Technical Report : ContikiMAC performance analysis

Mathieu Michel and Bruno Quoitin

Computer Science Department, University of Mons, Belgium
firstname.lastname@umons.ac.be

Abstract. This paper evaluates the performance of X-MAC and ContikiMAC. Both protocols use different mechanisms to send a packet to a neighbour node. To send a data packet to a neighbour, ContikiMAC repeatedly send the full data packet until it is acknowledged while X-MAC send small "probe" frames to advertise the destination of the incoming transmission before sending the packet. The objective of this study is to understand why ContikiMAC is more efficient than X-MAC despite an intuitively more consumptive transmission procedure. We have then compared ContikiMAC against an updated version of X-MAC which has been designed with the goal to obtain a more objective comparison. Both protocols performances are evaluated in terms of number of retransmissions, latency, packet delivery ratio (PDR) and duty-cycle. Our study reveals that the better efficiency of ContikiMAC can be attributed to two specific mechanisms: the "fast sleep" optimization shortening the wake-up period and a transmission procedure more efficient. The combination of both mechanisms helps ContikiMAC to achieve a better PDR with a reduced latency and a drastically lower energy consumption than X-MAC.

1 Introduction

Several MAC duty-cycle protocols [1] have been proposed during the last decade to address specific WSNs requirements and constraints such as a low energy consumption linked to battery operated nodes or a limited bandwidth. Radio Duty-Cycle (RDC) MAC protocols try to reduce the energy consumption by allowing a node to keep its radio-transceiver off most of the time. This allow a node to avoid to keep the radio on unnecessarily, i.e when not involved in any transmission. To limit this situation called idle-listening, a RDC MAC protocol forces every node to periodically switch its radio transceiver between short active (listen) periods and long inactive (sleep) periods. The duty cycle represents the ratio between the length of the active period and the interval between two successive wake-ups (Fig.1).

Among the duty-cycle MAC RDC protocols the two following categories can be distinguished: the asynchronous and the synchronous protocols. Synchronous duty-cycle protocols force the nodes to synchronize their duty-cycles while asynchronous protocols let every node free to determine it’s own independent sleep schedule. By removing the need to create, communicate and maintain
common active period, the asynchronous protocols benefit from a lower implementation complexity making them particularly energy efficient in case of low bandwidth networks [1]. Several asynchronous protocols have been developed such as LPL [2], B-MAC [3], WiseMAC [4] and many others. The availability of X-MAC [5] and ContikiMAC [6] implementation in Contiki [7] make them two of the most popular duty cycle protocols.

ContikiMAC, the most recent, is used by default in Contiki. ContikiMAC and X-MAC differ in their transmission procedures: while X-MAC uses a stream of short-sized strobes to advertise its transmissions, ContikiMAC is using a more consumptive transmission procedure which consists of repeatedly data packet transmissions until acknowledgement. Surprisingly ContikiMAC seems to outperform X-MAC. Indeed, the literature presents ContikiMAC as able to reduce drastically the energy consumption and to obtain a better latency at the expense of strict timing constraints and noise sensibility [6,8,9].

To better understand the efficiency of ContikiMAC we performed controlled experiments in Cooja [10], the Contiki simulator. The objective is to quantify the gain of ContikiMAC against X-MAC. To reach that objective we first identified the main mechanisms involved in ContikiMAC and then designed a new version of X-MAC that include some of the ContikiMAC mechanisms. This evaluation against an advanced version of X-MAC allows to focus on the main differences between both protocols: the transmission and the wake-up procedures.

This paper, which extends the results presented in [11], is organized as follows. We first introduce X-MAC and ContikiMAC in Section 2. Then, we describe our test methodology and the enhancements brought to X-MAC in Section 3. We provide our experimentations results in Section 4. Finally we conclude and discuss possible future works in Section 5.
2 X-MAC and ContikiMAC

X-MAC and ContikiMAC are two asynchronous MAC RDC protocols which differ mainly by their approach regarding, on one side, the data packet transmission procedure and on the other side the wake-up procedure of the nodes.

2.1 Packet transmission

**X-MAC** The X-MAC unicast transmission process derives from the preamble sampling technique used by X-MAC predecessors B-MAC \([3]\) and LPL \([2]\). X-MAC outperforms those protocols by replacing the use of a long extended preamble with a stream of short preamble packets used to advertise transmissions (Fig 2). This stream of strobes is stopped at the reception of a strobe-ACK allowing the transmission of the data packet (Fig 2). Compared to B-MAC or LPL, this methodology leads to better performance resulting from an overhearing and latency reduction.

1. The **overhearing reduction** is achieved by embedding the destination address in the strobes. Checking this destination ID enables a node to determine the destination of the packet and then eventually return to sleep if not concerned.
2. The **latency reduction** is achieved during unicast transmission by inserting a short delay between each strobe to allow the destination neighbour to send a strobe-ACK. Once received, the stream of strobes is interrupted and the data packet is sent without further delay. The use of strobe-ACK doesn’t apply for broadcast communication.

This strobe based approach is also used by X-MAC for the broadcast mechanism. Every broadcast packet is preceded by a stream of strobes. This stream is slightly longer than a wake-up interval so each potential receiver may wakes up at least once. To indicate broadcast packet, strobes, which don’t need to be acknowledged any more, do not embed a destination ID to force each node within transmission range to stay awake until the reception of the broadcasted packet.

**ContikiMAC** To send a packet, a node running ContikiMAC repeatedly sends the full data packet until an ACK is received (Fig 3) or the transmission times out after a duration equal to a wake-up interval. The packet destination field allows a node to reduce overhearing by going back to sleep if not concerned. In the opposite case the node acknowledges the correct reception of the packet. The transmission is then stopped by the destination.

2.2 Wake-up procedure

**X-MAC** X-MAC forces each node to wake up at regular interval for a short ”active” period to sense the medium looking for potential incoming packet. If no incoming transmissions are detected the node goes back to sleep until the
Fig. 2. X-MAC uses a stream of strobes to advertise a transmission.

Fig. 3. ContikiMAC sends repeatedly the data packet until an ACK reception.

next scheduled wake-up. This sleep period is called the "passive" period which together with the active period forms the duty-cycle period. The length of an active period is commonly 5 or 10% of the full duty-cycle period (125 ms by default within Contiki [7]). The active period can be extended (beyond the cycle length) if the node is still involved in transmissions and the end of this one.

ContikiMAC A node running ContikiMAC wakes up periodically using a fast sleep optimization. This methodology consists in quickly check the medium with two successive CCAs (Clear Channel Assessment) which determine if there is possible incoming radio transmissions based on the radio signal strength. If the signal strength is below a specified threshold, the medium is considered as quiet and the node goes back to sleep until the next wake up. If the medium is busy the node stays awake trying to determine if this signal strength is related to a packet transmission or just noise. In case of noise the node can go back to sleep,
if not it stays awake to receive the incoming packet and identify the destination of the packet. Depending on the identification result the node goes back to sleep or stays awake to handle the transmission.

Given the use of this fast sleep optimization the timing of the wake-up phase is critical (Fig. 4). ContikiMAC ensures this timing through specific constraints.

Using the following notations:

1. $T_i$ interval between each packet transmission
2. $T_r$ time required for the channel assessment
3. $T_c$ interval between two CCA
4. $T_a$ time needed to send an ACK
5. $T_d$ time required to receive the ACK.
6. $T_s$ transmission time of the shortest packet.

Those constraints may be expressed as:

- the interval between two transmissions $T_i$ must be bigger than the time needed to send and receive the ACK.
- to ensure that at worst two CCAs are enough to detect a transmission, the interval $T_c$ between two CCAs must be bigger than the interval between two transmissions $T_i$.
- to avoid that a full transmission occurs between two CCAs, the shortest packet size must not be larger than $t_r + t_c + tr$

Those constraints may be resumed by $T_a + T_d < T_i < T_c < T_c + 2T_r < T_s$

![Fig. 4. timing constraints in ContikiMAC](image)
This fast sleep optimization show less interest under X-MAC given the time interval between two strobes ($T_{sInt} = 3.9$ ms) is by default 4 times bigger than the interval between two ContikiMAC successive data transmissions ($T_i = 0.9$ms). A substantial increase of the number of successive CCAs needed to sense the medium would be required to cover an interval of time long enough to detect a transmission.

2.3 Additional features

The performances of ContikiMAC are also improved due to two additional mechanisms:

1. The phase-lock mechanism (as introduced by WiseMAC [4]) allows a node to learn the wake-up phase of a neighbour after a first unicast transmission. Once a node receives an ACK it knows that the destination was awake at this moment (Fig 5), using this time as reference the sender can start its successive data transmissions just before the supposed destination wake up (Fig 6). By allowing a node to achieve a transmission with on average only two packets (the first being used to announce the transmission), the phaselock mechanism decreases the time and the energy spent in TX mode. Another benefit of this shortened stream of data transmissions is to reduce the channel utilization and then the risk of collisions.

2. Before starting a transmission, ContikiMAC uses a collisions avoidance mechanism which checks the availability of the channel with several successive CCAs. This is done in such a way that the interval of time between two successive data packet transmissions $t_i$ is covered. The lack of activity during this period informs a node that apparently no transmissions are occurring. Those would have cause collisions. The strict timing constraints in ContikiMAC allows to ensure the efficiency of the mechanism.

2.4 Protocols implementation in Contiki

Contiki offers implementation for both ContikiMAC (used as default MAC protocol) and X-MAC. Compared to [5] the Contiki X-MAC implementation offers several enhancements [12]:

1. **Broadcast**: The broadcast mechanism uses an alternative approach, similar to ContikiMAC, which consists in repeated transmissions of the data packet. This broadcast mechanism is proposed by default in Contiki. The number of repetitions must be defined in such a way that each potential receiver has the possibility to wake up to catch the packet. This number must be sufficient enough to cover a full wake-up interval.

2. **Encounter**: The encounter optimization mechanism allows a transmitter to reduce the size of the stream of strobes after learning the receiver’s wake-up schedule. This information is collected through the strobe-ACKs reception clock time. Considering a potential clock de-synchronization, on average a
stream of two strobes is enough to contact the destination \[13\]. This allows to reach similar results than ContikiMAC phaselock which need two packets to achieve the transmission.

3. **Reliable data transmission:** Contiki X-MAC uses a flag which is set to indicate the need for a data packet acknowledgement. For such packets the transmitter stays awake waiting for the ACK. This ACK can be sent without requiring any strobes since the transmitter is already awake. Remember that \[5\] consider the data packet acknowledgement as optional.

4. **Streaming:** Using streaming a node doesn’t need to send a strobe before each packet transmission in a same data flow. This mechanism is equivalent to the *burst mode* in ContikiMAC \[6\].
3 Methodology

This section describes the methodology used to compare ContikiMAC and X-MAC. For the purposes of this comparison we have performed some experiments by simulation considering the following metrics: the number of retransmissions (linked to a collision or lack of acknowledgement), the latency, the packet delivery ratio (PDR) and the duty cycle. This last one provides an information regarding the energy consumption by evaluating the time spent by a node in the following states: listen, RX and TX.

Our simulations are performed with the Contiki 2.6 Cooja Simulator [10]. Cooja is a Java-based event-discrete simulator. It allows to perform simulations at the network and firmware levels. This means that simulations can be run with almost the same code as deployed on real platforms. In addition Contiki offers several protocols implementation, among these we can find X-MAC, ContikiMAC and RPL.

The experiments are based on the Zolertia Z1 platform [14] as it is supported by Cooja. The default configuration of ContikiMAC and X-MAC have been kept with the following modifications:

1. For ContikiMAC, the number of CCA used by the collision avoidance mechanism has been changed from 6 to 2, which is enough to detect a transmission given a packet interval \( t_i = 0.4ms \) and an interval between two CCAs \( t_c = 0.5ms \).
2. For X-MAC, the broadcast mechanism used the "strobe based" approach in place of the default "data repetition" approach (cf Subsection 2).
3. The burst (ContikiMAC) and streaming (X-MAC) modes have been disabled to focus on standard transmissions without benefit from the fact that the destination is already awake after the reception of a previous packet in the same data flow.

All protocols use a wake-up interval of 125ms (default value within Contiki). X-MAC used the default (as specified in Contiki) 5% duty cycle. Due to the fast sleep optimization the size of the duty-cycle within ContikiMAC is "dynamic" and so cannot be directly specified.

3.1 Protocols under evaluation

To make a fair comparison focusing on the main differences between X-MAC and ContikiMAC it seems essential to equilibrate the protocols by implementing the phaselock and the CCA collision avoidance mechanisms in X-MAC.

Phase lock While the original X-MAC protocol doesn’t learn the wake-up schedules of the nodes, the Contiki X-MAC protocol implementation offers an
enhancement called *encounter optimization* which helps a node to learn the wake-up schedules of its neighbours. A drawback of that implementation is that it involves a blocking wait before meeting an encounter: if a node wants to send a data packet to a node for which the wake up schedule is established, the sending process is paused until the destination’s wake up. This prevents the node to handle other tasks. ContikiMAC avoids this by using a timer which expires a little bit before the destination’s wakeup allowing the node to send the data at the best time. To obtain a fair comparison between both protocols we disabled this *encounter optimization* and implementing instead a phase-lock mechanism similar to that of ContikiMAC. When waiting for a node’s wake up the node stays active in respect with its own phase schedule.

**Collision avoidance** The Contiki X-MAC implementation doesn’t provide a real collision avoidance mechanism: no Clear Channel Assessment (CCA) is performed before a transmission. At most X-MAC uses a flag, set up when a strobe has been received, to indicate that a transmission has been detected previously, if so X-MAC considers that the channel is busy and postpones the transmission, whether or not the node is targeted by the transmission advertised by the strobe. The major drawback of this implementation is that a sleeping node can’t set or clear this flag. This flag is only cleared if the node detects a data packet or reset at the next wake-up period. The collision avoidance mechanism has been adapted to use a CCA based approach as used in ContikiMAC. Before starting the transmission, a node checks if it is targeted by a transmission (a strobe addressed to itself has been received earlier), then another check ensures that the radio isn’t actually receiving a packet. Once those two checks passed, just before starting the strobes transmission, the channel availability is ensured by a bunch of successive CCAs. We have computed the number of CCAs in such a way that the interval between two strobes transmissions has been covered and then to avoid concurrent streams of strobes co-existing.

The addition of both phaselock (P) and CCA collision avoidance (C) mechanisms to the default X-MAC implementation leads to a new “advanced” X-MAC implementation: **X-MAC-CP** This version is compared with the ”basic” version of **X-MAC** (as proposed by Contiki) and **ContikiMAC**. In order to quantify the benefits of the addition of each mechanism, two additional versions under which those functionalities have been implemented independently are evaluated: **X-MAC-C** (CCA collision avoidance) and **X-MAC-P** (phaselock). As a result, the 5 protocols listed in Tab.1 are evaluated.

### 3.2 Network topologies

Two kind of scenarios are used to compare the protocols:

1. Small topologies with static routing.

---

\(^1\) To the best of our knowledge, the results of that extension have not been validated!
|                  | CCA Col. Av. | Phaselock |
|------------------|--------------|-----------|
| ContikiMAC       | X            | X         |
| X-MAC            |              |           |
| X-MAC-C          | X            |           |
| X-MAC-P          |              | X         |
| X-MAC-CP         | X            | X         |

Table 1. Summary of the protocols under evaluation

2. Large topologies using an RPL collect application.

In a first step, we compare the protocols on regular topologies using static routing. The size, density and traffic intensity can be scaled arbitrary. The test topology consists of one "central" node C surrounded by N neighbours. Among those N neighbours, we create N/2 couples Source-Destination (as illustrated by the green and yellow nodes on Fig. 7). N vary from 2 to 14 neighbours by step of 2. All neighbours are fitted with a default route to the central node. So if a node S wants to send a data to a node D this involves a two-hops transmission via the central node C.

![Fig. 7. Density/bandwidth test topology with 14 neighbours. The effective transmission and interference ranges of node 9 are respectively represented by the green and grey disks.](image)

Each test topology is evaluated with three different levels of transmission rate. Those three levels are obtained by varying the average interval between two application data transmissions. The lower this interval, the higher the transmission rate.
rate. The three levels of interval used for the experiments are respectively for the high, moderate and low tx rates: 5, 10 and 15 seconds. The results are averaged by 25 runs of the experimentation.

Second we performed simulations a in more realistic setup. For those scenarios the nodes are running a collect-to-sink application forcing each node to send periodically data packets to the sink. RPL [15], a routing protocol pushed by IETF, is used to compute the route dynamically. The topology contains 49 nodes including a sink and 48 sources. At every moment 5 nodes are "active" and 43 "passive". A passive node doesn’t send any application layer packet but is allowed to forward the active nodes traffic. Each MAC protocol allow a maximum of 4 retransmissions before dropping the packet.

The nodes are positioned on a square grid on a random manner: $x$ and $y$ coordinates are picked randomly according to a uniform random variate. The sink is always the node at the top left-corner (node 1 on Fig 8) while the 5 "active" nodes are randomly selected.

![Fig. 8. The 49-nodes nodes-to-sink topology. The effective transmission range and the interference range are respectively represented by the green and grey disk.](image)

The five MAC protocols are evaluated with the default Contiki RPL configuration and the results are averaged over 100 runs, each one with a new random deployment.
4 Results

4.1 Density and bandwidth evaluation

Our preliminary tests, involving both density and bandwidth variation on a small topology, led to the conclusion that ContikiMAC outperforms clearly X-MAC in regards to latency and energy consumption.

**Number of retransmissions** The latency is evaluated by the average end-to-end transmission time which is the time needed by a node A to send a packet to a node B. This time includes the retransmissions needed in case of collision or lack of acknowledgement. The bigger this number of retransmissions, the higher the average end-to-end transmission time.

A retransmission is triggered in the following cases:

1. The collision avoidance mechanism post-pones a transmission to avoid a collision.
2. A collision has been detected and the two, or more, transmissions involved have to been re-planned.
3. A lack of acknowledgement has been stated.

Except in case of node failures, the lack of acknowledgement can be caused by:

1. The occurrence of an undetected collision. This issue can be related to the "hidden terminal" issue when an interfering node is within destination but not source range.
2. Two nodes trying to transmit simultaneously, making the collision avoidance mechanisms useless given both nodes start sending packets at the same time (Fig 9).

![Fig. 9](image)

**Fig. 9.** In case of two simultaneous transmissions, the CCA collision avoidance mechanism can be powerless.
Fig. 10 illustrates the average number of transmissions for three different TX rates (high, moderate, low). Fig 10 clearly shows that ContikiMAC, despite sending full data packet, has the lowest number of retransmissions. The highest transmissions count of the default X-MAC protocol can be attributed to

1. the lack of collision avoidance mechanism
2. a higher channel use due to the lack of phaselock mechanism.

Sensing the channel prior to starting the transmission procedure, allows to avoid some collisions. The use of the CCA based collision avoidance mechanism in X-MAC-C decreases considerably the number of retransmissions, due to collisions, compared to default X-MAC. When a collision occurs, two retransmissions (one for each transmission involved in the collision) have to be re-planned, while a node detecting a collision has just to re-planned its own transmission.

A lower average number of retransmissions than with XMAC-C can be achieved by reducing the channel use via the phase-lock mechanism implemented in X-
MAC-P. But same if this average number of retransmissions is lower in X-MAC-P compared to X-MAC-C, Fig 11 shows that X-MAC-P achieves a lower PDR. This is explained by the fact that despite a reduced channel use which decreases the probability of a collision, nothing prevents two concurrent transmissions from a potential collision. A collision which can occur if both transmissions target the same destination or if both destinations have close wake-up schedules. So considering two destinations nodes in the same range with very closed schedules, the transmissions targeting them will collide and collide again until the maximum number of retransmissions is reached and the transmission dropped.

Reducing the probability of a collision by the combination of both phase-lock (reducing channel use) and CCA-based collision avoidance mechanisms allows X-MAC-CP to get similar results to ContikiMAC except at high transmission rates. The increase of transmissions count at high TX rates is due to the fact that X-MAC stops the transmission procedure as soon as the data packet is sent. This decision is unrelated to a data-ACK\(^2\) reception as for ContikiMAC. So if the transmission packet failed (which is more likely with a high transmission rate) X-MAC needs to reschedule a transmission. The difference related to the number of retransmissions between (default) X-MAC and ContikiMAC has been however considerably reduced.

Latency

Even if with a higher density and a higher transmission rate (Fig 12), ContikiMAC tends to performs better than X-MAC. The combination of phase-lock and CCA-based collision avoidance mechanisms allows X-MAC-CP to decrease the average end-to-end TX time compare to the default X-MAC (Fig 12).

From Fig 12 it appears that at very low density X-MAC and X-MAC-C have a lower latency than X-MAC-CP and X-MAC-P. This is explained by an offset in calibration of the phaselock mechanism. This offset is caused by the Contiki implementation of the phaselock mechanism. Within Contiki, a sender records the wake-up schedule of a neighbour as the timing of reception of a data-ACK (ContikiMAC) or strobe-ACK (X-MAC). This methodology involves an unavoidable offset with the effective wake-up schedule. The minimum offset is around 7 ms for both X-MAC and ContikiMAC. In case of idle listening a destination could answer at a time distant from the effective wake-up period increasing the size of this offset leading to a biased calibration of the phaselock timer. When a node wants to start a transmission, this one will then be delayed despite the fact that the destination is perhaps already awake. Lack of phaselock involves direct transmission without delay. It explains why X-MAC-C tends to be faster than X-MAC-CP.

---

\(^2\) As described in Subsec 2.4 the Contiki X-MAC implementation uses data-ACK which has been considered as optional in 5.
Fig. 11. Average packet delivery ratio.

Thanks to the fast-sleep optimization which decreases the idle-listening ContikiMAC is more susceptible to reduce the offset in the calibration of the phaselock mechanism\(^3\). Compared to X-MAC-P and X-MAC-CP, ContikiMAC compensates quickly the lack of precision of the phaselock mechanism by the reduction of retransmissions when the density increases.

This better probability to reduce the offset in the phaselock calibration explains partly that ContikiMAC performs better than the X-MAC protocols in terms of latency. Another reason for the faster transmission procedure of ContikiMAC is that thanks to the phaselock mechanism, and excluding collisions, ContikiMAC doesn’t need more than two data packets transmission to send successfully a data packet, while X-MAC needs two strobes and one data packet transmission. Which involves that, once the destination’s wake-up schedule learned, a hop-to-hop transmission is achieved by:

1. ContikiMAC in 7.9 ms = \( T_d + T_{sData} + T_d \)

\(^3\) In the best case, when ContikiMAC can contact the destination with no more than 2 strobes, this offset is 7ms
2. X-MAC in 9.7 ms = $T_s + T_{sInt} + T_s + T_{bfDat} + T_d^4$

With the following parameters:

- a strobe transmission time $T_s = 0.7$ ms (X-MAC)
- a data packet transmission time $T_d = 3.5$ ms (X-MAC and ContikiMAC)
- a time interval between two strobes $T_{sInt} = 3.9$ ms (X-MAC)
- a time interval between two data packets $T_{sData} = 0.9$ ms (ContikiMAC)
- a time interval between a strobe successfully acknowledged and a data packet $T_{bfDat} = 0.9$ ms (X-MAC)

Given a wake-up interval of 125 ms and a 2-hop end-to-end transmission, the minimal time needed to achieved a end-to-end transmissions is 140.8 ms with ContikiMAC and 144.4 ms with X-MAC.

Despite a lower number of retransmissions it looks like X-MAC-P has a higher latency than X-MAC-C. This is explained by the fact that in case of collision

\footnote{With the Contiki X-MAC default settings and a X-MAC 5\% duty cycle. The intervals between strobes are computed dynamically by X-MAC depending on the duty cycle.}
detected (and avoided) the Contiki CSMA implementation plans a retransmis-
sion at the next wake-up schedule, while the lack of ACK involves a random
delay (proportional to a cycle length) using a linear back-off so that interval be-
tween the transmissions increase with each retransmit. While X-MAC-C avoids
a lot of collisions, X-MAC-P is more impacted by a lack of ACK. At higher
density and transmission rate, X-MAC-P misses more acknowledgements than
X-MAC-C (+/- 20%).

It can also be observed that at lower density and bandwidth the basic X-MAC
implementation has a lower latency than X-MAC-C. This is explained by the lack
of CCA avoidance mechanism which slightly delays the transmission to process
the CCAs. The lower number of retransmissions doesn’t compensate this delay
at low density.

Energy Fig 13 represents the average network duty cycle for the time spent by
each node in listen, RX and TX mode. Same if the energy consumption of
X-MAC-CP is quite important compared to ContikiMAC, the use of the phase-
lock mechanism decreases that consumption compared to X-MAC and is still
enhanced by the addition of the CCA-based collision avoidance mechanism. Same
if the duty-cyle for the four versions of X-MAC is fixed at 5% we can observe
bigger values in the figures. This is due to the fact that this Duty-Cyle can be
extended by X-MAC if a node is involved in any transmissions: for example if
the node detects a broadcast transmission it will stay awake until the delivery
of the packet, independently of its initial schedule.
Fig. 13. Average duty cycle for listen, tx and rx.

4.2 Topologies using RPL

Table 2 illustrates our experiments results when using RPL in larger topologies. The following metrics are included: the percentage of unicast transmissions which are actually retransmissions, the packet delivery ratio (PDR) and the end-to-end latency.

PDR Regarding PDR, X-MAC is clearly the weakest protocol. The X-MAC-CP protocol shows the results closest to ContikiMAC. The difference between both protocols is explained by the lower reliability of the X-MAC protocols. Regardless of a data-ACK reception, X-MAC stops a transmission procedure as soon as it sends the data packet after the reception of a strobe-ACK. In case of no reception of a data-ACK, a brand new retransmission has to be scheduled, while ContikiMAC waits the data-ACK to consider the transmission done and stop the transmission procedure.

Number of retransmissions Based on the results observed in Subsection 4.1, the number of transmissions needed to send successfully a single data packet is
| Protocol    | PDR(%) | ReTX (%)       | Latency (s)       |
|-------------|--------|----------------|-------------------|
| ContikiMAC  | 71.9   | 1.19 (+/-0.52) | 1.06 (+/-0.80)   |
| X-MAC       | 65.3   | 1.53 (+/-1.24) | 1.30 (+/-1.55)   |
| X-MAC-CP    | 68.7   | 1.26 (+/-0.67) | 1.17 (+/-1.06)   |
| X-MAC-C     | 68.1   | 1.32 (+/-0.92) | 1.05 (+/-1.08)   |
| X-MAC-P     | 67.2   | 1.27 (+/-0.73) | 1.13 (+/-0.98)   |

Table 2. Average number of transmissions needed to send successfully a single data packet, PDR and latency (+ std. dev.)

considerably dropped by the use of the CCA collision avoidance and phaselock mechanisms. This explains the significantly smaller number of retransmissions of X-MAC-C and X-MAC-CP compared to the genuine X-MAC. However ContikiMAC, which achieved a better PDR than X-MAC-CP, requires less retransmissions.

**Latency** As it appears in Tab.2, ContikiMAC is the protocol showing the lowest latency. This is coherent in regards with the results observed for the number of retransmissions. Fig 14 illustrates the average end-to-end transmission time noticed on the 100 runs of the simulation. Each box extends from the 25th to the 75th percentiles with the line and the circle respectively as the median and the average value. By reducing the number of retransmissions X-MAC-CP can enhance the latency compared to X-MAC but ContikiMAC still performs better thanks to the combination of a lower number of retransmissions with a faster transmission procedure as explained in Subsection 4.1.

**Energy** The level of energy consumed is depends on the activity of the nodes: sleeping, listening, transmitting or receiving. Tab 3 illustrates the average time spent by nodes in the three following states: LISTEN, TX and RX. The first state represents the fact that the node radio transceiver is on and waits for an incoming transmission, the transmitting state is the time spent to actually send packets (strobe or data) while the receiving state is the time during which a node is actually busy receiving packets.

The time spent in listening mode by X-MAC is significantly higher than with ContikiMAC. This was expected given the ContikiMAC fast sleep optimization shortens the wake-up period allowing nodes to reduce at their active period to the strict minimum. Regardless of which version of X-MAC is used, the difference between the listening modes remains considerable same if we can observe an enhancement when introducing the phaselock and collision avoidance mechanisms in X-MAC. These two mechanisms bring the following enhancements:

1. The reduction of the retransmissions number decreases the waiting time each potential receiver sent listening for a strobe-ACK.
2. The reduction of the channel use caused by the phaselock decreases the probability of overhearing.
Looking at the transmitting state, basic X-MAC without any enhancement spends a similar amount of time transmitting compared to ContikiMAC, despite a higher number of retransmissions. This result can be explained by the difference of both sending procedures: on one side X-MAC uses strobes before sending the data, on the other side ContikiMAC uses a more expensive methodology using repeated data transmissions. Introducing the phaselock mechanism and a CCA-based collision avoidance mechanism allows X-MAC-CP to spend half as much time transmitting than X-MAC.

Finally, the fact that X-MAC is more submitted to overhearing cases increases the time spent in receiving mode compared to ContikiMAC where a node can reduces its wake-up period.

The results show that although X-MAC-CP is less energy consumptive than X-MAC, ContikiMAC consumes drastically less energy than this advanced version of X-MAC.

Fig. 14. Average end-to-end transmission time.
|                | LISTEN Mode (%) | TX Mode (%)     | RX Mode (%)     |
|----------------|-----------------|-----------------|-----------------|
| ContikiMAC     | 1.39 (+/-0.39)  | 0.34 (+/-0.17)  | 0.06 (+/-0.06)  |
| X-MAC         | 6.78 (+/-1.37)  | 0.31 (+/-0.31)  | 0.18 (+/-0.11)  |
| X-MAC-CP      | 5.98 (+/-0.58)  | 0.16 (+/-0.12)  | 0.15 (+/-0.10)  |
| X-MAC-C       | 6.51 (+/-1.21)  | 0.28 (+/-0.27)  | 0.16 (+/-0.10)  |
| X-MAC-P       | 6.13 (+/-0.67)  | 0.18 (+/-0.13)  | 0.11 (+/-0.11)  |

Table 3. Average (+std. dev.) duty cycle statistics

5 Conclusion and future works

In this paper we have demonstrated that ContikiMAC is not only better than X-MAC because of the addition of a phase-lock mechanism and a collision avoidance mechanism. The main mechanisms of ContikiMAC outperforms the design choices of X-MAC. Those mechanisms are a faster and more reliable transmission process based on repeated data transmissions and a fast-sleep methodology optimizing the wake-up procedure by making it less energy consumptive. We have also contributed to make X-MAC more efficient, especially considering an important decrease of the number of retransmissions and a slight enhancement of the energy consumption. But despite those enhancements X-MAC stays clearly below ContikiMAC, especially in terms of energy consumption. This enhanced implementation could however be interesting to evaluate in case of topologies making ContikiMAC wake-up procedure suffer from ”false wake-up” issues caused by an inappropriate CCA threshold configuration (as described in [8]). By forcing a node to interpret noise on the channel as a possible transmission and then to wake-up unnecessarily, this issue makes the ContikiMAC wake-up procedure too sensitive involving an increase of the energy consumption. Regarding the issue linked to a non optimal implementation of the phaselock within Contiki and the resulting biased calibration of this mechanism, one way to enhance it, as well for X-MAC and ContikiMAC, could be to force each node to embed it’s exact wake-up schedule in the strobe or data ACK, this way the sender would be able to target the real wake-up period of the destination.

References

1. C. Cano et al., “Low energy operation in wsns: A survey of preamble sampling mac protocols,” in *Comput. Netw.*, doi:10.1016/j.comnet.2011.06.022, 2011.
2. J. Hill and D. Culler, “Mica: a wireless platform for deeply embedded networks,” 2002.
3. J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *SenSys’04*, Baltimore, November 2004.
4. A. El-Hoiydi and J.-D. Decotignie, “Wisemac: An ultra lowpower mac protocol for the downlink of infrastructure wireless sensor networks,” in *9th Symposium on Computers and Communication*, June 2004.
5. M. Buettner et al., “X-mac: A short preamble mac protocol for duty-cycles wireless sensor networks,” in Technical Report CU-CS-1008-96, May 2006.
6. A. Dunkels, “The contikimac radio duty cycling protocol,” in SICS Technical Report T2011:13, December 2011.
7. [http://www.contiki-os.org/] 2013.
8. M. Sha, G. Hackmann, and C. Lu, “Energy-efficient low power listening for wireless sensor networks in noisy environments,” in IPSN13, April 811, 2013, Philadelphia, Pennsylvania, USA, April 2013.
9. C. Pinola, “Evaluating the performance of synchronous and asynchronous media access control protocols in the contiki operating system,” in Technical Report 2012:12, 2012.
10. [https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja#The_COOJA_Simulator] 2013.
11. M. Michel and B. Quoitin, “Analyse des raisons de l’efficacité de contikimac,” in ALGOTEL 2014 – 16èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, Le Bois-Plage-en-Ré : France, 2014.
12. [https://github.com/contiki-os/contiki/wiki/Radio-duty-cycling#The_Contiki_XMAC] 2014.
13. [https://github.com/contiki-os/contiki/wiki/Radio-duty-cycling] 2013.
14. [http://zolertia.sourceforge.net/wiki/index.php/Mainpage:Contiki] 2013.
15. T. Winter et al., “Rpl: Ipv6 routing protocol for low-power and lossy networks,” March 2012.