# Formal Analysis of Energy Consumption in IoT Systems

Oualid Demigha and Chamseddine Khalfi
*Ecole Militaire Polytechnique, Department of Computer Science, PO Box 17, 16111 Bordj El-Bahri, Algiers, Algeria*

Keywords: Internet of Things, IEEE 802.15.4, Energy Efficiency, Model-checking, CSMA/CA, PROMELA, SPIN.

Abstract: In this paper, we apply model-checking approach to formally analyze energy consumption of the radio interface in order to guarantee network lifetime in the context of the Internet of Things. We propose a joint MAC-physical model of the IEEE 802.15.4 standard to capture and represent key operations that consume energy resources of the nodes at the radio interface component. We argue that the combination of the radio interface ON/OFF state switching mechanism with CSMA/CA medium access method leads to better modeling of the energy consumption and help understanding the interaction between MAC and physical layers. Our model provides accurate representation of simulation models at the first two layers of node's protocol stack compliant with IEEE 802.15.4 standard.

## 1 INTRODUCTION

Internet of Things (IoT) is the next wave of the Internet that may change the world (Lin et al., 2017). It brings intelligence to ordinary things such as light switches, heaters, fridges, air conditioners, water supplies, etc. by connecting them to the Internet and transforming them to smart objects that are able to use its huge potential of ubiquitous and smart services. Consequently, they contribute to generating data for big data analysis and artificial intelligence for better decision-making and control (CISCO, 2015), (Behera et al., 2015).

This technology is enabled by the recent advances in Wireless Sensor Networks (WSN) and low-power devices with high storage and communication capabilities. It is supported by low energy profile platforms and standards and/or dedicated protocol stacks for low-rate short and long range communications. However, the main challenge that faces it in the near future is the energy problem because most of IoT devices are energy-constrained as they rely on limited-capacity batteries that are hard to replace or inaccessible after being deployed (Burdett, 2015). Additionally, they are intended to function for several months or years without human intervention. Another technical obstacle to expand IoT technology is the problem of dependability because these systems are in permanent interaction with human beings and critical systems in every-day life. Hence, it is necessary to bring confidence, reliability and privacy to IoT systems for technical maturation and large public acceptance.

Therefore, despite the unavoidable interest of experimental tests and simulation-based performance studies in tackling the above-mentioned challenges, formal methods remain a certain way to verify IoT standards and provide mathematical proofs of correctness and security. Even more, formal methods can also be used to guarantee energy performance via quantified verification of protocol operations. Relying on a deep understanding of the latter, and via a rigorous mathematical formalism, these methods can capture critical operations that consume the energy resources of the network and/or threaten its security.

In this paper, we propose to use one of the most powerful formal methods, i.e. model-checking to analyze the energy consumption behavior of CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) in IEEE 802.15.4 standard. We propose a simple yet generic and effective joint MAC-physical model that combines the ON/OFF state switching mechanism of the radio interface with CSMA/CA in non-beacon enabled mode of IEEE 802.15.4. By enhancing the model in (Rege and Pecorella, 2016), we are able to integrate a network dimension of LP-WPAN simulation models and analyze the energy consumption.

The rest of the paper is organized as follows: In Section 2, we give a brief description of IEEE 802.15.4 standard with special emphasis on CSMA/CA. In Section 3, we present the basics of the model-checking approach and the linear temporal



Demigha, O. and Khalfi, C.
Formal Analysis of Energy Consumption in IoT Systems.
DOI: 10.5220/0007702801030114
In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS 2019)*, pages 103-114
ISBN: 978-989-758-369-8

logic used to express and evaluate our model. In Section 4, we describe our joint MAC-physical model in terms of state/transition automaton. In Section 5, we give an implementation of our model in PROMELA language with the different process automata generated by SPIN and the energy consumption parameters of the radio interface. In Section 6, we give a conclusion and future work.

# 2 IEEE 802.15.4 STANDARD SPECIFICATION

IEEE 802.15.4 standard offers up to layer 2 a low-rate short-range communication service with flexible throughput and latency for wireless personal networks (Society, 2017). Its extreme low-energy consumption profile and support of most off-the-shelf IoT devices, make it a strong candidate for lower protocol stack standard of the Low-Rate Wireless Personal Area Networks (LR-WPAN), as well as other technologies such as: Bluetooth Low-Energy (BLE) 4.0, Bluetooth 5, 5G, etc. (Agerstam et al., 2018).

At the physical layer, IEEE 802.15.4 uses three different frequency bands, namely: the 868 MHz band that offers one channel with 20 Kbit/s binary rate and 1 Km radio range, the 915 MHz band that offers ten channels each with 40 Kbit/s rate, and the ISM (Industrial-Scientific-Medical) band (2.4 GHz) that offers 16 channels at 250 Kbit/s rate. The maximum radio communication range is 200 m. All these specifications makes it a flexible data communication standard with great physical capabilities.

At the MAC layer, IEEE 802.15.4 uses CSMA/CA method to control access to the medium for multiple nodes. CSMA/CA is a probabilistic method based on sensing the communication channels before sending data to avoid collision between nodes' frames. Recently, another deterministic MAC method has been added to IEEE 802.15.4 for reliable communications in industrial and mission-critical environments called Time Slotted Channel Hopping (TSCH) method (Dunagan et al., 2008). In this paper, we focus only on CSMA/CA modeling that in conjunction with the radio interface (transceiver) model aims to reduce the energy consumption to the minimum by switching OFF the latter for most of the time.

## 2.1 Operating Modes

IEEE 802.15.4 standard uses four types of frames: (1) beacon frames that are used by coordinator nodes to manage the network, (2) data frames that are used to transfer useful data between nodes, (3) acknowledgment frames that are used to notify the reception of data frames, and (4) command frames that are used to request specific commands such as "join the network".

It offers two operating modes. The first one is the *non beacon-enabled* mode called also unslotted mode where no duty-cycling constraints are specified. There is no synchronization requirements as nodes must stay constantly tuned to the radio activity and continually asking the coordinator node for communication activities on the channels. Energy consumption in this mode should be high because collisions are more frequent. The second mode is the *beacon-enabled* mode called also slotted mode where nodes are synchronized via beacon frames. Transmissions usually start by sending periodic beacon frames by the coordinator containing network information to inform the other nodes about the start, duration and allocation of each slot (see Figure 1 for a description of the Superframe). The performance of the MAC protocol in this mode is higher than in the unslotted mode in terms of throughput, data delivery, reliability and energy consumption because nodes are not required to stay tunned to sense the radio activity, but switch off their radio interfaces until their respective slots arrive. Thus, collisions are reduced to the minimum (without discarding them all) because each node is programmed to send/receive its data in its specific slot.

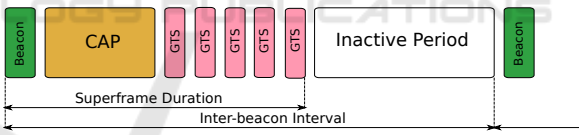


Figure 1: The structure of the Superframe in IEEE 802.15.4.

## 2.2 CSMA/CA Medium Access Method

CSMA/CA avoids collisions between frames sent by different nodes by desynchronizing their respective sending times using a random draw within an interval depending on a *backoff exponent*. Nodes wait for their respective random time after which they check for the availability of the channel. If a node verifies that the channel is available, then it sends its frame but with no guarantee that it will reach the receiver because there may be another node that assesses the channel and sends its frame at the same time. To remedy to this situation, an acknowledgement mechanism is provided to notify the well reception of a particular data frame (Sanabria et al., 2013).

Depending on time representation within the protocol, two types of medium access methods are defined: slotted CSMA/CA where time is divided into

duration-identical periods called *slots* at the beginning of which transmissions can be started, and unslotted CSMA/CA where transmissions can start at any arbitrary moment (after sensing the channel and observing the backoff period). Slotted CSMA/CA is used in IEEE 802.15.4 beacon-enabled MAC protocol mode and unslotted CSMA/CA is used in the non beacon-enabled mode.

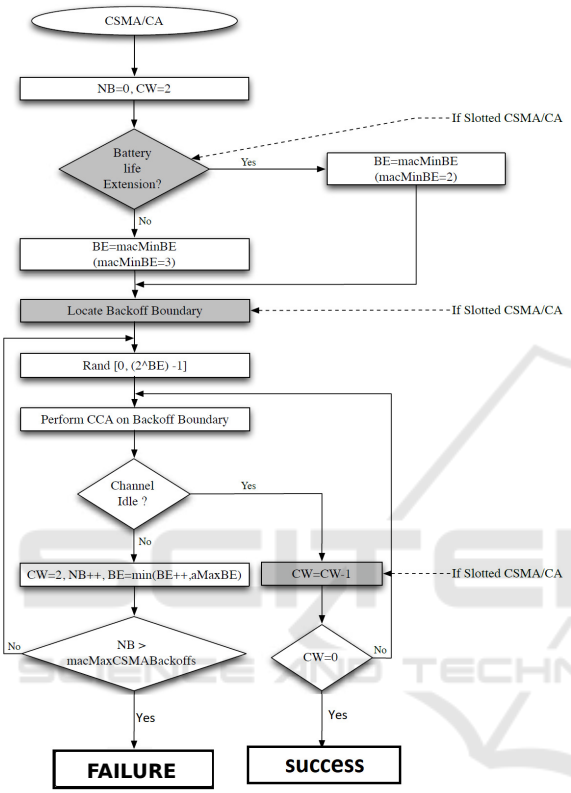


Figure 2: Slotted and unslotted CSMA/CA in IEEE 802.15.4 (Kamgueu, 2017).

Before every frame transmission in slotted CSMA/CA, nodes must execute the following steps:

1. Initializing parameters $(CW, NB, BE)$ to their respective default values (see Table 1 for the definition of these parameters);
2. Drawing a random time (backoff) to desynchronize transmissions;
3. Aligning the beginning of the Unit Backoff Period $(U)$ to that of the Superframe. All the nodes must ensure it remains sufficient time for the backoff period, the two $CW$ periods, the frame transmission and the acknowledgement times if it is requested before the end of the Superframe.
4. At the end of the backoff period, the nodes must check for the channel availability using Clear Channel Assessment method (CCA) that takes 8 symbols. Three possible cases are to be considered at the end of CCA:
   - If the channel is detected busy, then the node updates its parameters as follows: $NB \leftarrow NB + 1, BE \leftarrow min(BE + 1, MACminBE)$ and $CW \leftarrow CW - 1$.
   - If $NB > MACmaxCSMABackoffs$, then the node backs-off the transmission of the current frame. Otherwise, it draws a new backoff time for another attempt.
   - If the channel is detected clear, then the node decreases its CW parameter ($CW \leftarrow CW - 1$). If $CW = 0$, then it sends its frame at the beginning of the next Unit Backoff Period. Otherwise, it executes another CCA after the current Unit Backoff Period.

The above procedure differs in unslotted CSMA/CA in terms of two points:

- The parameters are decreased immediately after the backoff period without waiting for the start of the next Unit Backoff Period because there is no Superframe; and
- Before transmission in unslotted CSMA/CA, channel is assessed only once whereas in slotted CSMA/CA, it is assessed for a double Unit Backoff Period.

Diagram in Figure 2 shows the flow control of both slotted and unslotted CSMA/CA.

Most of the literature about CSMA/CA in IEEE 802.15.4 standard do not use formal methods to evaluate and verify it (Wu et al., 2018), and focus only on experimental tests and simulation-based performance evaluation (Wu et al., 2014). Furthermore, even formal-based ones do not consider joint models of both MAC and physical layers (Kapus, 2017), especially in the context of energy consumption analysis in IoT systems (Groß et al., 2007), (Kauer et al., 2016), (Gawanmeh, 2011). Therefore, in this paper we apply rigorous model-checking method to analyze critical protocol operations in IoT systems such as the IEEE 802.15.4 standard to bring confidence and reliability to them.

## 3 MODEL-CHECKING

Model-checking is a formal method based on the exploration of the whole state space of the system under verification. Assisted by a software tool, this method checks all the possible scenarios and executions of the system in a systematic manner. Qualitative properties such as: safety, liveness, stability, responsiveness,

Table 1: CSMA/CA Notations.

| Term-Notation | Description |
|---|---|
| Unit Backoff Period ($U$) | Time unit = 20 symbols |
| Backoff Exponent ($BE$) | Integer initialized by default to MACminBE=3 |
| Backoff | Random waiting time drawn from interval $[0, 2^{BE}-1] \times U$ |
| Contention Window ($CW$) | This parameter is only used in slotted CSMA/CA and represents the number of time units after the backoff time in which the channel should stay available before transmission |
| $NB$ | Number of current attempts to send, initialized to 0 |
| MACmaxCSMABackoffs | Number of times the channel is found not available before backing-off current frame transmission, initialized to 3 |
| MACmaxFrameRetries | Number of maximum retransmissions of each frame, initialized to 4 |

invariance, possibility, persistence, fairness, etc., as well as quantitative properties such as: time to failure, energy consumption, etc. can be verified by model-checking. It is one of the most powerful verification methods because it uses exhaustive analysis of reactive operational systems that include random and/or probabilistic behavior such as medium access protocols. However, the algorithm behind it is exponential because it explores the whole state space of the system[1]. It generates all the possible executions and verifies the required properties in each of them.

The methodology of model-checking consists of an abstract model of the system $\mathcal{M}$ and a temporal logical formula $\phi$ representing the property to be verified. The model-checker checks whether $\mathcal{M} \models \phi$ or not. It explores the whole state space to prove that $\mathcal{M} \cup \{\neg\phi\}$ has no model. Otherwise, if there is at least one execution path that *satisfies* $\neg\phi$, then it outputs a counter-example, and the user may conclude that the property is not verified for that model.

The abstract model can be expressed as Kripke Automata (Kripke, 1963) or other formalisms (such as Petri Nets, Finite State Automata, Temporal Automata, etc.), and the properties can be expressed in Linear Temporal Logic (LTL), Computation Tree Logic (CTL), Timed Computation Tree Logic (TCTL), etc. whose negation is translated to Büchi Automata for verification in the model.

## 3.1 Linear Temporal Logic

Temporal Logic (TL) is an extension of the classical propositional logic where new temporal connectors are added to express the notion of time. In statements where temporal expressions such as: "before", "after", "next", "never", etc. are used, classical logic constructions are not sufficient to express formulas because of their time-independent interpretations (Pnueli, 1977).

[1]Research efforts focus on optimizing this aspect (Clarke et al., 2001).

The execution of a reactive system can be captured by a sequence of states called *trace*, noted $(\sigma, i) =< s_i \rightarrow s_{i+1} \rightarrow s_{i+2} \rightarrow \cdots >$. Each element $s_i$ in the trace represents the state of the system where a set of logical variables is satisfied. Events represent state change in time, i.e. the evolution of the set of satisfied variables. If this evolution is linear, then Linear Temporal Logic (LTL) formalism is sufficient to model the system. Otherwise, formalisms with richer interpretations of time such as CTL or TCTL are required.

In LTL, a language is a set of formulas constructed from the combination of a finite set of propositional symbols and logical connectors. In addition to the basic logical connectors, a set of temporal connectors is defined as follows:

1. **Next:** $\bigcirc p$ which means that "*proposition p is true in the next instant*".
2. **Always:** $\Box p$ which means that "*proposition p is true in all future instants starting from the current one*".
3. **Finally:** $\Diamond p$ which means that "*proposition p will be finally true at an arbitrary future instant*".
4. **Until:** $p \mathcal{U} q$ which means that "*proposition p remains true in future instants until proposition q will be true*".
5. **Leads-to:** $p \rightsquigarrow q$ which means that "*When proposition p becomes true, then proposition q will be true after that*".

Connectors $\bigcirc$ and $\mathcal{U}$ form a complete minimum set of temporal connectors because we can express the other connectors as follows:

$$\Diamond p \triangleq \top \mathcal{U} p$$

$$\Box p \triangleq \neg\Diamond\neg p$$

$$p \rightsquigarrow q \triangleq \Box(p \rightarrow \Diamond q)$$

The semantic of well-formed formulas of LTL is defined in terms of traces. A system $S$ verifies a specification property `spec` iff every trace of $S$ verifies

spec, i.e. the set of formulas $\Sigma$ representing $S$ is a model for the formula $\phi_{\text{spec}}$ representing spec:

$$\frac{\forall \sigma \in Exec(S) : (\sigma, 0) \models \phi_{\text{spec}}}{\Sigma \models \phi_{\text{spec}}}$$

### 3.2 Model-checker SPIN

SPIN (Simple Promela INterpreter) is an open source tool developed by Gerard J. Holzmann (Holzmann, 2003), and used to verify mission-critical industrial systems written in PROMELA language. PROMELA (PROcess MEta LAnguage) is a specification language for asynchronous and concurrent systems such as communication protocols, with non-deterministic control structures. It has a C-like syntax with two types of finite-state based communication primitives: global variables and communication channels. A PROMELA model can be analyzed by SPIN in two manners: (i) simulation of special cases of the model, and (ii) verification of the whole model given a certain property.

SPIN transforms the PROMELA model into a Kripke structure composed of a map of individual processes and a finite state Kripke automata representing the states and the communication channels. The negation of the LTL formulas are transformed into Büchi automata, and the synchronized product of the two automata is checked whether it is empty or not: if it is empty, then the system models the formula, and thus it verifies the property. Otherwise, SPIN outputs a trace as a counter-example.

SPIN uses an optimized variant of depth-first graph traversal algorithm to explore the state space of the system model called *nested depth-first search*, but reduces first the number of reachable states using an algorithm called *partial order reduction*. Further, it optimizes memory management to represent huge state spaces using *state compression* and *bit-state hashing* (Holzmann, 2003).

## 4 OUR JOINT MAC-PHYSICAL MODEL

Some simulation models of LP-WPAN do not consider realistic energy consumption when using IEEE 802.15.4 standard as a protocol stack. Specifically, they do not handle radio interface ON/OFF state switching mechanism in conjunction with beaconless MAC protocols. Thus, the radio interface is always ON, which does not meet the real situations in state-of-the-art hardware and may provide wrong simulation results in terms of energy consumption and delay metrics.

Therefore, as data transmission/reception is the main energy-consuming activity of battery-powered IoT devices, our model aims to define the states/transitions representing the real energetic characteristics of an IEEE 802.15.4 based IoT node. It is inspired by (Rege and Pecorella, 2016), and represents a generic framework to formally analyze energy consumption models and ensure their correctness. In particular, our model combines the radio communication interface behavior with that of CSMA/CA MAC protocol.

The radio interface in (Rege and Pecorella, 2016) is characterized by the following states:

- TRX_OFF: the transceiver is off;
- TX_ON: the transmitter is on;
- RX_ON: the receiver is on;
- BUSY_TX: the transceiver is sending;
- BUSY_RX: the transceiver is receiving.

This simple model captures all the possible transitions of a radio interface compliant with IEEE 802.15.4 in a two-node communication scenario. However, it does not consider the network effect due to collisions.

The transmitter (respectively receiver) may have three states: off, on and busy. For two communicating nodes node1 and node2: if node1 needs to send a data frame to node2 and their respective radio interfaces are OFF, then node1 should first turn its radio interface ON for transmission (transition TRX_OFF→TX_ON). After that, it sends its data frame to node2 (transition TX_ON→BUSY_TX). For node2 to receive a data frame sent by node1, it must first turn its radio interface ON for reception (transition TRX_OFF→RX_ON). This transition is supported in beacon-enabled mode but requires synchronization between node1 and node2 in non-beacon enabled mode.

After that, node2 changes the radio state to busy for effective reception (transition RX_ON→BUSY_RX). If node2 receives the data frame successfully, then it sends an acknowledgement in the same manner, then both nodes turn their radio interfaces OFF (transitions BUSY_TX→TX_ON→TRX_OFF and BUSY_RX→RX_ON→TRX_OFF respectively).

We enhance this model by handling the network dimension of IoT systems where collisions may occur due to the probabilistic nature of CSMA/CA method. As it is shown in Figure 3, we add a new state called BACKOFF to abstract the network activity that may lead to extra-energy consumption caused by collisions.

We modify the semantics of the state/transition model as follows:

- IDLE: this state is similar to TRX_OFF but it is different from the complete shutdown state. Some state variables are reset and the transceiver is capable to assess the channel and detect energy without using synchronization beacons.
- TX_ON: in this state the transmitter is ON ($trx = tx\_on$) but it is not effectively transmitting. Unlike in model of (Rege and Pecorella, 2016), the transceiver state does not change immediately to TRANSMIT without any energy cost. Before transmitting a data frame, a node with a radio interface compliant with IEEE 802.15.4 *should* first assess the channel whether it is clear or not: if it finds the channel clear ($cca = 0$), then it sends its data frame. Otherwise, it changes its state to BACKOFF to resolve the contention. After the backoff, if it senses the channel (via low-power listening) and finds it clear, then it sends its data frame. Otherwise, the number of attempts is increased at the MAC layer and the current frame is dropped if it exceeds 4.
- TRANSMIT in this state, the node is effectively transmitting its data frame ($trx = sending$). The energy consumption: of the node depends on the time spent in this state and the corresponding voltage. Normally, the radio state should be clear in this state, but if it is not, then a collision is reported to the MAC layer. In this state, the transceiver circuitry may accept another receiving request but drops all the other transmits. Immediately after finishing the transmission, the transceiver returns back to TX_ON state to handle other transmitting/receiving operations if the MAC layer has some frames waiting, then the channel becomes occupied ($cca = 1$). In TX_ON state, if there are no transmitting/receiving operations waiting at the MAC layer, then the transceiver switches to IDLE state. If an acknowledgement is required, then the transceiver may change its state directly to RX_ON without traversing the IDLE state.
- RX_ON: in this state, the transceiver is ready to receive frames but it is not effectively receiving them ($trx = rx\_on$). If it detects a preamble ($preambule = 1$), then it changes its state to RECEIVE for effective reception. If no preamble is detected, then the transceiver returns back to IDLE state. When a frame is passed from the MAC layer to the physical layer, then the transceiver can change its state directly to TX_ON without traversing the IDLE state. Otherwise, it goes to IDLE.
- RECEIVE: in this state, the transceiver is effectively receiving a data or an acknowledgment frame ($trx = receiving$). Normally, a preamble would have been detected before arriving to this state ($preambule = 1$). If this is the case, the frame is passed to the MAC layer to filter it based on the destination address. If the received frame is addressed to the current node, then an acknowledgement frame is sent by MAC layer. If the preamble is missing ($preambule = 0$), then the transceiver reports a collision to the MAC layer and no acknowledgment is sent.
- BACKOFF: in this state, the node waits for the channel to be clear ($trx = backoff$). If the physical layer (via low-power listening) detects that the channel is clear ($cca = 0$), then it changes its state to TX_ON to resume transmission. Otherwise, the number of backoffs is increased and if it exceeds 3, then the current frame is dropped.

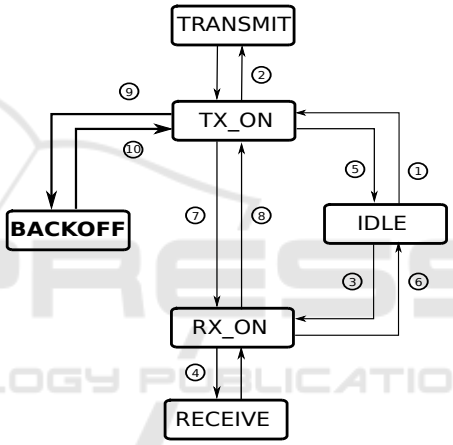


Figure 3: Our joint MAC-physical model.

Table 2 shows the main state variables used in our model: $trx, rmsg, smsg$ and $rid$ are controlled by the physical layer, and $data, ack, xdata, rdata, rack, attempt, b\_attempt$ and $passed$ are controlled by the MAC layer.

## 4.1 Communication Model

Our two-node communication model combines MAC-triggered with physical-triggered events. The Link Layer Control (LLC) protocol passes data sequentially to the MAC layer upon receiving a notification that the MAC process is ready. The latter assigns a unique sequence number to data frames to track those that have not been acknowledged yet, and passes them the physical process. The physical process does not accept data or acknowledgement frames while it is busy. Thus, the MAC layer needs to be notified (using variable $passed$) that its frame has be accepted. After transmission, the MAC layer waits

Table 2: State variables used in our joint MAC-Physical model.

| Layer | Variable | Definition |
|---|---|---|
| PHY | $trx$ | Transceiver state within $\{idle, tx\_on, rx\_on, transmit, receive, backoff\}$ |
| | $rmsg$ | Boolean variable to indicate if the physical layer has received a frame from MAC |
| | $smsg$ | Boolean variable to indicate if the physical layer has received a frame addressed to MAC |
| | $rid$ | Destination address in the received message |
| MAC | $data$ | Sequence number of data frame to send |
| | $ack$ | Sequence number of acknowledgment frame waiting for |
| | $xdata$ | Expected sequence number of the received data frame |
| | $rdata, rack$ | Sequence number of the received data frames and acknowledgement frames, respectively |
| | $attempt, b\_attempt$ | Number of sending attempts (backoffs respectively) of a data frame |
| | $passed$ | Boolean variable to indicate that a frame is passed from MAC layer to physical layer while the transceiver is transmitting or receiving |

for an acknowledgement: if it receives one with the right sequence number, then it notifies the LLC layer. Otherwise, it drops the acknowledgement frame and waits for the timer to expire. After timer expiration, it increases the number of attempts and re-execute the same process. After a maximum number of attempts, the MAC layer backs-off the transmission of the current frame and notifies the LLC.

Similarly, when the MAC layer receives a data frame with the expected sequence number ($xdata$), then it accepts it and sends an acknowledgement with the same sequence number. Otherwise, it sends an acknowledgement frame with $rdata$ as a sequence number.

#### 4.1.1 Transmission

A normal transmission cycle without acknowledgement is as follows:

$$\texttt{IDLE} \rightarrow \texttt{TX\_ON} \rightarrow [\texttt{BACKOFF}]^* \rightarrow \texttt{TX\_ON} \rightarrow \texttt{TRANSMIT} \rightarrow \texttt{TX\_ON} \rightarrow \texttt{IDLE}$$

The sequence of transitions is shown in Figure 3:

1. Transition 1 is enabled by a MAC layer event indicating that a new data frame is available for transmission. The $rmsg$ variable is set to 1 and the MAC layer is notified. The residual energy of the node is updated consequently.
2. Transition 2 is enabled by a physical layer event that indicates the channel is clear for transmission ($cca = 0$). This variable is non-deterministically set by the radio process described in subsection 4.2. As mentioned in (Rege and Pecorella, 2016), this transition does not consume energy at the radio interface level.
3. Transitions 9 and 10 are enabled by a physical layer event that indicates the channel is not clear ($cca = 1$), then becomes clear after the backoff. Transition TRANSMIT $\rightarrow$ TX_ON is internal to the radio interface circuitry and it is not observable by the MAC process. It has zero energy cost.
4. Transition 5 is enabled by a joint MAC-physical event when $rmsg = 0$ and the channel is detected clear (via low-power listening).

When acknowledgements are required, the transmission cycle becomes as follows:

$$\texttt{IDLE} \rightarrow \texttt{TX\_ON} \rightarrow [\texttt{BACKOFF}]^* \rightarrow \texttt{TX\_ON} \rightarrow \texttt{TRANSMIT} \rightarrow \texttt{TX\_ON} \rightarrow \texttt{RX\_ON}$$

Transition 7 may reduce reception delays in some cases (when the sender and receiver are close to each other and are not too loaded).

#### 4.1.2 Reception

A normal reception cycle of a data frame with acknowledgement is as follows:

$$[\texttt{IDLE} \rightarrow \texttt{RX\_ON}]^* \rightarrow \texttt{RECEIVE} \rightarrow \texttt{RX\_ON} \rightarrow \texttt{TX\_ON}$$

The sequence of transitions is shown in Figure 3:

1. Transition 3 is enabled by the Clear Channel Assessment (CCA) function at the physical layer. In non-beacon enabled mode, nodes must be synchronized to wakeup and assess the channel: if it detects energy, then it sets $smsg$ variable to 1 and residual energy is updated consequently.
2. Transition 4 is enabled by a physical layer event when a preamble of a frame is detected ($preambule = 1$). This transition does not consume energy and its reverse transition is integrated in the circuitry of the radio interface.
3. Transition 8 is enabled by a joint MAC-physical event that indicates that a new frame is available for transmission. Variable $rmsg$ is set and the transceiver changes its state directly to TX_ON. In addition, the MAC layer is notified that its frame is accepted for transmission.

Transitions 3 and 6 can be repeated many times as long as no valid preamble is detected, which is an energy-consuming cycle.

A normal reception cycle of an acknowledgement frame does not have transition 8 but has transition 6 instead as shown below:

$$[\texttt{IDLE} \rightarrow \texttt{RX\_ON}]^* \rightarrow \texttt{RECEIVE} \rightarrow \texttt{RX\_ON} \rightarrow \texttt{IDLE}$$

Transition 6 is enabled by a physical layer event ($preambule = 0$) indicating that no valid preamble is detected on the occupied channel ($cca = 1$).

## 4.2 Radio Model

The radio model represents a multiple access channel behavior. It is controlled by three variables: $cca, preambule$ that are set by the radio process and reset by the nodes, and *collision* that is set by the nodes and reset by the radio process.

In a three-node communication scenario shown in Figure 4, transmissions are between node0 and node1, and node2 is set to send data frames to node1 just to induce collisions. This latter does not acknowledge node2's frames. Thus, channel *out*2 is always empty.

A collision is reported by a transmitting node ($collision = 1$) when its transceiver is in state `TRANSMIT` with $rmsg = 1$ and detects that the channel is not clear ($cca = 1$). Similarly, when a receiving node with $smsg = 1$ whose transceiver state is `RECEIVE` and detects no preamble ($preambule = 0$), then a collision is also reported.

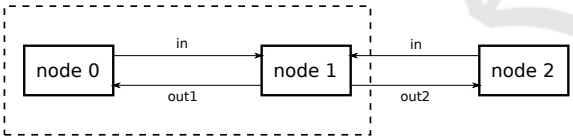


Figure 4: Collision representation in 3-node communication scenario.

Upon detecting a collision, the radio model clears the channels and reset $cca, preambule$ and *collision* variables. It can also inspect the channels and set *cca* when *in* or *out*1 are not empty, and set *preambule* only when channel *in* is filled by node0 or channel *out*1 is filled by node1. *preambule* is reset by receiving nodes upon successful reception.

# 5 IMPLEMENTATION & VALIDATION

We implement our joint MAC-physical model in PROMELA. As we show in Figure 5, each node in three-node communication scenario consists of 3 independent processes: app, mac and phy representing the LLC, MAC and physical layer protocols, respectively.

The processes communicate between each others using the following *SPIN channels*:

- `a2m` channel: it has capacity 1, and it is used to notify the MAC process that a new data message is available.
- `m2a` channel: it has capacity 1, and it is used to notify the LLC (app) process that the MAC process is ready to accept new data frames. In case when a data frame transmission fails, the MAC process notifies the LLC process using a negative value (-1).
- `m2p` channel: it has capacity 1, and it is used to pass data and acknowledgement frames from the MAC layer to the physical layer process with the corresponding sequence number and the appropriate destination address.
- `p2m1` channel: it has capacity 1, and it is used to notify the MAC process about the admission of data/acknowledgement frames by the physical layer process while the transceiver is busy.
- `p2m2` channel: it has capacity 1, and it is used to pass received data/acknowledgement frames to the MAC layer process.
- `in` channel: it has capacity 2 to make collisions possible, and it is used to receive data/acknowledgement frames from the radio process.
- `out` channel: it has capacity 2 to be able creating collisions, and it is used to transmit data/acknowledgement frames to the radio process.

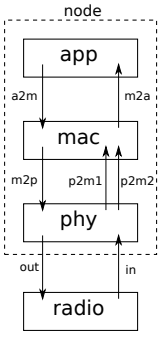


Figure 5: Model implementation.

The PROMELA source models of the radio, MAC and physical processes are shown in Figures 7, 9 and 12, respectively.

Additionally, Figure 6 shows the state/transition automaton of the LLC process that is composed of 7 states and 9 transitions. State `S13` is the initial state

from which all the possible executions start. The automaton is quite simple because the other functions of LLC such as traffic control, frames numbering, error control, etc. are not considered in our model. We assume that the MAC layer handles one frame at a time and all frame buffering and traffic regulation operations are left for the upper layers.

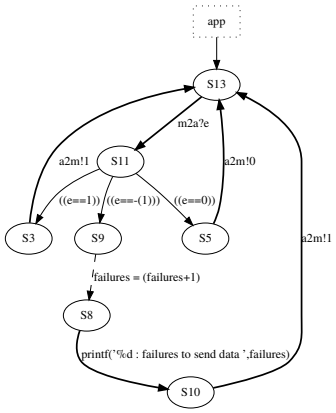


Figure 6: State/Transition automaton of the LLC process.

The state/transition automata of the radio, MAC and physical processes generated by SPIN are depicted in Figures 8, 10 and 11.

The state/transition automaton of the radio process is shown in Figure 8, and it has 21 states and 29 transitions.

```
1  proctype radio(chan out1; chan out2; chan in) {
2    mtype:msg m;
3    int s;
4    do
5    assert(empty(out2));
6    :: collision -> // clear channels
7      if
8      :: nempty(in) -> in?m,s;
9      :: nempty(out1) -> out1?m,s;
10     :: nempty(out2) -> out2?m,s;
11     :: (nempty(in) && nempty(out1)) -> atomic{in?m,s;
         out1?m,s;}
12     :: (nempty(in) && nempty(out2)) -> atomic{in?m,s;
         out2?m,s;}
13     :: (nempty(out1) && nempty(out2)) -> atomic{out1?m,
         s; out2?m,s;}
14     :: (nempty(in) && nempty(out1) && nempty(out2)) ->
         atomic{in?m,s; out1?m,s; out2?m,s;}
15     fi
16     cca = 0; preamble = 0; collision = 0; // reset
         variables
17   :: (cca && nfull(in) && empty(out1)) -> cca = 1;
       preamble = 1;
18   :: (cca && empty(in) && nfull(out1)) -> cca = 1;
       preamble = 1;
19   od
20 }
```

Figure 7: PROMELA source model of the radio process.

The automaton of the MAC process has 26 states and 37 transitions: state `S1` is the initial state from which the MAC process execution starts and state `S50` is the one to which all the possible executions return. It has 6 possible non-deterministic transitions.

Table 3: Energy consumption parameters.

| Parameter | Value (unit) | Definition |
|---|---|---|
| `MAX_EN` | 1.000.000 | Maximum initial residual energy |
| `TXON` | 1 | Energy consumption in transition `IDLE` → `TX_ON` |
| `RXON` | 1 | Energy consumption in transition `IDLE` → `RX_ON` |
| `TXRX` | 5 | Energy consumption in transition `TX_ON` → `RX_ON` |
| `RXTX` | 5 | Energy consumption in transition `RX_ON` → `TX_ON` |
| `TX_DATA` | 100 | Energy consumption for transmitting a data frame |
| `TX_ACK` | 20 | Energy consumption for transmitting an acknowledgement frame |
| `RX_DATA` | 80 | Energy consumption for receiving a data frame |
| `RX_ACK` | 10 | Energy consumption for transmitting an acknowledgement frame |

The state/transition automaton of the physical process has 64 states and 86 transitions. The energy consumption parameters used in our physical model implementation are shown in Table 3. These values are approximately based on those in (Rege and Pecorella, 2016) which depend on state duration and voltage in a particular transceiver hardware (AT89RF231), but can be easily adapted to another one for more accurate model simulation.

## 6 CONCLUSION & FUTURE WORK

In this paper, we have proposed a formal joint MAC-physical model of a radio interface compliant with IEEE 802.15.4 standard to be used as a framework for energy consumption simulation of IoT systems. Our model generalizes the one proposed in (Rege and Pecorella, 2016) by considering network dimension and handling extra energy consumption due to collisions and contention resolution procedure. It captures all the possible transitions induced by the radio in-

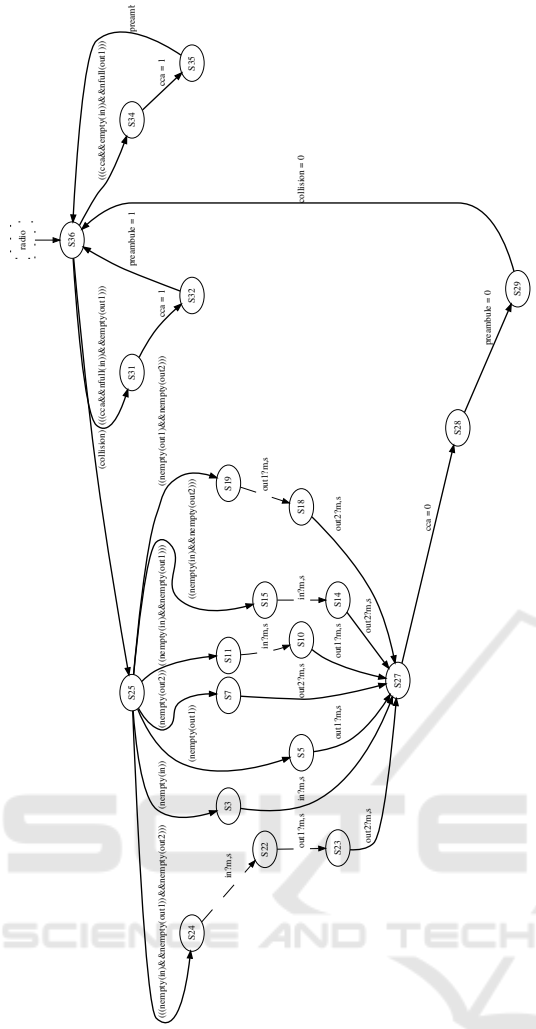


Figure 8: State/Transition automaton of the radio process.

```
1  proctype mac(int id; chan a2m; chan m2a; chan m2p; chan
       p2m1; chan p2m2) {
2    int data = 0;
3    int ack = 0;
4    int xdata = 1;
5    int event;
6    int rdata, rack;
7    int attempt = 0;
8    mtype:msg msgT;
9    int seq;
10   bool passed = 0;
11
12   m2a!1;
13   do
14   :: a2m?event ->
15     if
16     :: event -> data = (data + 1)%MAX; attempt = 0;
17     :: else -> skip;
18     fi
19   :: (data == (ack + 1)) -> m2p!DATA(data); ack = (ack
         + 1)%MAX;
20   :: p2m1?msgT,seq, passed ->
21     if
22     :: !passed -> m2p!msgT(seq);
23     :: else -> skip;
24     fi
25   :: p2m2?DATA(rdata) ->
26     if
27     :: (rdata == xdata) -> m2p!ACK(xdata); xdata = (
         xdata + 1)%MAX;
28     :: else -> m2p!ACK(rdata);
29     fi
30   :: p2m2?ACK(rack) ->
31     if
32     :: (rack == ack) ->m2a!1;
33     :: else -> skip;
34     fi
35   :: timeout ->
36     if
37     :: passed -> attempt = (attempt + 1);
38       if
39       :: ((attempt < MAX_ATTPT) || (id == 2)) -> m2p!
          DATA(data);
40       :: else -> attempt = 0; printf("Failure to send
         data/ack: %d\n", data); m2a!-1;
41       fi
42     :: else -> skip;
43     fi
44   od
45 }
46
```

Figure 9: PROMELA source model of the MAC layer process.

terface ON/OFF state switching and combines them with the internal functions of the CSMA/CA medium access method.

We have implemented our model in PROMELA language and generated the automata of the MAC, physical and radio processes.

As a future work, we consider to refine the energy consumption parameters in concordance with the state-of-the-art transceivers hardware and formulate specification properties to be verified by our model such as:

1. The number of frame retransmissions: "*if* node0 *rejects its frame, then the number of unsuccessful transmissions has reached the bound value of retransmissions*",
2. Hidden senders: "*if 2 hidden senders, i.e.,* node0 *and* node2 *have started their transmissions simultaneously, then a collision will occur and will be detected by the receiver*",
3. Maximum probability of at least $k$ collisions, and
4. Minimum energy consumption of at most $k$ collisions.

We also consider to simulate some special cases of two-node and three-node communication scenarios to evaluate our model in terms of the number of data transmission failures, the number of collisions and the amount of energy consumption consumed in the contention resolution procedure.

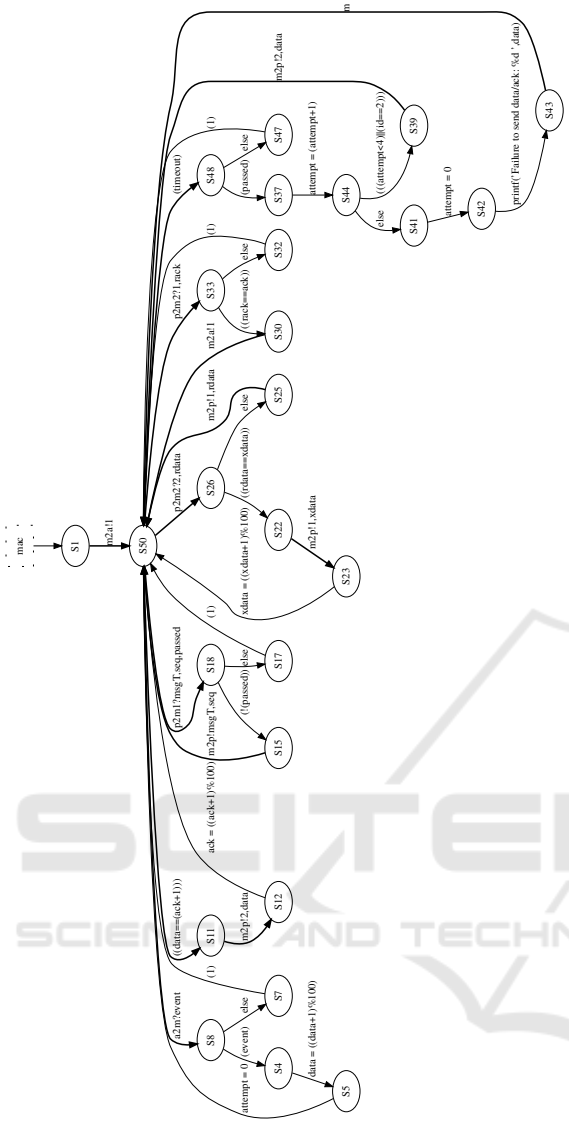


Figure 10: State/Transition automaton of the MAC layer process.

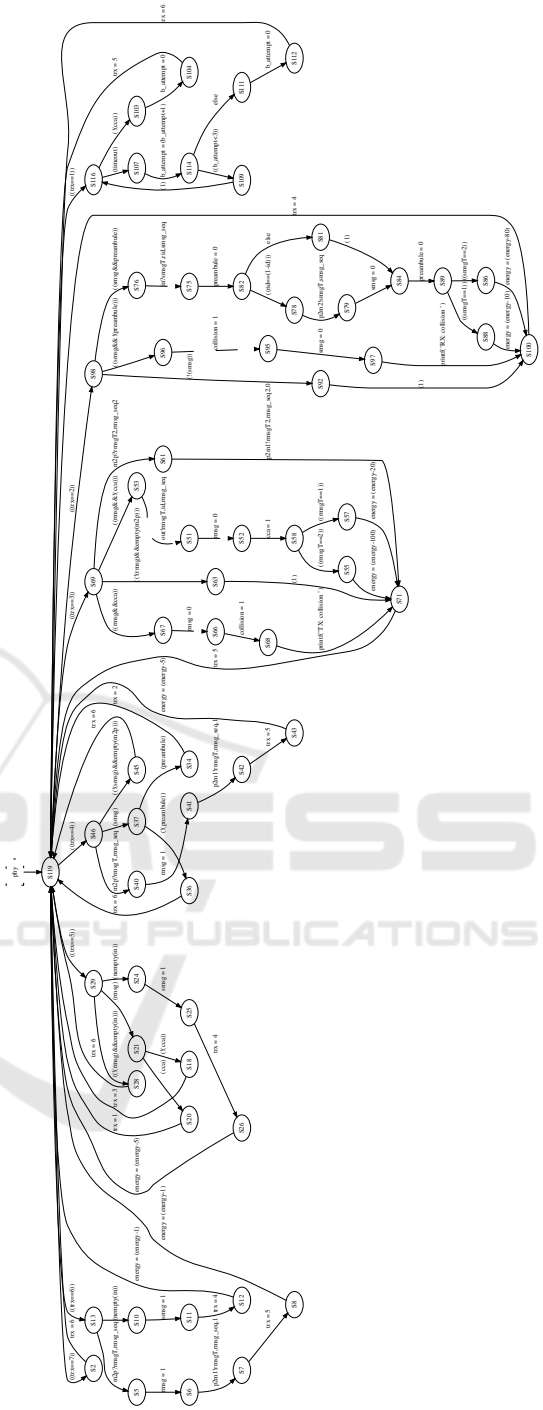

Figure 11: State/Transition automaton of the physical layer process.



# REFERENCES

Agerstam, M., Colby, R., Donohue, P., Meyer, S. J., and Sanghadia, P. (2018). Reduce iot cost and enable scaling through open wireless sensor networks. White paper, Intel Corp.

Behera, R. K., Gupta, S., and Gautam, A. (2015). Big-data empowered cloud centric internet of things. In *Proceedings of the International Conference on Man and Machine Interfacing*, MAMI.

Burdett, A. (2015). Ultra-low-power wireless systems: Energy-efficient radios for the internet of things. *IEEE Solid-State Circuits Magazine*, 7(2):18–28.

CISCO (2015). Fog computing and the internet of things: Extend the cloud to where the things are. Technical report, CISCO Corp.

Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2001). Progress on the state explosion problem in model checking. *Informatics. 10 Years Back. 10 Years Ahead, LNCS 2000*, pages 176–194.

Dunagan, J. D., Bahl, P., and Chandra, R. (2008). Slotted seeded channel hopping for capacity improvment in wireless networks.

Gawanmeh, A. (2011). Embedding and verification of zigbee protocol stack in event-b. *Procedia Computer Science*, 5:736–741.

Groß, C., Hermanns, H., and Pulungan, R. (2007). Does clock precision influence zigbee's energy consumptions? In Tovar, E., Tsigas, P., and Fouchal, H., editors, *Principles of Distributed Systems*, pages 174–188, Berlin, Heidelberg. Springer Berlin Heidelberg.

Holzmann, G. J. (2003). *The SPIN MODEL CHECKER: Primer and Reference Manual*. Addison-Wesley Pearson Education, first edition.

Kamgueu, P. O. (2017). *Configuration Dynamique et routage pour l'internet des objets*. PhD thesis, University of Lorraine (France), University of Yaoundé.

Kapus, T. (2017). Using prism model checker as a validation tool for an analytical model of ieee 802.15.4 networks. *Simulation Modelling Practice and Theory*, 77:367 – 378.

Kauer, F., Köstler, M., Lübkert, T., and Turau, V. (2016). Formal analysis and verification of the ieee 802.15.4 dsme slot allocation. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 140–147, New York, NY, USA. ACM.

Kripke, S. (1963). Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94.

Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., and Zhao, W. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS'77, pages 46–57, Washington, DC, USA. IEEE Computer Society.

Rege, V. and Pecorella, T. (2016). A realistic mac and energy model for 802.15. 4. In *Proceedings of the Workshop on ns-3*, pages 79–84. ACM.

Sanabria, L., Faridi, A., adn Jaume Barcelo, B. B., and Olivier, M. (2013). Future evolution of csma protocols for the ieee 802.11 standard. In *Second IEEE ICC Workshop On Telecommunication Standards: From Research to Standards*.

Society, I. C. (2017). Ieee 802.15.4-2017 - ieee standard for low-rate wireless networks. Electronic.

Wu, B., Lemmon, M. D., and Lin, H. (2018). Formal methods for stability analysis of networked control systems with ieee 802.15.4 protocol. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, 26(5):1635–1645.

Wu, S., Mao, W., and Wang, X. (2014). Performance study on a csma/ca-based mac protocol for multi-user mimo wireless lans. *IEEE Transactions on Wireless Communications*, 16(6):3153–3166.

```
proctype phy(int id; chan in; chan out; chan m2p; chan p2m1;
      chan p2m2) {
  mtype:state trx = start;
  int energy = MAX_EN;

  bool rmsg = 0;
  int rmsg_seq = 0, rmsg_seq2 = 0; // seq. No. of current data/
      ack received from MAC
  mtype:msg rmsgT;
  mtype:msg rmsgT2; // message received from MAC
  int b_attempt =0;
  bool smsg = 0;
  int smsg_seq = 0; // seq. No. of current data/ack received to
       MAC
  mtype:msg smsgT; // message sent to MAC
  int rid;

  do
  :: trx == start -> trx = idle
  :: trx == idle ->
    if
    :: m2p?rmsgT,rmsg_seq ->
      rmsg = 1; p2m1!rmsgT,rmsg_seq,1; trx = tx_on;
      energy = energy - TXON;
    :: nempty(in) -> smsg = 1; trx = rx_on;
       energy = energy - RXON;
    fi
  :: trx == tx_on ->
    if
    :: rmsg ->
      if
      :: !cca -> trx = transmit;
      :: cca -> trx = backoff;
      fi
    :: (nempty(in)) -> smsg = 1; trx = rx_on;
       energy = energy - TXRX;
    :: (!rmsg && empty(in)) -> trx = idle;
    fi
  :: trx == rx_on ->
    if
    :: smsg ->
      if
      :: preamble -> trx = receive;
      :: !preamble -> trx = idle;
      fi
    :: m2p?rmsgT,rmsg_seq ->
      rmsg = 1; p2m1!rmsgT,rmsg_seq,1; trx = tx_on;
      energy = energy - RXTX;
    :: (!smsg && empty(m2p)) -> trx = idle;
    fi
  :: trx == transmit ->
    if
    :: (rmsg && !cca) ->
      atomic{out!rmsgT,id, rmsg_seq; rmsg = 0; cca = 1;}
      if
      :: rmsgT == DATA -> energy = energy - TX_DATA;
      :: rmsgT == ACK -> energy = energy - TX_ACK;
      fi
    :: m2p?rmsgT2,rmsg_seq2 -> p2m1!rmsgT2,rmsg_seq2,0;
    :: (!rmsg && empty(m2p)) -> skip;
    :: (rmsg && cca) -> atomic{rmsg = 0; collision = 1;}
       printf("TX: collision\n");
    fi
    trx = tx_on;
  :: trx == receive ->
    if
    :: (smsg && preamble) ->
      atomic{in?smsgT,rid,smsg_seq; preamble = 0;}
      if
      :: rid == 1-id -> p2m2!smsgT,smsg_seq; smsg = 0;
      :: else -> skip;
      fi
      preamble = 0;
      if
      :: smsgT == DATA -> energy = energy - RX_DATA;
      :: smsgT == ACK -> energy = energy - RX_ACK;
      fi
    :: !smsg -> skip;
    :: (smsg && !preamble) -> atomic{collision = 1; smsg = 0;}
       printf("RX: collision\n");
    fi
    trx = rx_on;
  :: trx == backoff ->
      do
      :: !cca -> b_attempt = 0; trx = tx_on; break;
      :: timeout ->
        b_attempt = (b_attempt + 1);
        if
        :: (b_attempt < MAX_BACKOFFS) -> skip
        :: else ->  b_attempt = 0; trx = idle; break;
        fi
      od
  od
}
```

Figure 12: PROMELA source model of the physical layer process.