Formal Modeling of Greedy Behavior in Secure Internet of Things Networks

Yassine Boufenneche*, Nawel Gharbi*, Rafik Zitouni‡ and Laurent George**
Department of Computer Science*, USTHB, Algiers, Algeria
SIC Lab‡, ECE Paris, Paris, France
LIGM**, UPE/ESIEE Paris, Paris, France
Email: yboufenneche@usthb.dz, ngharbi@usthb.dz, rafik.zitouni@ece.fr, laurent.george@esiee.fr

***Abstract*—In the Internet of Things (IoT) paradigm, objects should communicate through secure wireless channels and would be able to connect to the Internet. IEEE 802.15.4e and its old versions are the de-facto standards for Wireless Sensor Networks (WSNs) based IoT. IEEE 802.15.4e and its old versions define a slotted CSMA-CA medium access control and physical layer of WSNs. The IoT may experience a multitude of malicious attacks. One critical attack that threatens the network availability is the greedy behavior. In this paper, we propose a UPPAAL model to describe greedy and sane behaviors of IoT devices. We are interested in the contention access period of the beacon-enabled mode of the slotted CSMA-CA. The developed model shows that we can assess and identify both honest and greedy devices. However, it is the first step towards thwarting this attack.**

***Index Terms*—Internet of Things (IoT), Wireless Sensor Networks, Security, Greedy Behavior, UPPAAL Modeling, IEEE 802.15.4e, Slotted CSMA-CA**

I. Introduction

Internet of Things is a new wireless communication paradigm. Physical objects called Things with unique identities can connect to the Internet. However, they do not own the same capacities as traditional electronic equipment. Many of these objects have very restrained resources such as energy, computation and storage [1]. Internet of Things builds on top a set of enabling technologies [2], [3]. IEEE 802.15.14e [4] is a standard that specifies PHYsical (PHY) and Medium Access Control (MAC) layers for several technologies adopted by the Internet of Things.

Indeed, communication nature in wireless networks makes them very vulnerable. They may undergo both internal and external attacks. Malicious nodes participating in the network can trigger the former. Damage caused by that nodes can be hard because of their perfect knowledge of the network. One kind of the most dangerous internal attacks is the greedy behavior. A greedy node does not respect protocol rules. It deliberately alters some configurations, like backoff parameters in a channel access method, to gain bandwidth at the expense of honest nodes [5]. Such behavior can be harmful on the overall network performance [6] and may engender famine situations to honest nodes [7].

In view of the above considerations, securing the Internet of Things against greedy behavior is a fundamental task. The usual security mechanisms are not applicable to the Internet of Things because of the limitations of nodes' resources [8]. New techniques for detecting and preventing attacks are needed. This piece of work is a first step towards detecting and thwarting greedy behavior in the Internet of Things. The contributions of this paper are twofold. First, we develop a formal model that provides a way to quickly assess the performance of an IoT network. Second, we consider the greedy behavior attack in an IoT network. This way, we can help researchers to study the impact of this attack and provide insights to develop countermeasures.

The rest of this paper is organized as follows. We summarize some related works in section II. Then, we give an overview of IEEE 802.15.4 protocol. We higlight the need for upgrading IEEE 802.15.4e protocol in section III. Next, we give a short description of the UPPAAL tool in section IV. After that, we expose and explain our proposed model in section V. Finally, we conclude the paper, and we outline our future works in section VI.

II. Related Work

The greedy behavior has been widely studied in wireless networks [6], [7], [9]–[18], but it is relatively new in IoT networks. Several techniques have been developed for detection and prevention, including statistical approaches, formal methods, game theory or even fuzzy logic.

In [6], [9], [10] and [11] statistical approaches have been proposed. Authors in [6] proposed a detection system based on linear regression to find greedy nodes at the 802.11 MAC layer. This method relies on the presence of correlation between the different nodes' access times to the channel. In [9], [10], authors combined both linear regression and watchdog concepts for detecting greedy nodes in VANETs. Linear regression allows distinguishing between a VANET under attack and a safe one. A watchdog tool supervises some connection parameters and identifies the greedy nodes. Authors in [11] proposed a non-parametric statistical method that yields the detection of selfish nodes in a network-coding-aided scenario through the processing of control packets.

Several other researchers opted for formal methods. In [12], authors dealt with greedy behavior in WSN over IEEE 802.15.4 standard, using Time Petri Nets. In [13], authors coupled timed automata and Markov chains to investigate the end-to-end delay in an 802.11 network with the presence of greedy nodes. Authors in [14] used a Markov chain model to analyze the performance of an 802.11 wireless network

including various types of greedy nodes manipulating the backoff parameters. In [15], authors used Time Petri Net to study the impact of greedy behavior on the network performance and energy efficiency in an 802.15.14 network using the non-slotted CSMA-CA. Authors in [16] employed a Markov chain model to study the impact of selfish backoff attack on the throughput in an 802.15.14 network. In [7], timed automata were employed to model the greedy behavior in an 802.15.4 network. We point out that differently from this paper, the model in [7] considers the nonbeacon-enabled mode where nodes compete to access the channel through non-slotted CSMA-CA protocol.

Works in [17], [18] rely on game theory. Authors in [17] modeled an 802.15.4 network as a dynamic game. Nodes are modeled as players with the ability to modify their backoff parameters to improve their throughput and energy consumption in a misbehaving scenario. In [18], authors proposed collaborative-based tit-for-tat strategies to detect greedy nodes modifying the size of the contention window in an 802.11 CSMA-CA protocol.

Although the works in [12], [15] and [7] have considered the greedy behavior in an 802.15.4 network, none addressed this problem in the beacon-enabled mode of such a network. In this paper, we characterize and study the greedy behavior during the contention access period of an IoT network operating over the IEEE 802.15.4e standard.

III. From IEEE 802.15.4 to 802.15.4e Standard

The IEEE 802.15.4 [19] standard defines PHY and MAC sublayer for low-rate, low-power and low range communications between resource-constrained devices. It offers two communication modes: a *nonbeacon-enabled* mode and a *beacon-enabled mode*. Considering the latter, a coordinator uses a superframe (see Fig. 1) bounded by the transmission of two beacons. It may have an active period and an inactive period, in which the coordinator may enter in a low power mode. The beacon interval (BI) separates two successive beacons, and the superframe duration (SD) represents the active period length. The active period may be split into a contention access period (CAP) and a contention free period (CFP). Unfortunately, this standard suffers from several drawbacks, such as unbounded delays in the CAP period thus unbounded delay to obtain a Guaranteed Time Slot (GTS) in the CFP, high packet dropping rate, along with vulnerability to interferences and multi-path fading [20]. The IEEE 802.15.4e standard has been developed to address those shortcomings.

IEEE 802.15.4e [4] and its recent version [21] were released with general functional improvements and additional MAC modes, especially the Channel Sequence Frequency Hopping (TSCH). TSCH combines slotted access with multi-channel and channel hopping capabilities. Accordingly, it can provide predictable latency, high network capacity, while keeping very low duty cycles [22]. Furthermore, it reduces the effects of interference and multipath fading considerably, reflecting on communication reliability.

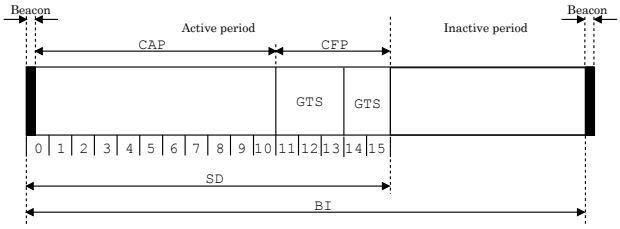

Fig. 1. IEEE 802.15.4 superframe structure

In TSCH mode, time is structured in repeated stotframes. Each slotframe comprises a fixed number of time slots. One timeslot is long enough for a node to transmit a packet to a neighbor and receive its acknowledgment [23], [24]. Multiple communications are achievable at the same timeslot and a different channel offset. At every timeslot, each node in the network either transmit, receive or sleep. A TSCH schedule tells each node what to do in each slot. A slot is a time offset combined with a channel offset. It can be shared or dedicated to a couple of neighbors. IEEE 802.15.4e is not in charge of the building of such schedule; that task is left to upper layers. That is why the IETF created the 6TiSCH working group. It aims at standardizing the way a schedule is built.

IEEE 802.15.4e uses a CSMA-CA mechanism to manage concurrent access to the channel. Fig. 2 shows three variants of this mechanism: *slotted CSMA-CA*, *non-slotted CSMA-CA* and *TSCH CSMA-CA*. The first one is used within the CAP whenever the beacon-enabled mode is activated. The second one regulates concurrent access in the nonbeacon-enabled mode, while the last one enables the access to shared slots in the TSCH mode. Below, we explain the operation of the slotted CSMA-CA.

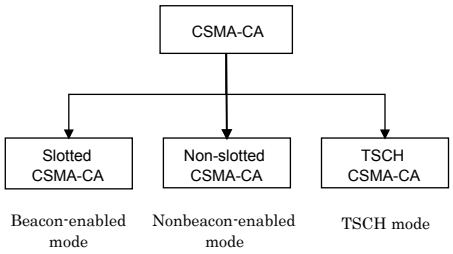

Fig. 2. IEEE 802.15.4e CSMA-CA variants

Slotted CSMA-CA Algorithm

The slotted CSMA-CA algorithm is depicted in Fig. 3 It uses units of time called backoff periods. The first step is the initialization of three variables: *NB*, *CW* and *BE*. *NB* is the number of times the algorithm has backed off for the current transmission. *CW* is the contention window length. It specifies how many times the channel should be idle before starting the transmission. *BE* determines the number of backoff periods a device must wait before attempting the channel assessment. After variables initialization, the algorithm should locate the next backoff period boundary.

In the second step, a delay of a random number of backoff periods in the range of $[0, 2^{BE} - 1]$ will be applied. If the ongoing CAP expires before the end of the backoff delay, the backoff countdown should be paused, and resumed by the start of the next CAP. It is worth noting that, once the random delay is elapsed, the MAC sublayer proceed to frame transmission only if there is enough time in the CAP to carry out the entire transaction (two clear channel assessments (CCA), frame transmission and any acknowledgment reception) [21]. If the MAC sublayer cannot proceed, it should wait for the next CAP and apply a further random backoff delay.

In the case where the MAC sublayer can proceed, the node applies two CCAs (because *CW0* equals 2) before transmitting. If the channel is found idle, the node transmits the frame, and the algorithm ends with success. If not, *BE* and *NB* are incremented by one, and the algorithm goes back to the second step. This process repeats until either the frame is successfully transmitted or until *NB* reaches a specific maximum value. In the latter case, the algorithm ends with a failure, that is, the frame is dropped.

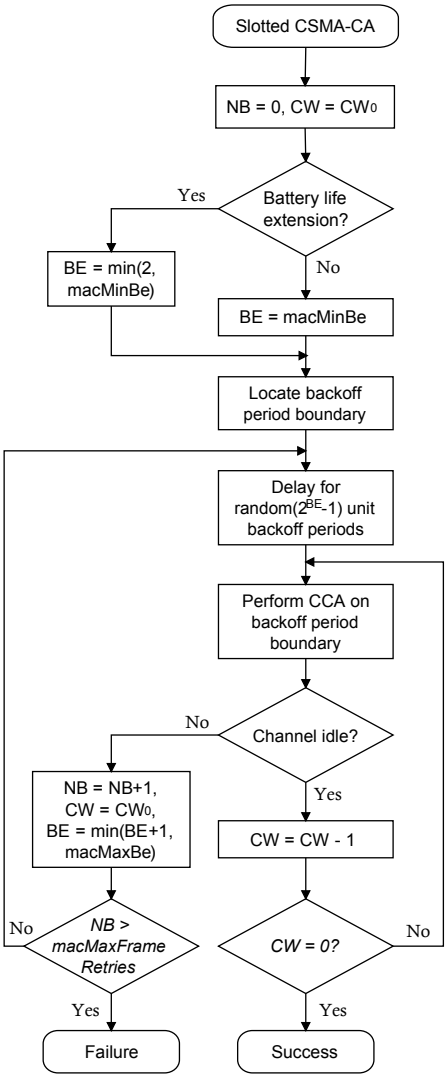

Fig. 3. Slotted CSMA-CA algorithm

IV. UPPAAL Tool

Timed automata [25] are a formal notation which allows the modeling and the verification of real-time systems behavior [26]. A timed automaton is an extension, with real-valued variables, of a finite automaton. A finite automaton is a graph containing a finite set of locations and a finite set of labeled edges. Clock constraints (invariants and guards) restrict the automaton's behavior. An invariant assigns to each location a sojourn condition which may be true. A transition represented by an edge can be taken when the clocks' values satisfy the guard labeled on the edge. Clocks may be reset to zero when a transition is taken [27]. The state of a timed automaton consists of the current location and the current clocks' values.

UPPAAL is a toolbox for simulation and verification of timed automata. A system is modeled in UPPAAL as a network of timed automata (processes) extended with additional features, such as templates, variables, channels, selection, etc. Templates serve to instantiate many processes of the same automaton. A channel *c* is declared as *chan c*. One edge labeled with *c!* synchronizes with another one labeled with *c?*. A broadcast channel *c* is declared as *broadcast chan c*. One sender *c!* synchronizes with all receivers labeled with *c?*. A channel can be declared urgent using the keyword *urgent*. No delay will occur if a synchronization transition is enabled on an urgent channel. A *select label* is a list of *variable:type*; the variable will take a non-deterministic value in the range of its type.

V. Proposed Model

The work discussed in this section is a UPPAAL modeling of the contention access period in an IEEE 802.15.4 star network. *N* nodes communicate among a shared wireless medium with a single channel. They compete to access the channel using the slotted CSMA-CA algorithm. For the sake of simplicity, we assume that all superframes, all beacon intervals and all frames have the same lengths. We also assume that nodes always have frames to transmit during the contention access period.

Fig. 4 and Fig. 5 illustrate the model, which consists of a network of two automata templates: *Medium* and *Node*. The first one models the wireless medium status, while the second one describes the behavior of nodes transmitting frames during the CAP. Templates synchronize with each other using channels. We declared two arrays of channels and one broadcast channel as follows:

$chan\ BeginSend[N];$
$chan\ EndSend[N];$
$broadcast\ chan\ collision;$

TABLE I summarizes the symbols used in the two templates. Before exhibiting the models of the medium and the nodes, we describe first the difference between honest and greedy nodes in the following subsection.

A. Honest and greedy nodes

A honest node follows strictly the slotted CSMA-CA protocol rules. Consequently, it respects the values of the

TABLE I
LIST OF USED SYMBOLS

Symbol	Definition
NB	Number of times the algorithm has backed off for the current frame transmission
CW	Contention window length
CW0	Initial value of *CW* (2 by default)
BE	Backoff exponent
minBe	Minimum value of *BE* (3 by default)
maxBe	Maximum value of *BE* (5 by default)
bb(int n)	A function that locates the next backoff period boundary after the value *n*
backoff(int b)	A function that calculates the backoff delay according to parameter *b*
reset()	A function that resets the values of *NB*, *CW* and *BE*
interval1	Interval $[0, 1]$, *i.e.*, $[0, 2^1 - 1]$
interval2	Interval $[0, 3]$, *i.e.*, $[0, 2^2 - 1]$
interval3	Interval $[0, 7]$, *i.e.*, $[0, 2^3 - 1]$
interval4	Interval $[0, 15]$, *i.e.*, $[0, 2^4 - 1]$
interval5	Interval $[0, 31]$, *i.e.*, $[0, 2^5 - 1]$
TaT	Turnaround time
BCD	Backoff delay
SD	Superframe duration
BI	Beacon interval
CAP	Contention access period duration
CFP	Contention-free period duration
BL	Beacon length
ET	Elapsed time in the current superframe
RT	Remaining time in the CAP
WT	Waiting time before assessing the channel
CCA	Clear channel assessment duration
BP	One backoff period duration
TTran	Time needed for the entire transaction
FTD	Frame transmission duration (including ACK)
maxBackoffs	Maximum value of *NB*
N_COL	Number of collisions
N_SUCCESS	Number of successful transmissions
N_ERR	Number of errors occurred when *NB* exceeds the maximun value
chIdle	Wireless medium status (idle or busy)

parameters defined in the IEEE 802.15.4e standard. Regarding a greedy node, it would reduce its waiting time before transmitting frames. To do so, it could falsify the initial contention window length (*CW0*), the minimum value of *BE* (*minBe*) and the maximum value of *BE* (*maxBe*). For instance, by replacing its *CW0* value by one, which is equal to two in the standard, a greedy node performs only one CCA analyze rather than two. Clearly, this behavior increase collision risks. Reducing *minBE* and *maxBE* values enables a greedy node to compute and apply a small backoff delay. This could result in an increase of the delay of honest nodes.

B. Medium Template

This template is similar to the one used in [7]. It models the channel status; the channel can be either idle or busy. The initial location is *Idle*, indicating that the channel is initially idle. If a node *e* starts a new transmission, *i.e.*, this template receives *BeginSend[e]?* from the sending node, then, the control moves to location *Propagation*. This location interprets the signal propagation. Invariant $x <= TaT$ and guard $x == TaT$ mean that the control remains on that location for a period equal to *TaT*. After that, the control moves to location *Busy*. Any new transmission started by another node *a* while control is still on location *propagation* or on location *Busy* arises a collision. In that case, this template triggers a *collision!* event, the number of collisions is incremented by one, and the control returns to location *Idle*. If there is no collision, the medium turns to status busy. The control stays on location *Busy* until the end of the frame transmission, *i.e.*, until the template receives *EndSend[from]?*. After that, *N_SUCCESS* is increased by one, and the control moves to location *Idle* again, and thus, the medium is ready for a new transmission.

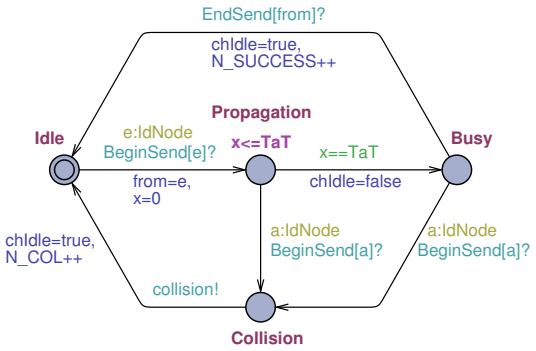

Fig. 4. Medium template

C. Node Template

The template *Node* describes the behavior of both honest and greedy nodes. It has four arguments: the node identifier (*id*), the minimum value of *CW* (*CW0*), the minimum value of *BE* (*minBe*) and the maximum value of *BE* (*maxBe*). We instantiate the template as many times as the nodes' number. To simulate honest nodes, we assign true values to arguments *CW0*, *minBe* and *maxBe* (*CW0 = 2, minBE = 3* and *maxBe = 5*). Besides, to simulate greedy nodes, we assign fake values to those arguments. For example, the lines below give the UPPAAL code used to instantiate respectively one honest node, one greedy node cheating with *CW0* (1 rather than 2), one greedy node cheating with *minBe* (1 rather than 3) and one greedy node cheating with *maxBe* (3 rather than 5). The first argument denotes the node identifier.

$node0 = Node(0, 2, 3, 5);$
$node1 = Node(1, 1, 3, 5);$
$node2 = Node(2, 2, 1, 5);$
$node3 = Node(3, 2, 3, 3);$

By creating the desired number of honest and greedy nodes, we can launch the simulation in the UPPAAL tool. At the end of the simulation, we obtain several performance measures, such as the delay of honest and greedy nodes, number of collisions, the number of transmission errors, etc.

We explain the evolution of the template state as follows. Time is modeled by two clocks: *x* and *y*. The initial location is *Sync* modeling the synchronization of all nodes with the beacon reception; nodes wait for the backoff period boundary that comes after the beacon. Thereafter, the transition to

location *pre_backoff* fires, backoff parameters (*CW0*, *NB* and *BE*) are initialized and the elapsed time (*ET*) is updated. There are five outgoing transitions on location *pre_backoff* but only one fires at a time, according to *BE*. Value b is randomly chosen from one of the five intervals (see TABLE I). The backoff delay is computed by the function *backoff(b)*. After one transition firing, the control moves to the next location. If the remaining time in the CAP is less than *BCD*, *i.e.*, $ET + BCD \geq CAP$, the control goes to location *backoff_paused*. This location models the fact that the backoff countdown will be paused by the end of the CAP, and resumed in next superframe (note that the resumed *BCD* is derived by the expression $BCD + ET - CAP$). Otherwise, the transition leading to location *backing_off* fires. The control should stay on that location for a time equal to *BCD*.

Once the backoff time is elapsed ($x == BCD$), the outgoing transition on location *backing_off* fires, along with updating *ET*, *RT* and *WT* values. The next location is *RT_test*, which serves to test whether there is enough time in the CAP to carry out the entire transaction or not. If *RT* is less than *TTran*, meaning that the transaction cannot take place, the control moves to location *trans_deferred* and stays there until the end of the superframe. Then, it returns to location *Sync* to try to transmit the same frame again. Otherwise, the control goes to location *channel_assess*, depicting the channel assessment. Nodes assess the channel for a period equal to *CCA* (this constraint is modeled by the invariant $x \leq CCA$ on location *channel_assess* and the guard $x == CCA$). By the end of this period, one of the following two cases will apply.

a) Case one: This case applies if the channel is found idle after assessment. The contention window value is decreased by one. If *CW* does not reach zero, the control goes to location *waiting_bp_2*. Nodes should wait for the next backoff period boundary, and then, the control returns to location *channel_assess*. In contrast, when *CW* reaches zero, location *waiting_bp_3* will be enabled, and the control stays there until the next backoff period boundary. After that, the node starts transmitting the frame. This template triggers a *BeginSend[id]!* event, and the control moves to location *transmitting*. If this template receives a *collision?* event from the medium template while the node is transmitting; *i.e.*, a collision has occurred, the control moves to location *Collision*. The node waits for the next backoff period following a delay equal to *FTD*. Thenceforth, the control returns to location *pre_backoff*. If no collision occurs, the control remains on location *transmitting* until the end of the frame transmission. Then, an *EndSend[id]!* event is triggered indicating that the frame is successfully transmitted. The control remains on location *waiting_bp_1* until the next backoff period. Then, it returns to location *pre_backoff*, and the node attempts to transmit the next frame.

b) Case two: This case applies if the channel is found busy after assessment. If *NB* did not reach the value *maxBackoffs* yet, the control goes to location *waiting_bp_4*, while the node waits for the next backoff period. After that, *CW* will be reset, *NB* and *BE* will be increased by one, and the transition to location *pre_backoff* fires. When the channel is found busy, but *NB* is equal to *maxBackoffs*, there will be a transmission error. The control will go to location *error* and will stay there untill the next backoff period boundary. The node resets its backoff parameters, updates the elapsed time value, and increases the number of errors by one. Next, the control returns to location *pre_backoff*, meaning that a new transmission could be attempted.

VI. Conclusion

The primary purpose of this paper was the modeling of the channel access and frame transmission during the contention access period of an IoT network operating over IEEE 802.15.4e standard. We have considered the presence of greedy nodes that cheats with the contention window value, the minimum, and the maximum value of the backoff exponent. We have been able to describe both honest and greedy nodes behavior by the same UPPAAL model. Honest and greedy nodes can be created just by instantiating the template with appropriate values of the parameters.

In our future work, we aim to extend this work with an experimental study through simulation. On the one hand, we aim at studying the impact of greedy nodes on the performance of honest nodes and the performance of the whole network. On the other hand, we expect to model the greedy behavior in the TSCH MAC mode and study its impact on the performance and on the energy efficiency of a 6TiSCH network. Furthermore, we intend to develop a detection method and a punishment mechanism of greedy nodes.

References

[1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec 2016.

[2] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266 – 2279, 2013.

[3] A. Čolaković and M. Hadžialić, "Internet of things (iot): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144, pp. 17 – 39, 2018.

[4] "IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans)," *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, Sep. 2011.

[5] M. Raya, I. Aad, J. . Hubaux, and A. E. Fawal, "Domino: Detecting mac layer greedy behavior in IEEE 802.11 hotspots," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1691–1705, Dec 2006.

[6] A. Hamieh, J. Ben-Othman, A. Gueroui, and F. Nait-Abdesselam, "Detecting greedy behaviors by linear regression in wireless ad hoc networks," in *2009 IEEE International Conference on Communications*, June 2009, pp. 1–6.

[7] Y. Hammal, J. Ben-Othman, L. Mokdad, and A. Abdelli, "Formal modeling of greedy nodes in 802.15.4 wsn," *ICT Express*, vol. 1, no. 1, pp. 10 – 13, 2015.

[8] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer Networks*, vol. 76, pp. 146 – 164, 2015.

[9] M. N. Mejri and J. Ben-Othman, "Detecting greedy behavior by linear regression and watchdog in vehicular ad hoc networks," in *2014 IEEE Global Communications Conference*, Dec 2014, pp. 5032–5037.

[10] ——, "Gdvan: A new greedy behavior attack detection algorithm for vanets," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 759–771, March 2017.

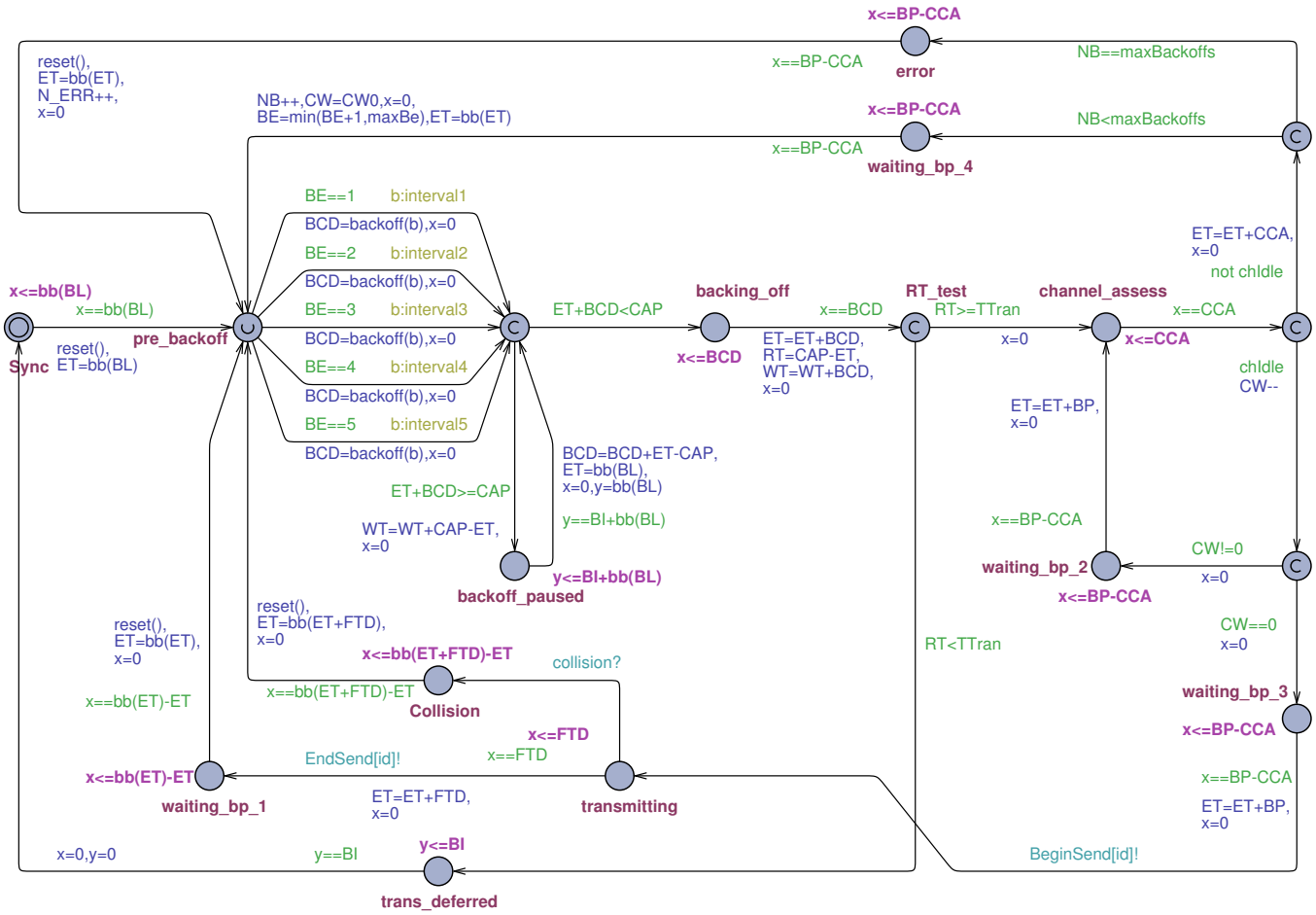

Fig. 5. Node template.

[11] A. Antonopoulos and C. Verikoukis, "Misbehavior detection in the internet of things: A network-coding-aware statistical approach," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, July 2016, pp. 1024–1027.

[12] L. Mokdad, A. Abdelli, and J. Ben-Othman, "Detection of greedy behavior in wsn using IEEE 802.15 protocol," in *2014 IEEE 22nd International Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems*, Sep. 2014, pp. 106–111.

[13] L. Mokdad, Y. Hammal, J. Ben-Othman, and A. Abdelli, "Formal modeling and analysis of greedy behaviors in IEEE 802.11 protocols," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 3579–3584.

[14] X. Liu and T. N. Saadawi, "Markov chain analysis of IEEE 802.11 wireless networks with greedy behavior and heterogeneous traffic," in *2011 11th International Symposium on Communications Information Technologies (ISCIT)*, Oct 2011, pp. 372–377.

[15] A. Abdelli, L. Mokdad, J. B. Othman, and Y. Hammal, "Dealing with a non green behaviour in wsn," *Simulation Modelling Practice and Theory*, vol. 84, pp. 124 – 142, 2018.

[16] J. Kim and K. S. Kim, "Detecting selfish backoff attack in ieee 802.15.4 csma/ca using logistic classification," in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2018, pp. 26–27.

[17] J. Abegunde, H. Xiao, and J. Spring, "A dynamic game with adaptive strategies for IEEE 802.15.4 and iot," in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug 2016, pp. 473–480.

[18] D. Al-Terri, H. Otrok, H. Barada, M. Al-Qutayri, and Y. A. Hammadi, "Cooperative based tit-for-tat strategies to retaliate against greedy behavior in vanets," *Computer Communications*, vol. 104, pp. 108 – 118, 2017.

[19] "IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpans)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pp. 1–320, Sep. 2006.

[20] D. De Guglielmo, G. Anastasi, and A. Seghetti, *From IEEE 802.15.4 to IEEE 802.15.4e: A Step Towards the Internet of Things*. Springer International Publishing, 2014, pp. 135–152.

[21] "IEEE standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[22] M. Mohamadi and M. R. Senouci, "Scheduling algorithms for IEEE 802.15.4 tsch networks: A survey," in *Advances in Computing Systems and Applications*, O. Demigha, B. Djamaa, and A. Amamra, Eds. Cham: Springer International Publishing, 2019, pp. 4–13.

[23] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled tsch," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '15. New York, NY, USA: ACM, 2015, pp. 337–350.

[24] R. T. Hermeto, A. Gallais, and F. Theoleyre, "Scheduling for IEEE802.15.4-tsch and slow channel hopping mac in low power industrial wireless networks: A survey," *Computer Communications*, vol. 114, pp. 84 – 105, 2017.

[25] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183 – 235, 1994.

[26] R. Alur, "Timed automata," in *Computer Aided Verification*, N. Halbwachs and D. Peled, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 8–22.

[27] J. Bengtsson and W. Yi, *Timed Automata: Semantics, Algorithms and Tools*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 87–124.