Dependability Analysis of Asynchronous Radio Duty Cycling Protocols

Salih Serdar Guclu, Tanir Ozcelebi, Johan J. Lukkien
Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
E-mail: {s.s.guclu, t.ozcelebi, j.j.lukkien}@tue.nl

Abstract—**Radio duty cycling (RDC) is a method for making resource constrained Internet-of-Things devices more energy efficient. RDC protocols achieve energy efficiency by keeping their radio off most of the time. Asynchronous RDC (ARDC) forms a subclass in which devices are not synchronized. ARDC behavior may lead to reliability and predictability problems for broadcast and unicast message transmissions in the MAC layer when the parameters of the protocol are not carefully selected, which we demonstrate by concrete examples. We investigate the dependability issues of generic ARDC protocols and provide analytical results for parameter settings that enhance dependability, while maintaining performance of the ARDC protocols in terms of energy efficiency and link delay. We also provide an analytical comparison of two widely accepted protocols, ContikiMAC and IEEE 802.15.4e Coordinated Sampled Listening.**

I. Introduction

Energy efficiency is the most prominent requirement of low power and lossy networks (LLNs) of Internet-of-Things devices. State-of-the-art research aims to have a longer battery lifetime than the device lifetime for LLN nodes. This ambitious goal requires novel communication protocols at every layer of the protocol stack. A complication is the verification and analysis of these protocols in terms of dependability due to complex interference between the layers, and not much work has been done on the analysis of energy efficient protocols. In this work we analyze the dependability of one of the most important classes of energy saving protocols: Asynchronous Radio Duty Cycling (ARDC) protocols, a subclass of Radio Duty Cycling (RDC) protocols.

During the lifetime of an LLN node, idle listening constitutes the highest energy expense [1]. RDC protocols avoid idle listening by turning-off the radio according to some scheme. This behavior is proven to be successful for conserving energy and even the earliest RDC protocols were remarkably successful [2][3][4][5]. Recent RDC protocols are more energy conserving and provide more functionality compared to older ones thanks to better techniques and more effective optimizations. As a subclass, ARDC protocols operate without explicit synchronization between the nodes. ARDC protocols like ContikiMAC [6] and IEEE 802.15.4e CSL [7] natively support broadcasting, offer very low duty cycles and have flexible parameter settings for different application needs. The ContikiMAC protocol has been proven to fully support IPv6 communication [8] through 6LoWPAN (IPv6 over Low-power Wireless Personal Area Networks) [9] with years of battery lifetime. This achievement itself could be sufficient for a huge number of applications on top of LLNs. However, when it comes to dependability there are still open issues in RDC protocols, which may lead to failures in receiving MAC frames. This is a vital problem for broadcast MAC frames, which can only be solved at the application layer since broadcast MAC frames are not acknowledged. This paper points out dependability problems of ARDC protocols and proposes practical solutions without violating the protocols.

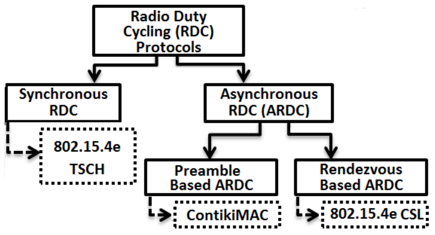


Fig. 1: A taxonomy of IPv6 compatible RDC protocols where the leaves constitute example protocols.

The contributions of this paper are threefold. First, we point out general dependability problems with ARDC protocols and show how these apply to ContikiMAC and IEEE 802.15.4e CSL. Second, we give solutions for these problems. Third, we show how to analyze these protocols in general. The paper is organized as follows. Section 2 gives a technical background on ARDC protocols. Section 3 presents dependability problems of ARDC protocols and points out solutions. Section 4 investigates average energy consumption and delay of ARDC protocols from an analytical point of view while considering dependability problems. Section 5 concludes the paper.

II. Radio Duty Cycling Background

We are particularly interested in RDC protocols that can carry IP packets as this enables the vision of Internet-of-Things. This section gives an overview of the standardized or widely accepted RDC protocols that are capable of IP communication. Figure 1 shows a categorization of RDC protocols based on their synchronicity and reception times.

A. Synchronous RDC

IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) aims to provide robustness by using time synchronized communication and channel hopping [7]. By using 6TiSCH, it is possible to allow IPv6 communication over this protocol. RDC behavior in TSCH can be implemented as a scheduling policy where the radio is turned off during some of time slots. Additional overhead for maintaining the synchronicity is a disadvantage compared to ARDC protocols. Synchronous RDC protocols are outside the scope of this work.

B. Asynchronous RDC

In ARDC protocols, nodes are not synchronized. In order to receive a MAC frame, nodes check the channel periodically. The time interval between the beginning of two consecutive channel checks is called t_{cycle}. The frequency of these checks is called *Channel Check Rate (CCR)*, expressed in hertz (Hz). If no radio activity is detected in the channel during a check, the radio is turned off until the next check. Due to this asynchronous behavior, every node is allowed to have any phase shift compared to its neighbors.

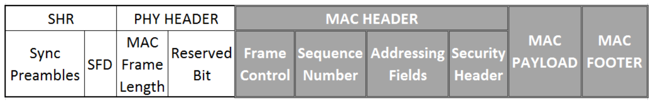
Communication can be initiated by either the sender or the receiver in ARDC. However, broadcast support is very little to non-existent in receiver initiated ARDC protocols. The IEEE 802.15.4e Receiver Initiated Transmission does not support broadcasting [7]. Other non-standardized but widely accepted receiver initiated ARDC protocols do not natively support broadcasting. They suggest either repeatedly unicasting the packet to the known neighbors or repeating the packet throughout t_{cycle} to a broadcast address as destination [10]. The first approach might have consequences in an ad-hoc environment as the sender cannot send the packet to a recently joined node. In addition, the duration of broadcasting increases linearly with the number of neighbors. The second suggestion is against the very idea behind receiver initiated communications, leading to further complications in the protocol design. Thus, receiver initiated ARDC protocols are not suitable for a general purpose network stack due to the lack of broadcasting. Therefore, this paper focuses on sender initiated protocols.

In sender initiated ARDC protocols, the sender repeatedly sends a wake-up frame until it has made sure that the receiver has received the MAC frame. Receivers check the channel every t_{cycle} for detecting a transmission of any sender, by utilizing the Clear Channel Assessment (CCA) method of the physical layer. In frame based ARDC protocols, the wake-up frame contains the actual packet. In rendezvous based ARDC protocols, the wake-up frame contains the relative sending time and the channel offset of the actual packet.

For collision avoidance, the state-of-the-art ARDC protocols use Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) without RTS/CTS exchange.

Table I explains the parameters of ARDC protocols and the model elements used for analysis in this paper. The parameters marked with * have the same meaning as within the ContikiMAC protocol specification [6].

1) Unicasting in ARDC: In state-of-the-art sender-initiated ARDC protocols like ContikiMAC [6] and IEEE 802.15.4e CSL [7], the sender stores the wake-up phase of the receiver after the first transmission. While unicasting, the sender delays the transmission until the wake-up of the receiver. Sender updates the information about the phase shift of the receiver after every unicast. This optimization called *phase-lock* [11] decreases the channel usage while unicasting.



SHR		PHY HEADER		MAC HEADER				MAC PAYLOAD	MAC FOOTER
Sync Preambles	SFD	MAC Frame Length	Reserved Bit	Frame Control	Sequence Number	Addressing Fields	Security Header		

Fig. 2: IEEE 802.15.4 [12] compliant PPDU and MAC frame formats. MAC frame is depicted in gray.

2) Broadcasting in ARDC: While broadcasting, the recipients of a packet cannot be expected to have the same wake-up phase. Consequently, ensuring the reception of a wake-up frame for a broadcast packet requires the sender to repeatedly send the wake-up frame strobe throughout a t_{cycle} and some additional extension time, namely t_{strex}. If nodes are allowed to use different t_{cycle} values, wake-up frame strobes must be sent throughout the maximum of these t_{cycle}'s plus the t_{strex}. While using rendezvous based ARDC, the relative rendezvous time is updated for each repeated wake-up frame so that receivers await the packet all at the same time.

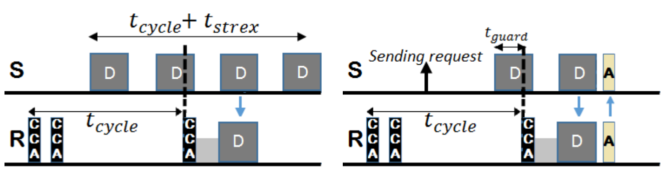
3) Sending the wake-up frames: In ContikiMAC, there are waiting times between sending subsequent wake-up frames (denoted t_i). If t_i is large, the channel check of the receiver may fail to detect the transmission. ContikiMAC suggests to perform two subsequent CCAs for each channel check. The waiting time between subsequent CCAs, called t_c, must be slightly larger than t_i. Furthermore, a wake-up frame may fall between two subsequent CCAs during the channel check [6]. In order to prevent this problem, the wake-up frame length must be sufficiently large [6].

It is also possible to send back-to-back wake-up frames without waiting times in between as specified by IEEE 802.15.4e CSL, in which case only one CCA is sufficient for a channel check. Furthermore, there is no requirement on the minimum wake-up frame length.

C. IEEE 802.15.4 header structure for ARDC

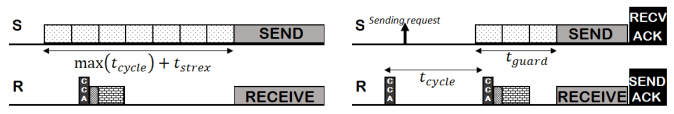
We consider the IEEE 802.15.4 compliant generic physical layer protocol data unit (PPDU) for the analysis of ARDC protocols [12]. The structure is given in Figure 2 and subsequently explained below.

- Synchronization Header (SHR): Each SHR begins with a predefined sequence of synchronization preambles. The purpose of the SHR preamble is synchronizing the receiver's radio with the sender's radio [13][14]. Without synchronization, the receiver cannot decode the packet [13]. In order to achieve synchronization, the receiver must know the content of SHR preambles and the receiver must be already sampling the channel before receiving the SHR preambles [13][14]. The SHR preamble sequence is followed by the Start of Frame Delimiter (SFD).
- PHY HEADER: Contains the payload length (MAC Frame length). Maximum MAC frame length is 127 bytes.



(a) Broadcasting (b) Unicasting

Fig. 3: Broadcasting and unicasting in ContikiMAC. D denotes wake-up frame, A denotes acknowledgement. [6]



(a) Broadcasting (b) Unicasting

Fig. 4: Broadcasting and unicasting in IEEE 802.15.4e CSL[7]

- MAC HEADER: Contains a control field, a sequence number for detecting duplicates, source and destination addresses and fields related to security.
- MAC PAYLOAD: Contains the network layer packet.
- MAC FOOTER: Contains the frame check sequence.

D. ContikiMAC

ContikiMAC is a sophisticated variant of BoX-MAC-2 [2]. It is a sender-initiated frame-based ARDC protocol that supports IP communication through 6LoWPAN [9] and uses IEEE 802.15.4 compliant frames. In ContikiMAC, packets are placed inside the wake-up frames and every node of the network checks the channel activity with the same CCR. Since every node has the same CCR, wake-up frames of broadcast packets are repeated throughout t_{cycle} plus t_{strex}. Figure 3 depicts unicasting and broadcasting in ContikiMAC. The protocol sends wake-up frames with waiting times in between. Waiting times are used for receiving acknowledgement from the receiver. Thus, the waiting time, t_i, must be greater than or equal to the necessary time for receiving an acknowledgement.

E. IEEE 802.15.4e CSL

IEEE 802.15.4e CSL is a sender-initiated rendezvous-based ARDC protocol that can support IP communication through 6LoWPAN [9]. It uses IEEE 802.15.4 compliant frames. In this protocol, wake-up frames contain relative rendezvous time and channel offset of the packet to be transmitted. Therefore, receivers do not acknowledge wake-up frames. The rendezvous time begins right at the end of the wake-up frame sequence. The protocol allows using different CCRs across the network. Thus, wake-up frames of broadcast packets are sent throughout the maximum t_{cycle} plus t_{strex}. As illustrated in Figure 4, IEEE 802.15.4e CSL relies on back-to-back wake-up frames with no waiting time between consecutive wake-up frames.

F. Parameters of ARDC

Parameters of ARDC protocols have dependencies on each other as given by the example of ContikiMAC [6]. In ContikiMAC and other frame based protocols, t_i must be long enough to allow the receiver to send an acknowledgment after the reception of a unicast packet. t_c must be greater than t_i in order to detect the wake-up frame strobe. The minimum packet length, t_s, must be greater than $t_c + 2t_r$ for ensuring that packets cannot be missed between subsequent CCAs of receivers.

TABLE I: Parameters of the ARDC protocols and other elements that we introduce for the analysis.

Parameter	Explanation
t_{cycle}	Regular duration between two channel checks.
CCR*	Frequency of channel checks per second ($1/t_{cycle}$).
t_i*	Interval between subsequent wake-up frames.
t_c*	Interval between two subsequent CCAs during the same channel check.
t_{wp}	Amount of time that is required for sending a wake-up frame. It is equal to (PPDU Length)/Bitrate.
t_s*	Amount of time that is required for sending the shortest wake-up frame.
t_r*	Duration of a single CCA, equal to $t_{sense} + t_{sg}$
t_{sense}	Amount of time that is required to detect the activity.
t_{sg}	Extension of sensing time, equal to: $t_r - t_{sense}$
t_{guard}	Guarding time interval for possible clock drifts while using phase-lock optimization.
$CCACount$	Number of CCAs performed by a receiver for a single channel check (2 in ContikiMAC).
t_{strex}	Guarding time for the broadcast wake-up frame. Used for guaranteeing reception of every receiver.
n	Number of wake-up frames while broadcasting. It is equal to $\lceil (t_{cycle} + t_{strex})/(t_{wp} + t_i) \rceil$ since the numerator may not be a multiple of denominator.
t_0	Starting moment of sending the first wake-up frame.
$t_{LatestMoment}$	Relative to t_0. Latest moment for receiving the last wake-up frame of the broadcast strobe.
$t_{LatestPhase}$	Relative to t_0. Receiver with the latest phase shift for receiving a wake-up frame from the broadcast strobe.
$t_{idleTimeout}$	Timeout limit for idle listening. If channel is idle for this amount of time, listening is terminated.

Receiving can start only after a first energy detection which must be performed on a preceding frame than the one that is received and understood. This requirement is enforced by the 802.15.4 PPDU structure, since the SHR is in the beginning of the PPDU. Thus, there must be a period of idle listening the channel until beginning of the wake-up frame.

For better comprehension of the analytic description, we define four other elements of the model:

t_0: Represents the start time of sending the first wake-up frame. $t_{LatestMoment}$ and $t_{LatestPhase}$ are relative to t_0.

Phase shift: The difference between t_0 and the starting moment of first channel check of the receiver after t_0. Therefore, phase shift is between [0, t_{cycle}). In addition, a negative phase shift represents the starting moment of the last channel check of the receiver before t_0, relative to t_0. This can be calculated by subtracting t_{cycle} from the phase shift.

$t_{LatestMoment}$: This is the latest moment for receiving the last wake-up frame of the broadcast strobe. In order to receive the last wake-up frame, the receiver must have sensed the activity from the previous wake-up frame. The receiver must have started sampling the channel at least t_{sense} before the end of the preceding wake-up frame. Therefore,

$$t_{LatestMoment} = (n-1)(t_{wp} + t_i) - t_i - t_{sense} \quad (1)$$

As given in Table I, n is calculated using the ceiling operator. Hence, $t_{LatestMoment}$ is given by:

$$t_{cycle}+t_{strex}-t_i-t_{sense}-[(t_{cycle}+t_{strex})mod(t_{wp}+t_i)] \quad (2)$$

If a receiver starts sampling the channel after this (relative) moment, it will not be able to receive wake-up frames from this broadcast strobe.

$t_{LatestPhase}$: This defines the phase shift of the latest possible receiver. Such latest receiver starts to execute its last CCA of channel check right before the first wake-up frame but fails to sense the activity.

$$t_{LatestPhase} = t_{cycle} - CCACount(t_r+t_c)+t_c+t_{sense} \quad (3)$$

III. Dependability of ARDC protocols

A. RSSI related dependability problems in the ARDC protocols

In wireless communications, CCA is an essential part of the MAC layer, indicating the availability of a channel for packet transmission. If the CCA suggests that the channel is idle, the MAC layer can attempt to send the packet. ARDC protocols use CCA for checking the channel every t_{cycle}. The method of CCA can be either sampling the channel for observing a specific type of signal or measuring the energy level on the channel to check whether the energy is above a threshold [6][12][15]. IEEE 802.15.4 allows both energy measurement based and channel sampling based CCA, both of which are implemented in IEEE 802.15.4 compliant radios such as CC2420 [15].

There is a near-linear relationship between the battery lifetime and the CCR of an ARDC network [16]. Regular channel checks are typically performed several times per second which is significantly more frequent than sending and receiving of packets. Thus, reducing the energy usage of regular channel checks can lead to a significant increase of the battery lifetime. Since sampling the channel costs a lot of energy, both ContikiMAC and IEEE 802.15.4e CSL rely on energy measurement based CCA [6].

Received Signal Strength Indicator (RSSI) is a tool for measuring the energy level on a channel frequency. RSSI based CCA simply compares the RSSI value against a predefined threshold. If the RSSI is below the threshold, the channel is assumed to be idle. Otherwise, the channel is considered to be busy. There are two approaches to RSSI calculation.

1) The most popular approach used in practice is to average the energy measurements (in dB) over a predefined amount of time (t_r). Current IEEE 802.15.4 standard uses this approach with $t_r = 0.128ms$ [12][15].
2) The second approach is to report the highest energy measurement over a predefined amount of time (t_r). This approach is specified in an obsolete version of IEEE 802.15.4 standard [17], but it is not used in practice.

These two approaches can result in radically different dependability and performance. The first approach (averaging) introduces a latency in sensing the channel activity, which may create dependability problems as the CCA result depends on:

- Duration of receiving the signal: The signal may start after the start of RSSI measurement duration or it may end before the end of the RSSI measurement duration. In this case, the average RSSI can be below the threshold as the time in which the channel is not active is used in the RSSI calculation.
- The difference between the received signal strength and the threshold: The value of the measured signal strength affects the result of CCA. This means that the channel characteristics, and the distances or obstacles between sending and listening radios affect the CCA result.
- The strength of the noise in the channel: The contribution of the background noise to the average RSSI calculation may change result of the CCA.

Formally, for detecting a busy channel, the following inequality must hold, where $RSS(t)$ denotes the received signal strength at time t:

$$threshold \leq \frac{\int_{t_\theta}^{t_\theta+t_r} RSS(t)dt}{t_r} \quad (4)$$

For simplicity, assume that the signal strength and the background noise are constant for the duration over which the RSSI is calculated. The assumption is synthetic, but the purpose of this exercise is just to illustrate the timing sensitivity of CCA (minimum t_{sense}) using the first type of RSSI calculation.

$$threshold \leq \frac{(t_r - t_{sense})noise + (t_{sense})RSS}{t_r}$$

where $RSS > threshold > noise$. This can be re-written as:

$$\frac{(threshold - noise)t_r}{RSS - noise} \leq t_{sense} \quad (5)$$

Inequality (5) gives a lower bound on the t_{sense}. Lim et al. have performed experiments about RSSI measurements of CC2420 [18]. They have shown that calculation of RSSI by energy averaging may lead to false positives (reporting activity while the channel is idle due to a recently terminated strong signal) and false negatives (failing to report the activity while there is a recently started signal due to high t_{sense}) while performing CCA. These results are in accordance with inequality (5), i.e. higher signal strengths are detected more quickly compared to lower signal strengths. Due to channel characteristics and topology, a sender's signal strength cannot be equally propagated to all the neighboring radios in its transmission range. This implies that some neighbors sense the signal faster than the others, even though the signal strength is always above the threshold for every neighbor. This is a potential dependability problem for upper layers.

Furthermore, collision avoidance prior to sending wake-up frames is negatively affected by this phenomenon. The CSMA/CA protocol sender checks the channel via CCA and starts using the channel immediately. False negatives in performing CCA lead to collisions between senders despite CSMA/CA. Probability of experiencing this problem increases with implicit synchronicity (forced by upper layers) between senders, longer sensing durations and packet rate of senders.

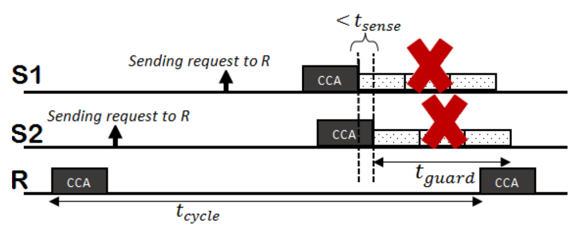


Fig. 5: Collisions while unicasting. Senders have their packet ready at different times, but they await the receiver's wake up.

Unicasting in ARDC protocols also has dependability problems due to averaged RSSI. For unicasting, state-of-the-art protocols use *phase-lock* optimization. The sender knows the phase shift of the receiver and delays sending the wake-up frame until receiver's closest channel check time. However, in practice, the sender starts sending the wake-up frame t_{guard} seconds earlier than the receiver's channel check time in order to deal with clock drifts. The protocol specification of IEEE 802.15.4e CSL does not specify how to set t_{guard}. In the reference implementation of ContikiMAC [19], all nodes use the same t_{guard} value, which is predefined. ContikiMAC calculates the wake-up phase of the receiver using reception times of acknowledgements, which is inaccurate. In IEEE 802.15.4e CSL, the receiver sends in the acknowledgement its wake-up phase relative to the frame reception time, which allows the sender to have a more precise knowledge of the wake-up phase.

Phase-lock optimization in combination with accurate phase detection, as in IEEE 802.15.4e CSL, can also cause problems. When multiple senders try to send a packet to the same receiver during the same t_{cycle} of the receiver, the senders start sending their wake-up frames at the same time [11]. As a result, the receiver may not be able to receive any wake-up frames, which might be frequent when the routing protocol (or the topology) forces some nodes to send packets through the same neighbor. The problem is not solved by CSMA/CA either because the senders are silent until they start sending frames. While clock drifts of the senders can lead to different sending times for each sender, long t_{sense} can easily eliminate the randomizing effect of clock drifts. Even after a long operational duration, clock drifts may not create significant timing differences compared to the long sensing time of IEEE 802.15.4 compliant RSSI. This is because the sender updates the information about the phase shift of the receiver after every unicast. Figure 5 shows possible scenarios.

Assuming that there are

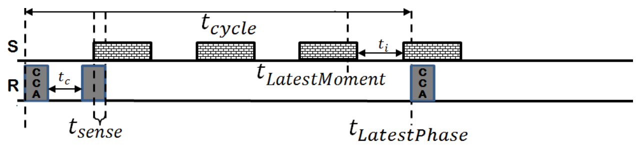


Fig. 7: Failing to receive broadcast packets. Receiver with the latest phase failed to detect the first wake-up frame. When it checks the channel again, it will be too late to receive the last wake-up frame.

a broadcast frame, the problem can only be solved within the application layer, which introduces significant complexity in terms of communication and computation.

We focus on the design problems of ARDC protocols that cause broadcast frame reception failures, even in noise-free channels. We illustrate these problems analytically and experimentally, and provide solutions for them.

This section employs a generic ARDC protocol that can be interpreted as both rendezvous-based and frame-based ARDC depending on the values of t_i, t_c, t_{wp} and $CCACount$. Using this, we are able to show broadcast dependability problems that are inherent in both types of ARDC protocols, exemplified by ContikiMAC and IEEE 802.15.4e CSL. These problems are observable in the current ContikiMAC implementation and, without the proposed solutions, they will be visible in (future) implementations of IEEE 802.15.4e CSL.

1) Optimal number of repeated wake-up frames: Broadcasting a packet requires sending repeated wake-up frames throughout t_{cycle} in order to make sure that any neighbor with any channel check phase can receive the packet. However, if a receiving neighbor completes its channel check right before the wake-up frame strobe, it may not receive any of the wake-up frames. Figure 7 shows a possible scenario. There are two solutions to this problem. The first solution is increasing the number of subsequent CCAs per channel check for the receiver. This solution is preferred by neither ContikiMAC nor IEEE 802.15.4e CSL as it linearly increases the energy cost of channel checks, reducing the battery lifetime significantly. The second solution is increasing the number of repeated wake-up frames by extending the time of a broadcast strobe by a certain amount, called t_{strex}. This approach is used by the implementation of ContikiMAC.

Increasing the time of repeated wake-up frames has its own risks. A short increment may not provide a dependable broadcasting while a long increment cause duplicate reception of wake-up frames. Duplicate receptions must be avoided too, since they cause a waste of energy for the receivers. There is a clear trade-off, as the first priority of an ARDC protocol should be guaranteeing dependable broadcasting while trying to decrease the probability of duplicate receptions.

Neither IEEE 802.15.4e CSL nor ContikiMAC mention t_{strex} at all in their specifications. However, the implementation of the ContikiMAC protocol forces to use a predefined constant for t_{strex}. Nevertheless, the optimal value of this constant needs to be determined. In the current ContikiMAC implementation it is possible to miss broadcast packets due to short t_{strex} using default parameters. This subsection provides the minimum value of t_{strex} in ARDC protocols for ensuring dependable broadcast transmission to neighbors. Then we compute the probability of losing broadcast packets in the implementation of ContikiMAC with default parameters.

As addressed in Section 2, a receiver cannot receive a wake-up frame if it performs its channel check after $t_{LatestMoment}$. Therefore, it must be ensured that the latest receiver to check the channel performs CCA before $t_{LatestMoment}$. Thus,

$$t_{LatestMoment} \geq t_{LatestPhase} \tag{7}$$

must be satisfied for dependable broadcasting. Otherwise, the receivers with a phase shift within $(t_{LatestMoment}, t_{LatestPhase}]$, will not be able to receive any wake-up frames of the broadcast strobe.
Substituting for $t_{LatestMoment}$ and $t_{LatestPhase}$ gives:

$$t_{cycle} + t_{strex} - t_i - t_{sense} - (t_{cycle} + t_{strex}) mod(t_{wp} + t_i) \\ \geq t_{cycle} - CCACount(t_r + t_c) + t_c + t_{sense}$$

This can be rearranged as:

$$(t_{cycle} + t_{strex}) mod(t_{wp} + t_i) \leq \\ t_{strex} - t_i - 2t_{sense} + CCACount(t_r + t_c) - t_c \tag{8}$$

In IEEE 802.15.4e CSL, $t_i = 0$, $CCACount = 1$, $t_c = 0$ and t_{sense} is in the interval $(0, t_r]$. The value of t_{sense} must be set to t_r since $t_i = 0$, i.e. the CCA must be finished before the starting instance of the last wake-up frame. Thus, the inequality for IEEE 802.15.4e CSL becomes:

$$(t_{cycle} + t_{strex}) mod(t_{wp}) \leq t_{strex} - t_r \tag{9}$$

This inequality is always satisfied in IEEE 802.15.4e CSL when the following holds.

$$t_{wp} + t_r \leq t_{strex} \tag{10}$$

In IEEE 802.15.4e CSL, the length of the wake-up frame is fixed. Thus, choosing $t_{strex} = t_{wp} + t_r$ ensures that all receivers get a chance to receive a wake-up frame.

For ContikiMAC, $CCACount = 2$ and t_{sense} is in the interval $(0, t_r]$ (it must be set to t_r for determining the minimum value of t_{strex}). The inequality for ContikiMAC becomes:

$$(t_{cycle} + t_{strex}) mod(t_{wp} + t_i) \leq t_{strex} - t_i + t_c \tag{11}$$

This inequality is always satisfied in ContikiMAC when:

$$t_{wp} + 2t_i - t_c \leq t_{strex} \tag{12}$$

In ContikiMAC, the length of a wake-up frame varies since it contains the actual packet. Thus, t_{wp} must be set to the maximum possible value of t_{wp}, namely $MAX(t_{wp})$. ContikiMAC uses other default parameter values, namely, $t_{strex} = 2.512ms$, $t_i = 0.4ms$ and $t_c = 0.5ms$. Hence, the inequality (12) is satisfied only when:

$$t_{wp} \leq 2.212ms \tag{13}$$

However, t_{wp} can take values in [0.884ms, 4.256ms] with a bitrate of 250 kbps, since $t_s = 0.884ms$ and maximum PPDU length is 133 bytes. If the time to send a wake-up frame is longer than 2.212ms, some of the receivers may not be able to receive any wake-up frames.

The probability of failing to receive a broadcast packet with the default parameters of ContikiMAC is calculated as follows.

$$P(failing\ to\ receive) = \begin{cases} \frac{t_{LatestPhase} - t_{LatestMoment}}{t_{cycle}}, & t_{LatestPhase} > t_{LatestMoment} \\ 0, & otherwise \end{cases}$$

The first case of the probability can be decomposed as:

$$\frac{t_{LatestPhase} - t_{LatestMoment}}{t_{cycle}} = \frac{[(t_{cycle} + t_{strex}) mod(t_{wp} + t_i)] - t_{strex} + t_i - t_c}{t_{cycle}} \quad (14)$$

In case of high CCRs, probability of failing to receive the broadcast packet increases. Due to the non-linear behavior of modulo operation, probability of failing to receive a broadcast packet does not linearly increase with the t_{wp}.

To the best of our knowledge, there is no publicly available implementation of IEEE 802.15.4e CSL. The protocol requires to send back-to-back wake-up frames for long amounts of time. Because staying in the TX mode for longer than the time to send the longest MAC frame is not mandated by IEEE 802.15.4, popular radios do not natively support this operation. A non-standard feature of CC2420 allows staying in TX mode by repeatedly sending the entire TX FIFO, which is used by BoX-MAC-1 [2]. However, a possible implementation of IEEE 802.15.4e CSL cannot easily utilize this feature of CC2420 since it causes problems while providing a unique rendezvous time for each wake-up frame.

Definition of the IEEE 802.15.4e CSL does not mention the t_{strex}. While testing the IEEE 802.15.4e CSL in simulation, we observed that conforming to the inequality (10) ensures dependability for broadcast frames.

During our simulations (in COOJA [20]) and experiments for ContikiMAC, we realized that the time interval between two consecutive wake-up frames are considerably more than the default t_i value (0.4 ms) despite the fact that the implementation initializes a timer for 0.4 ms before sending the next wake-up frame. After setting different values to t_i for comparison, we observed that in addition to t_i, there is a constant amount of time between two consecutive frames which varies depending on the hardware. Our analytical model considers the amount of time between two consecutive wake-up frames and, therefore, uses the experimentally proven t_i value instead of the value that is being used for initializing the timer. Main reason of experiencing higher t_i values is the wait for the radio wake-up. Same as the t_i case, there is some delay before each RSSI measurement, depending on the hardware. The analytical model can be adapted for this factor as well. In the analytical model, t_c is the duration between the end of first RSSI measurement and the beginning of second RSSI measurement. Therefore, the delay before the second RSSI measurement shall be included within the t_c duration. The delay before the first RSSI measurement does not have any effect to the analytical model since it only changes the phase shifts of receivers by the same amount of time. In addition to radio wake-up time, layered software design introduces more delay due to the amount of necessary time to run the code right before and right after busy waiting t_i and t_c, depending on the hardware. Thus, t_i and t_c are experimental values rather than being configured timer initializations. Using Wismote, when the ContikiMAC's default configuration of t_i is set to 0.4 ms we measured an experimental t_i value of 1.115-1.117 ms. For the t_c case, we measured approximately 0.830 ms although the default configuration of t_c is set to 0.5 ms.

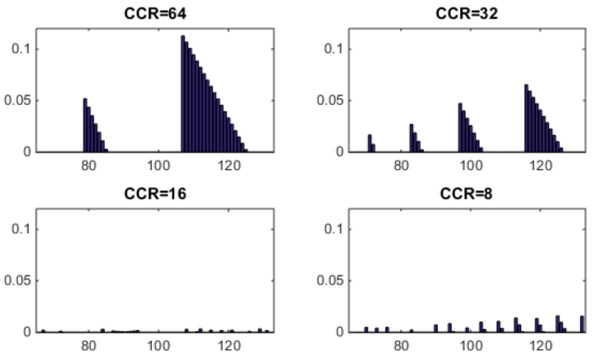


Fig. 8: Probability of failing to receive broadcast wake-up frames in ContikiMAC for various CCR while using CC2520 (horizontal axis is PPDU length, vertical axis is probability).

As the analytical work suggest, results in Figure 8 show non-linear behavior with respect to the length of t_{wp} between a PPDU length of [70,133] bytes. Our experimental results overlap with the analytical results, when the experimental t_i and t_c values are used within formula (14).

It might be reasoned that the dependability of the ContikiMAC protocol could be easily ensured by increasing t_{strex}. However, higher values of t_{strex} causes duplicate receptions which wastes energy of the receivers. By duplicate reception, we mean that a receiver has already received a wake-up frame of the broadcast strobe, but the next channel check of the receiver is performed while the sender is still sending the repeated wake-up frames. If the start of the next channel check is in $(t_{LatestPhase}, t_{LatestMoment}]$, the receiver will have a duplicate. The probability of receiving duplicate broadcast wake-up frames is:

$$P(duplicate) = \frac{t_{LatestMoment} - t_{LatestPhase}}{t_{cycle}} \quad (15)$$

Figure 9 shows two extreme cases of the duplicate broadcast packets problem. Note that $t_{LatestPhase} < t_{LatestMoment}$ must hold for receiving duplicates. The problem of receiving duplicate broadcast packets is complementary to the problem of failing to receive broadcast packets, since its solution requires the opposite of the inequality (7) for dependable broadcast transmission. When dependability is of utmost im-

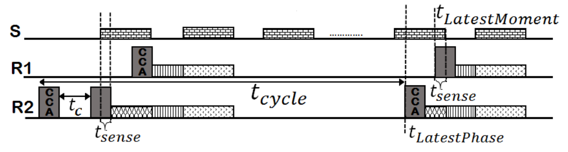


Fig. 9: Receiving duplicates. Some receivers can receive the last wake-up frame despite they already have one.

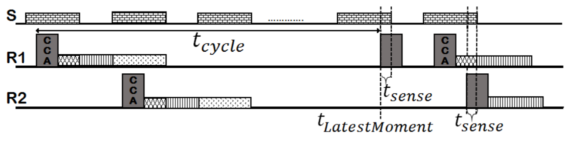


Fig. 10: Timed-out receivers. These receivers had already received a wake-up frame. Their next channel check enforced them to sample the last wake-up frame for receiving the next wake-up frame. Since there will be no more wake-up frames, they will sample the idle channel for $t_{idleTimeout}$.

portance, duplicate packets are inevitable. In order to ensure dependability for various packet lengths while using constant t_{strex}, t_{wp} of the minimum requirement must be set to $MAX(t_{wp})$ as t_{wp} is subject to change for every transmission. However, this increment creates larger differences between $t_{LatestMoment}$ and $t_{LatestPhase}$ for smaller packets which increases the probability of duplicate receptions. The source of this problem is using a predefined constant for t_{strex}. One solution could be calculating the t_{strex} for the current packet length before every broadcast. This dynamicity allows guaranteeing $t_{LatestMoment} \geq t_{LatestPhase}$ without further increasing the difference between them. Another solution is checking the current time (relative to t_0) right after sending a repeated wake-up frame. When

$$CurrentTime \geq (t_{LatestPhase} + t_{sense} + t_i + t_{wp}) \quad (16)$$

holds, there is no need to send another wake-up frame.

For the protocols that are sending wake-up frames back-to-back like 802.15.4e CSL, it is sufficient to set the t_{strex} to $t_r + t_{wp}$. This is because wake-up frames have fixed length and receivers turn off their radio until the rendezvous time which prevents duplicate receptions.

2) Timed-out broadcast receivers: There may be receivers that are trying to receive a wake-up frame for the second time, but fail to do so. If a receiver checks the channel while the sender is sending the last wake-up frame, the receiver will sample the channel for receiving the SHR. Since the SHR has been already missed, when the sender completes sending the last wake-up frame, the receiver continues to sample the channel anyway. After idle listening the channel for $t_{idleTimeout}$, the receiver concludes that there will be no more wake-up frames and turns the radio off. This is how the implementation of ContikiMAC handles timed-out receivers [19]. This process is quite convenient to use in practice, if $t_{idleTimeout}$ is not too high compared to t_i, which is the amount of waiting time between two consecutive wake-up frames. On the other hand, setting the $t_{idleTimeout}$ slightly higher than t_i protects the receiver from clock drifts.

Timed-out receivers waste their energy while sampling the channel for SHR and idle listening afterwards. A timed-out receiver has already received a wake-up frame of the broadcast strobe. However, the next channel check of the receiver is also performed while the sender is still sending the strobe. The receiver cannot receive the wake-up frame again, if the next channel check is started after $t_{LatestMoment}$. The probability of timing-out while receiving a broadcast wake-up frame is:

$$P(timing\text{-}out) = \frac{t_{wp} + t_i}{t_{cycle}} \quad (17)$$

High CCR and lengthy frames cause more receivers to face this problem. Figure 10 shows two extreme cases of the time-out problem. Note that the timed-out receptions cannot occur in rendezvous based ARDC protocols. In rendezvous-based ARDC protocols, receivers turn their radio off until the rendezvous time prior to reception of a wake-up frame.

IV. Average delay and energy consumption of ContikiMAC and IEEE 802.15.4e CSL

This section provides a formal analysis of the receivers' energy waste due to the aforementioned dependability problems (duplicate packets and time-outs). Our analysis is based on the average durations of idle-listening and active listening of the radio-channel. For conversion to energy units, these durations must be multiplied with the radio-specific power requirements for sampling the idle channel and sampling the incoming PPDU for SHR. The analysis shows that this waste can be significant for bad parameter settings, i.e. comparable to the energy cost of actually receiving the packets.

Energy consumptions of ARDC protocols depend on CCR and traffic rate [6][16]. However this section analyses the energy consumption for receiving a *single* MAC frame. This is because the state-of-the-art protocols waste a significant amount of energy prior to reception of the packet due to idle listening of the channel. To the best of our knowledge, this aspect has not been investigated in the literature.

Average radio energy consumption can be decomposed into reception and overhead parts. The overhead consists of contributions from idle listening time right before the reception, duplicate receptions and timed-out receivers. Although both are considered to be idle listening, we separately treat sampling the channel when there is no signal (*idle sampling*) and sampling an incoming PPDU for receiving SHR (*busy sampling*), since the radio may consume different amounts of energy for these two cases.

It is assumed that broadcast transmissions are reliable, i.e. t_{strex} is calculated before every transmission rather than using a predefined value. No receiver can receive the first wake-up frame of the broadcast strobe while using IEEE 802.15.4 compliant PPDUs. This is because for receiving a frame, a receiver must not miss the SHR, which requires sampling the channel prior to the transmission moment of that frame. However, receivers do not sample the channel until they detect energy.

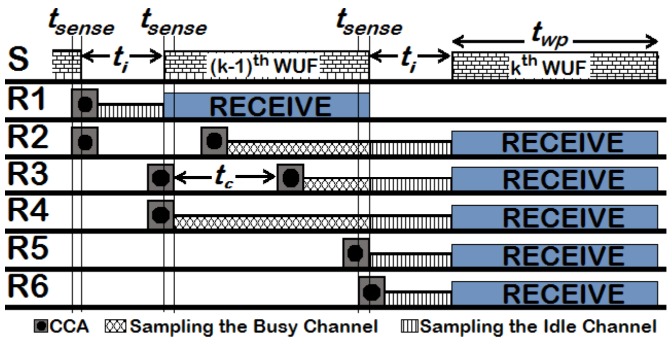


Fig. 11: Critical reception scenarios of ContikiMAC.

TABLE II: Mapping of receptions in ContikiMAC

Reception	**Interval of receiver's phase shifts relative to t_0**
2^{nd} WUF	$[-2t_r - t_c + t_{sense}, (t_{wp} + t_i) - t_i - t_{sense}]$
k^{th} WUF	$((k-2)(t_{wp}+t_i)-t_i-t_{sense}, (k-1)(t_{wp}+t_i)-t_i-t_{sense}]$
n^{th} WUF	$((n-2)(t_{wp}+t_i)-t_i-t_{sense}, t_{cycle}-2t_r-t_c+t_{sense}]$
Duplicate	$(t_{cycle}-2t_r-t_c+t_{sense}, (n-1)(t_{wp}+t_i)-t_i-t_{sense}]$
Timed-out	$((n-1)(t_{wp}+t_i)-t_i-t_{sense}, n(t_{wp}+t_i)-t_i-t_{sense}]$

A. *ContikiMAC*

For analysis, this section will use the Figure 11, which shows critical reception scenarios while receiving the k^{th} wake-up frame. Table II demonstrates receivers with which phase shifts receive which frame and which phase shifts can have the problems of duplicate wake-up frames and timed-out receptions. Note that all times are relative to t_0.

1) Unicasting: The sender starts sending a strobe of wake-up frames earlier than the phase shift of the receiver in order to avoid missing the receiver due to clock drifts. However, the receiver will not be able to receive the first wake-up frame. Until the next wake-up frame, the receiver samples the channel for SHR. Assuming that the receiver has a reasonable clock drift, it catches the wake-up frame strobe after it has already started. Thus, it will catch the k^{th} wake-up frame with a phase shift as shown in the Table II (phase shifts between [R2, R6] in Figure 11).

The shortest delay of receiving a unicast wake-up frame can be achieved when the first CCA is done $t_i + t_{sense}$ before the beginning of the k^{th} frame (phase shift of R6 in Figure 11). In this case delay would be $t_{sense} + t_i + t_{wp}$.

The longest delay of receiving a unicast wake-up frame occurs when the receiver slightly fails to detect the $(k-2)^{th}$ wake-up frame, does its second CCA inside the $(k-1)^{th}$, then it receives the k^{th} wake-up frame (R2 in Figure 11). This leads to a delay of $t_{sense} + 2t_i + 2t_{wp}$.

In the average case, the sender wants to send a packet halfway through the wake-up phase $t_{cycle}/2$ and waits until the phase-lock. Therefore, the main contributor of the delay is t_{cycle} since it is very large compared to t_{wp}. The average delay of unicasting is then:

$$t_{sense} + \frac{3(t_i + t_{wp})}{2} + \frac{t_{cycle}}{2} \tag{18}$$

There are two different cases where the energy consumption needs to be calculated.

Case 1: The first CCA detects radio activity, i.e. phase shift is in the interval ([R4, R6] in Figure 11):

$[(k-2)(t_{wp} + t_i) - t_{sg}, (k-1)(t_{wp} + t_i) - t_i - t_{sense})]$

Case 2: The second CCA detects radio activity, i.e. phase shift is in the interval ([R2, R3] in Figure 11):

$((k-2)(t_{wp} + t_i) - t_i - t_{sense}, (k-2)(t_{wp} + t_i) - t_{sg})$

Considering the probabilities of occurrence for these two cases, the average time for busy sampling becomes:

$$\frac{(t_i + t_r - 2t_{sg})(2t_{wp} + t_i - 3t_r - 2t_c) + (t_{wp} - t_{sense})^2}{2(t_i + t_{wp})} \tag{19}$$

Most receivers (i.e. [R2, R5] in Figure 11) perform sampling while the channel is idle for a duration of t_i. However, the receivers with a phase shift within:

$[(k-1)(t_{wp} + t_i) - t_i - t_r, (k-1)(t_{wp} + t_i) - t_i - t_{sense})$

do fewer idle channel samplings since they have performed the last portion of their CCA when the waiting time between wake-up frames has already started ([R5, R6] in Figure 11). The average duration of sampling the idle channel is:

$$\frac{(t_i - t_{sg}/2)t_{sg} + t_i(t_i + t_{wp} - t_{sg})}{t_i + t_{wp}} \tag{20}$$

In addition to sampling the idle channel and sampling the channel for SHR, the receiver spends energy for receiving the actual packet and sending the acknowledgement.

2) Broadcasting: The sender repeats wake-up frames throughout a $t_{cycle} + t_{strex}$ duration. Each of the wake-up frames contains the packet. The receiver goes back to sleep after receiving the packet.

Average delay of broadcasting is calculated relative to t_0. Using Table II, the average delay is equal to:

$$\sum_{k=2}^{n} (k(t_{wp} + t_i) - t_i) P_{receive}(k) \tag{21}$$

where $P_{receive}(k)$ denotes the probability of receiving the k^{th} wake-up frame for the first time. After obtaining the probabilities using Table II, the average delay is:

$$\frac{(t_{wp} + 2t_{sg} + t_c)(2t_{wp} + t_i)}{t_{cycle}} + \frac{[(n^2 - n - 6)(t_{wp} + t_i) - 2(n-3)t_i](t_{wp} + t_i)}{2t_{cycle}} + \frac{(t_{cycle} - 2t_{sg} - t_c + t_i - (n-2)(t_{wp} + t_i))((t_{wp} + t_i)n - t_i)}{t_{cycle}} \tag{22}$$

For simplicity, we assume that average energy overhead of receiving the first and the last wake-up frames are the same as the other wake-up frames since there is not a significant difference between them. Therefore, the average idle and busy sampling time for receiving a broadcasting wake-up frame prior to reception, will be the same as for unicasting since the receivers of broadcast frames also wake up in the interval:

$$((k-2)(t_{wp}+t_i)-t_i-t_{sense},(k-1)(t_{wp}+t_i)-t_i-t_{sense}]$$

In addition to the overhead right before receiving the wake-up frame, the average energy consumption must also consider duplicate receptions and timed-out receivers. Since every timed-out receiver samples the idle channel for $t_{idleTimeout}$, the average idle sampling duration of a receiver is:

$$t_{idleTimeout}\frac{t_{wp}+t_i}{t_{cycle}} \tag{23}$$

Moreover, the timed-out receivers sample the last wake-up frame, which further increases the energy overhead. In order to calculate the average duration of busy sampling the last wake-up frame, assume that the $(k-1)^{th}$ wake-up frame in Figure 11 is the last wake-up frame of the broadcast strobe. In this case, average duration of busy sampling the last wake-up frame before falling to time-out is equal to the average duration of busy sampling before receiving the k^{th} wake-up frame:

$$\frac{(t_i+t_r-2t_{sg})(2t_{wp}+t_i-3t_r-2t_c)+(t_{wp}-t_{sense})^2}{2t_{cycle}} \tag{24}$$

Note that (23) and (24) provide the average duration of busy sampling and idle sampling due to the time-out problem for any receiver with any phase shift.

The average energy overhead of receiving a duplicate wake-up frame can be calculated using the average durations below.

The average duration of idle sampling is:

$$\frac{t_{sg}}{t_{cycle}}(t_i-\frac{t_{sg}}{2})+t_i\frac{(n-1)(t_{wp}+t_i)-t_i-t_r-t_{LatestPhase}}{t_{cycle}} \tag{25}$$

The average duration of busy sampling is:

$$\frac{[(n-1)(t_{wp}+t_i)-t_i-t_r-t_{LatestPhase}]^2}{2t_{cycle}} \tag{26}$$

In addition to idle sampling and busy sampling of the duplicate wake-up frame, the receiver also spends energy for receiving the duplicate wake-up frame. The average duration of receiving the duplicate can be calculated by multiplying the probability of receiving a duplicate (formula (15)) and t_{wp}.

Note that (25) and (26) provide the average duration of busy sampling and idle sampling prior to the duplicate reception, for any receiver with any phase shift.

B. IEEE 802.15.4e CSL

1) Unicasting: Similar to the ContikiMAC case, the receiver cannot receive the first wake-up frame. The receiver samples the channel for SHR on average for half the duration of a wake-up frame, after which it receives the wake-up frame. The receiver goes back to sleep until the next rendezvous time. However, the sender continues to send wake-up frames. t_{guard} shall cover the rendezvous time from both forward and backward clock drifts. Therefore, the average delay becomes:

$$\frac{t_{cycle}}{2}+t_{guard}+t_{packet} \tag{27}$$

The receiver spends energy before receiving one of the back-to-back wake-up frames. The average time of sampling the channel for SHR (busy sampling) is:

$$t_{wp}/2 \tag{28}$$

The receiver also spends energy for receiving one wake-up frame and the packet itself and sending the acknowledgement.

2) Broadcasting: The sender repeats the wake-up frames throughout the duration of $t_{cycle}+t_{strex}$. In broadcasting, every neighbor receives the packet at the same time (right after wake-up frame strobe). Hence, the delay is defined as:

$$t_{cycle}+t_{wp}+t_r+t_{packet} \tag{29}$$

Similar to unicasting, nodes only receive one wake-up frame and go back to sleep. The average time of sampling the channel for SHR is:

$$t_{wp}/2 \tag{30}$$

The receiver also spends energy for receiving one wake-up frame and the packet itself.

C. ContikiMAC versus IEEE 802.15.4e CSL

IEEE 802.15.4e CSL wake-up frames are very short (18 octets in current specification) compared to wake-up frames of ContikiMAC. This causes a significant disadvantage for ContikiMAC by resulting in longer sampling times for SHR. Additionally, waiting times between wake-up frames further increase the energy consumption of ContikiMAC receivers.

While broadcasting, IEEE 802.15.4e CSL does not suffer from either duplicate receptions or timed-out receivers. On the other hand, these two problems cause additional energy waste for ContikiMAC. Figure 12 illustrates average busy sampling overhead of ContikiMAC broadcasting by giving the sum of: The average duration of busy sampling right before reception (formula (19)), the average duration of receiving the duplicate, the average duration of busy sampling of idle timing-outs (formula (24)), the average duration of busy sampling prior to duplicate receptions (formula (26)). The average idle sampling overhead of ContikiMAC broadcasting is illustrated in Figure 12 is given by the sum of: The average idle sampling right before reception (formula (20)), the average idle sampling of duplicate receptions (formula (25)) and average idle timing-outs (formula (23)). In Figure 12, average busy sampling overhead of IEEE 802.15.4e CSL is given by the sum of the average duration of busy sampling right before reception (formula (30)) and the length of one wake-up frame. As shown in Figure 12, IEEE 802.15.4e CSL does not have idle sampling overhead. According to Figure 12, busy sampling overhead of ContikiMAC linearly increases with MAC frame length and it is very close to the sampling time of actually receiving the MAC frame. The busy sampling overhead of IEEE 802.15.4e CSL is smaller and constant.

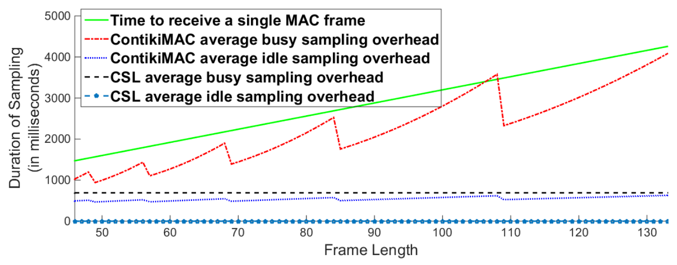


Fig. 12: Average durations of sampling the busy channel and sampling the idle channel while receiving a single broadcast frame in CSL and ContikiMAC (with default parameters). CSL has much less overhead compared to ContikiMAC.

ContikiMAC has one advantage over IEEE 802.15.4e CSL: The average delay of the broadcast transmissions is roughly half of IEEE 802.15.4e CSL. However, in IEEE 802.15.4e CSL every neighbor receives the packet at the same time.

While unicasting, IEEE 802.15.4e CSL is more energy efficient since it benefits from the short and back-to-back wake-up frames, leading to shorter idle listening times. Transmission delay is nearly the same for both due to a long waiting time for the receiver to wake-up.

V. Conclusions

This paper presents dependability problems of widely accepted IPv6 compatible ARDC protocols and proposes practical solutions to them. The presented dependability problems are shown by concrete examples and analytical deductions along with simulations. Our direction towards solving these problems is avoiding complex approaches and proposing lightweight solutions that can be practically applied.

We first analyzed sensitivity of RSSI and analytically showed that it can lead to false positives and false negatives while performing CCA due to long sensing duration. We presented how it can damage the dependability of unicasting in ARDC protocols while using phase-lock optimization. We have proposed a practical solution with parameter settings to mitigate this dependability problem.

Then we showed that the default parameters of ContikiMAC may disallow some receivers from receiving broadcast packets. However, solving this dependability problem inevitably increases the energy consumption by introducing more duplicate receptions. We analytically showed how careful parameter settings can allow regaining the dependability while keeping the number of duplicate receptions to a minimum. Moreover, we pointed out the problem of timed-out receivers from an analytical perspective. We showed the impacts of idle listening prior to reception, duplicate receptions and timed-out receivers to the average energy consumption of frame based ARDC protocols like ContikiMAC. According to the analytical results, the broadcasting mechanism of ContikiMAC spends almost twice as much time sampling the channel of receiving a single broadcast packet.

Lastly, we compared ContikiMAC and IEEE 802.15.4e CSL in terms of delay and energy consumption. IEEE 802.15.4e CSL is more energy efficient than ContikiMAC while broadcasting and unicasting. However, average delay of broadcasting is higher in IEEE 802.15.4e CSL, but it has zero variance. Depending on the application layer requirements, predictability can be favored over smaller average delay.

VI. Acknowledgement

This research is enabled by the OpenAIS project, supported by the Horizon 2020 program of the European Union under grant agreement number 644332.

References

[1] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks,” in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, ser. SenSys ’06. ACM, 2006, pp. 307–320.

[2] D. Moss and P. Levis, “Box-macs: Exploiting physical and link layer boundaries in low-power networking.”

[3] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys ’04. ACM, 2004, pp. 95–107.

[4] T. van Dam and K. Langendoen, “An adaptive energy-efficient mac protocol for wireless sensor networks,” in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys ’03. ACM, 2003, pp. 171–180.

[5] A. El-Hoiydi and J.-D. Decotignie, “Low power downlink mac protocols for infrastructure wireless sensor networks,” *Mob. Netw. Appl.*, vol. 10, no. 5, pp. 675–690, Oct. 2005.

[6] A. Dunkels, “The contikimac radio duty cycling protocol,” SICS, Tech. Rep. T2011:13, Dec. 2011.

[7] “Ieee standard for local and metropolitan area networks part 15.4: Low-rate wireless personal area networks (lr-wpans) amendment 1: Mac sublayer,” IEEE 802.15.4e, 2012.

[8] A. Dunkels, J. Eriksson, and N. Tsiftes, “Low-power interoperability for the ipv6-based internet of things,” in *Proceedings of the 10th Scandinavian Workshop on Wireless Ad-hoc Networks (ADHOC 11)*, May 2011.

[9] G. Mulligan, “The 6lowpan architecture,” in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, ser. EmNets ’07. ACM, 2007, pp. 78–82.

[10] X. Fafoutis, A. D. Mauro, M. D. Vithanage, and N. Dragoni, “Receiver-initiated medium access control protocols for wireless sensor networks,” *Computer Networks*, vol. 76, pp. 55 – 74, 2015.

[11] A. El-Hoiydi and J.-D. Decotignie, “Wisemac: An ultra low power mac protocol for multi-hop wireless sensor networks,” in *Algorithmic Aspects of Wireless Sensor Networks*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3121, pp. 18–31.

[12] “Ieee standard for local and metropolitan area networks part 15.4: Low-rate wireless personal area networks (lr-wpans),” IEEE 802.15.4, 2011.

[13] K. A. Jamieson, “The SoftPHY Abstraction: From Packets to Symbols in Wireless Network Design,” Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2008.

[14] R. M. Koteng, “Evaluation of SDR-implementation of IEEE 802.15.4 Physical Layer,” Master’s thesis, Norwegian University of Science and Technology, Norway, 2006.

[15] Texas Instruments, “CC2420 datasheet,” 2014.

[16] W. Dron, S. Duquennoy, T. Voigt, K. Hachicha, and P. Garda, “An emulation-based method for lifetime estimation of wireless sensor networks,” in *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on*, May 2014, pp. 241–248.

[17] “Ieee standard for local and metropolitan area networks part 15.4: Low-rate wireless personal area networks (lr-wpans),” IEEE 802.15.4, 2003.

[18] R. Lim, M. Zimmerling, and L. Thiele, “Passive, privacy-preserving real-time counting of unmodified smartphones via zigbee interference,” in *Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on*, June 2015, pp. 115–126.

[19] Contiki-OS v3.0. [Online]. Available: https://github.com/contiki-os

[20] F. Österlind, “A sensor network simulator for the Contiki OS,” *SICS Research Report*, 2006.