RI-MAC and PW-MAC, the energy consumption is high because all nodes have to be active more than 1 % of the time when they have data frames to send, while nodes in IEEE 802.15.4, the Basis protocol and SLACK-MAC keep a constant duty-cycle.

These results show that the standard IEEE 802.15.4 is not adapted to low duty-cycles. Likewise, the asynchronous MAC protocols X-MAC, RI-MAC and PW-MAC for which nodes do not operate at a fixed duty-cycle for all nodes are not adapted to low duty-cycles, since nodes are consuming more energy. In these protocols we also see that the energy

consumption is not fairly spread among all the nodes. For instance, the maximum energy consumption for a node with a small traffic generation period of 5 s is 60 J for X-MAC, 61 J for RI-MAC and 45 J for PW-MAC. When the period generation is 30 s, the maximum energy consumption for a node is 18 J for X-MAC, 23 J for RI-MAC and 13 J for PW-MAC.

Finally SLACK-MAC provides a better trade-off in terms of energy consumption, delivery ratio and average delay for environmental monitoring applications that require long network life time.

4.2. Experimentation results

We implemented SLACK-MAC on TelosB motes, using the NesC language for the TinyOS operating system [29]. The topology used for our experimentations is shown in Figure 10. It consists of a set of 20 nodes and one sink deployed on a table in our laboratory. The transmission power is set to the minimum value, which results into an actual communication range of ten to twenty centimeters. This results into a maximum of four hops from any node to the sink. There are six sources in total: two sources are four hops away from the sink, three sources are three hops away from the sink, and one source is two hops away from the sink. The radio environment is also polluted by WiFi communications, hence there are many interferences. We set the duty-cycle to 1%, and the traffic generation period goes from 5 seconds to 5 minutes.

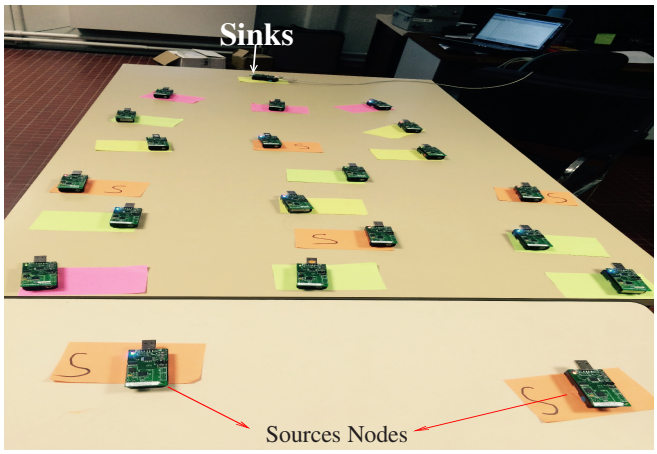

Figure 10. Experimentation topology.

Figure 11 depicts the delivery ratio as a function of the traffic generation period (using a logarithmic scale) for the Basic protocol and SLACK-MAC. Note that the left part of the figure shows heavy traffic (with small traffic generation period), while the right part of the figure shows low traffic (with large traffic generation period). The delivery ratio increases as the traffic generation period increases, which is expected. It can be seen that the delivery ratio with SLACK-MAC is above the delivery ratio with the basic protocol (with a gap of up to 10%). This is because SLACK-MAC is able to increase the probability of having common activity periods between nodes, and thus sources are able to deliver more packets. When compared to the simulation results, the delivery ratio appears to be very low. This is due to the fact that the experimental topology is dense and yields collisions. However, when the traffic generation period becomes large, the achieved delivery ratio becomes very high (it is close to 100%), it is because message queues are not often full with this traffic.

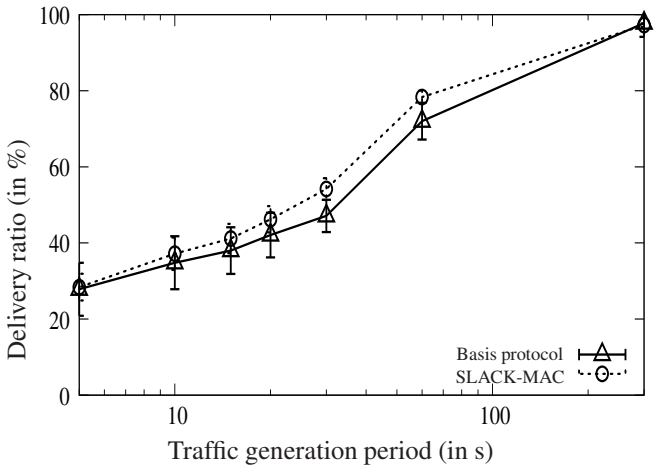

Figure 11. Delivery ratio as a function of the traffic generation period, for a duty-cycle of 1%, and for the Basic protocol and SLACK-MAC.

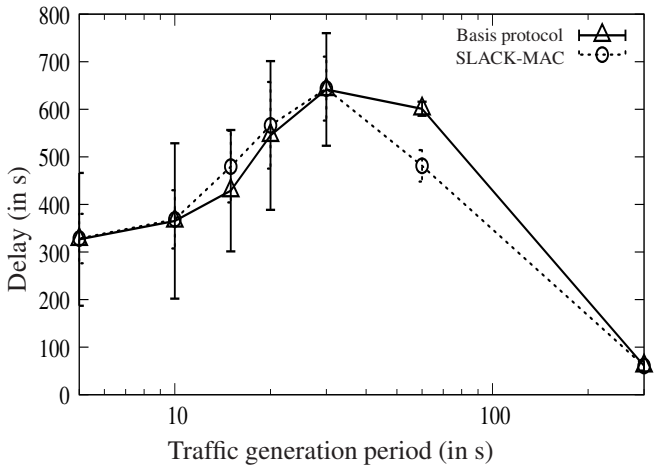

Figure 12. End-to-end delay of data packets as a function of the traffic generation period, for a duty-cycle of 1%, and for the Basic protocol and SLACK-MAC.

Figure 12 shows the average delay as a function of the traffic generation period (using a logarithmic scale). The delay for both SLACK-MAC and the basic protocol increases when the traffic generation period varies from 5 s to 30 s, and decreases when the traffic generation period varies between 30 s and 300 s. The worst delay is achieved at 30 s, with a value of about 600 s. It is due to two factors: the traffic

generation period is small enough to generate a heavy traffic for the protocol, and the losses are large enough (about 50%) to artificially reduce the delay by dropping frames that would have a long delay. The fact that the delay with SLACK-MAC is slightly larger than with the basic protocol, for traffic generation periods of 20 s and 30 s, is due to the fact that the size of the lists are configured for a sparser topologies. For this dense topology, two smaller lists would bring better performances in term of delay, but would decrease the delivery ratio.

5. Conclusion

In this paper, we proposed the SLACK-MAC protocol for WSNs with very low duty-cycles of 1% or less. Initially, nodes in SLACK-MAC activate their radio module randomly and independently, and build a history of successful communications. The history is used to determine the next activation time, which results into a self-adaptive behavior. We show that SLACK-MAC reaches a good behavior with a limited history size of only six entries. Only few activity cycles are needed to fill the memory lists. Then, we compare SLACK-MAC with existing protocols in terms of frame loss, end-to-end delay and consumed energy. We show that our low-cost protocol is able to significantly improve the performance of existing protocols. We also implemented SLACK-MAC on real sensor node hardware and show that it requires a small memory footprint. We use this implementation to perform experiments, and we show that our experimentation results are consistent with our simulation results.

Acknowledgement. This research was partially supported by the "Digital Trust" Chair from the University of Auvergne Foundation.

References

[1] W. Geoffrey, J. Jeff, R. Mario, L. Jonathan, and W. Matt, "Monitoring volcanic eruptions with a wireless sensor network," in *European Workshop on Wireless Sensor Networks (EWSN'05)*, 2005.

[2] S. Robert, M. Alan, P. Joseph, A. John, and C. David, "An analysis of a large scale habitat monitoring application," in *ACM Sensys*, 2004, pp. 214–226.

[3] J. Hart and K. Martinez, "Environmental sensor networks: A revolution in the Earth system science?" *Earth Science Reviews*, vol. 78, no. 3, pp. 177–191, 2006.

[4] T. Nagayamaa, M. Ushitab, and Y. Fujinoa, "Suspension bridge vibration measurement using multihop wireless sensor networks," in *East Asia-Pacific Conference on Structural Engineering and Construction (EASEC)*. Elsevier, 2011, pp. 761–768.

[5] A. T. Aby, A. Guitton, P. Lafourcade, and M. Misson, "SLACK-MAC: Adaptive MAC protocol for low duty-cycle wireless sensor networks," in *7th International Conference on Ad Hoc Networks (Adhocnets)*, 2015.

[6] K. Altisen, S. Devismes, R. Jamet, and P. Lafourcade, "SR3: Secure resilient reputation-based routing," in *IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2013, Cambridge, MA, USA, May 20-23, 2013*. IEEE, 2013, pp. 258–265. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/DCOSS.2013.33

[7] 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 *ACM Sensys*, 2008.

[8] L. Tang, , Y. Sun, O. Gurewitz, and D. B. Johnson, "PW-MAC: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1305–1313.

[9] IEEE 802.15, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs)," ANSI/IEEE, Standard 802.15.4 R2006, 2006.

[10] G. Lu, B. Krishnamachari, and C. Raghavendra, "An adaptive energy-efficient and low-latency MAC for tree-based data gathering in sensor networks," in *Wireless Communications and Mobile Computing*, vol. 7, September 2007, pp. 863–875.

[11] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, "DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," in *ACM MobiHoc*, 2008.

[12] L. Choi, S. H. Lee, and J. A. Jun, "SPEED-MAC: Speedy and energy efficient data delivery MAC protocol for real-time sensor network applications," in *Proceeding of the International Conference on Communications (ICC)*, May 2010, pp. 1–6.

[13] S. Wen-Zhan, H. Renjie, S. Behrooz, and L. Richard, "Treemac: Localized TDMA MAC protocol for real-time high-data-rate sensor networks," in *Pervasive and Mobile Computing*, 7 December 2009, pp. 750–765.

[14] D. I. Ozlem, V. H. Lodewijk, J. Pierre, and H. Paul, "MC-LMAC: A multi-channel MAC protocol for wireless sensor networks," in *Ad Hoc Networks*, vol. 9. Elsevier, 2011.

[15] Y. Z. Zhao, C. Y. Miao, and M. Ma, "An energy-efficient self-adaptive duty cycle MAC protocol for traffic-dynamic wireless sensor networks," in *Wireless Personal Communications*, 2012, pp. 1287–1315.

[16] O. Hoon and H. Trung-Dinh, "A demand-based slot assignment algorithm for energy-aware reliable data transmission in wireless sensor networks," in *Wireless Networks*, 2012.

[17] X. Deng and Y. Yang, "Cluster communication synchronization in delay-sensitive wireless sensor networks," in *IEEE International Conference on Distributed Computing in Sensor Systems*, 2013, pp. 36–43.

[18] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. B. Srivastava, and D. Ganesan, "Estimating clock uncertainty for efficient duty-cycling in sensor networks." *IEEE/ACM Transactions on Networking*, 2009.

[19] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *ACM Sensys*, November 2004.

[20] A. El-Hoiydil, J.-D. Decotigniel, and J. Hernandez, "WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks," in *Algorithmic Aspects of Wireless Sensor Networks*, ser. LNCS, vol. 3121. Springer Berlin / Heidelberg, 2004, pp. 18–31.

[21] M. Buettner, Y. Gary, V., E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *ACM Sensys*, November 2006.

[22] Z. Li, M. Li, and Y. Lui, "Towards energy-fairness in asynchronous duty-cycling sensor networks," in *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, April 2014.
[23] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. B. Johnson, "ADB: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks," in *ACM Sensys*, 2009, pp. 43–56.
[24] L. Tang, Y. Sun, O. Gurewitz, S. Du, and D. B. Johnson, "EM-MAC: A dynamic multichannel energy-efficient MAC protocol for wireless sensor networks." ACM, 2011, p. 23.
[25] H. Tang, C. Sun, Y. Liu, and B. Fan, "Low-latency asynchronous duty-cycle MAC protocol for burst traffic in wireless sensor networks," in *IWCMC'13*, July 2013, pp. 412–417.
[26] A. T. Aby, A. Guitton, and M. Misson, "Asynchronous blind MAC protocol for wireless sensor networks," in *IWCMC (International Wireless Communications and Mobile Computing Conference)*, 2014.
[27] "Network simulator 2," 2002, http://www.isi.edu.nsnam/ns.
[28] ZigBee, "ZigBee Specification," ZigBee Standards Organization, Standard ZigBee 053474r17, January 2008.
[29] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *In Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104.