Mobile Information Systems 9 (2013) 157–188
DOI 10.3233/MIS-130160
IOS Press

Modelling medium access control in IEEE 802.15.4 nonbeacon-enabled networks with probabilistic timed automata

Tatjana Kapus
Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ul. 17, SI-2000 Maribor, Slovenia
Tel.: +386 2 220 7213; Fax: +386 2 220 7272; E-mail: tatjana.kapus@um.si

Abstract. This paper concerns the formal modelling of medium access control in nonbeacon-enabled IEEE 802.15.4 wireless personal area networks with probabilistic timed automata supported by the PRISM probabilistic model checker. In these networks, the devices contend for the medium by executing an unslotted carrier sense multiple access with collision avoidance algorithm. In the literature, a model of a network which consists of two stations sending data to two different destination stations is introduced. We have improved this model and, based on it, we propose two ways of modelling a network with an arbitrary number of sending stations, each having its own destination. We show that the same models are valid representations of a star-shaped network with an arbitrary number of stations which send data to the same destination station. We also propose how to model such a network if some of the sending stations are not within radio range of the others, i.e. if they are hidden. We present some results obtained for these models by probabilistic model checking using PRISM.

Keywords: Wireless personal area networks, medium access control, hidden stations, formal specification, probabilistic model checking

1. Introduction

The IEEE 802.15.4 standard specifies the *Medium Access Control* (MAC) sublayer and the physical layer for low-rate Wireless Personal Area Networks (WPANs) [22,23]. The well-known WPANs that rely on it are the wireless sensor networks (WSNs) following the ZigBee standard, which defines the upper protocol layers for WPANs [1]. The IEEE 802.15.4 standard allows WPANs with star and peer-to-peer operation. In the former, there is a relaying device called a coordinator, and all the other devices can only communicate through it. In the latter, there is also a coordinator, but all the devices can communicate directly if they are within radio range of one another. The MAC sublayer specification is needed because those WPAN devices that are within range of each other share a common channel. Two modes of the MAC operation are defined: beacon- and nonbeacon-enabled. In the former, all communication takes place via the coordinator, which controls any access of the devices to the common channel by sending beacon frames to them. After obtaining a beacon frame, the devices compete for the transmission channel towards the coordinator by executing a slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) algorithm. No beacon frames are sent in the nonbeacon-enabled mode. All the devices within the same range, including the coordinator, compete for the common channel by executing an unslotted CSMA-CA algorithm.

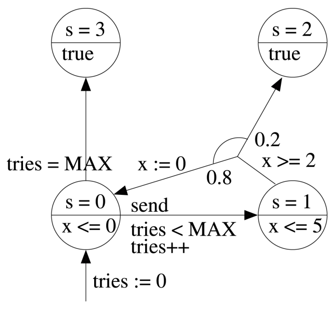
Traditional methods for the performance evaluation of networks and protocols comprise discrete-event simulation with simulators such as OPNET or ns-2 and mathematical analysis using stochastic modelling. These have extensively been used for studying the performance of beacon-enabled (e.g. [13, 18,21]) and nonbeacon-enabled IEEE 802.15.4 networks (e.g. [9,12,26]). Over the last decade, formal methods, probabilistic model checking in particular, have increasingly been investigated as a means of studying the performance and reliability of probabilistic systems [6,11,16]. Probabilistic model checking is an algorithmic method implemented in software called a *model checker*. Basically, it takes a formal description of the system in the form of a finite-state model in which transitions between states can fire with certain probabilities and, in contrast to the simulators, automatically checks the complete state space for determining the validity of a required performance or reliability property expressed in a kind of logic. Depending on the kinds of model and logic, it can for example be verified as to whether an event will happen with a given probability or within a given time-limit, or even what the probability or the time-limit is for that event.

Fruth was the first to use probabilistic model checking for the performance analysis of IEEE 802.15.4 MAC protocols [7,8]. The PRISM model checker was used [24], and both the beacon- and nonbeacon-enabled networks were analysed. In this paper, we are interested only in the nonbeacon-enabled networks. In [7,8], their MAC operation is basically modelled by using probabilistic timed automata. However, as probabilistic timed automata were not supported by PRISM before 2011, the models are in fact written by using Markov decision processes, in which there is no time. The time is represented by additional variables. In this paper, we explain how to write the models directly by using probabilistic timed automata which are supported by new versions of PRISM [14]. In [7,8], only a network consisting of two stations sending to two different destination stations within the same radio range is modelled, and we found that the model is slightly inaccurate. We first created an improved model of such a network. Based on it, we propose two ways of modelling a network with n sending and n receiving stations, for an arbitrary n. We show that the same models can be used to represent a star-shaped network with n stations sending to the same destination. We also show how these models can be adapted for those cases when some sending stations are hidden, i.e. not within radio range of the others. Using the proposed models, we carried out some experiments with the PRISM model checker regarding the probability of successfully sending data and the effect of the duration of the clear channel assessment, which is part of the MAC protocol.

This paper is organised as follows. Section 2 briefly introduces probabilistic timed automata. Section 3 contains a description of the MAC operation in the case where a station wants to send a data frame within a nonbeacon-enabled network. In Section 4, we explain the improved probabilistic timed automata model of the MAC operation for two sending and two receiving stations. In Section 5, we present the models for n pairs of communicating stations. Section 6 explains why the suggested models are also valid representations of the star topologies with one receiving station and how to model a star-shaped network with hidden stations. In Section 7, we present the results of model checking using PRISM. Section 8 concludes the paper with a discussion and suggestions for future work.

2. Probabilistic timed automata

We present the probabilistic timed automata rather informally (cf. [14]) and as supported by PRISM. For formal definitions please see e.g. [15]. A Probabilistic Timed Automaton (PTA) has a finite set of *states*, including an initial one. It is, in fact, a finite-state automaton enriched with non-negative real-valued variables, called *clocks*, and with discrete probabilistic choice. Additionally, it can have finite-range *data variables*. Initially, the values for all the clocks are zero. The values of the clocks increase



```
pta
const int MAX
module  sender
  // local variables
  s: [0..3] init 0;
  tries: [0..MAX] init 0;
  x: clock;
  invariants
    (s = 0 => x <= 0) & (s = 1 => x <= 5)
  endinvariants
  // guarded commands
  [send] s = 0 & tries < MAX  > (s' = 1) & (tries' = tries + 1);
  [] s = 0 & tries = MAX  > (s' = 3);
  [] s = 1 & x >= 2  > 0.2 : (s' = 2) + 0.8 : (s' = 0) & (x' = 0);
endmodule
```

Fig. 1. A graphical representation of a PTA and its corresponding PRISM code.

simultaneously over time. The execution of *transitions* between the states is controlled by two kinds of predicates. *Invariants* are predicates over the clock variables and are associated with the states. The PTA may stay in a state only as long as the invariant associated with it is true. *Guards* are predicates over the clock variables and the data variables, and are associated with the transitions. Only non-negative integer values may be used in invariants and guards. A transition may occur only if its guard is true. A transition can reset some of the clocks (to integer values), update the data variables, and lead the automaton to a new state. The effect of the transition is chosen probabilistically. The possible choices are specified by a discrete probability distribution. If multiple transitions can be taken from a state, one of them is chosen nondeterministically. Instead of a transition occurrence, the values of the clocks may increase implicitly. Formally, this is called a delay transition. Whether a delay transition will occur as well as the amount of the delay are chosen nondeterministically, subject to invariant satisfaction.

In Fig. 1, on the left, there is a graphical representation of a PTA, which represents the sending of a message over an unreliable channel. It has four states. In the lower half of the nodes, the invariants of the states are indicated. *tries* is a data variable, which counts the number of transmission attempts, and x is a clock. New sending attempts are made until *tries* is equal to a constant *MAX*. The state $s = 0$ must be left immediately because its invariant does not allow x to be greater than 0. The transition labelled with *action send* represents the beginning of a new transmission attempt. It increments the counter and leads to state $s = 1$. The invariant associated with this state and the guard of the transition leading from it mean that the transmission can take between 2 and 5 time units. This transition has two possible effects. With a probability of 0.2, the transmission succeeds, otherwise, the transmission fails, the clock is reset to 0, and the PTA returns to $s = 0$.

In PRISM, PTAs and other supported probabilistic models are specified by using a uniform textual language. The PRISM code for the presented PTA is given on the right of Fig. 1. A PTA is specified as a module. It should be noticed that in PRISM, the states are represented by local variables. The initial value of a variable is the lowest value within the declared range if it is not set by using `init`. As usual, primed variables denote the values in the next state. Transitions are described by guarded commands and can be labelled with actions. A system can be specified as a collection of modules. By default, they represent a parallel composition of PTAs with synchronisation on common actions, which means that all the PTAs are executed concurrently and that the transitions of different PTAs with the same label must be executed simultaneously. Each module may read local variables (i.e. use them in its guards) of the others, but can only write to its own. The values of all the clocks for all the modules increase simultaneously over time.

3. Operation of medium access control in nonbeacon-enabled mode

When a station wants to send a data frame in the nonbeacon-enabled mode, it starts to execute unslotted CSMA-CA algorithm as follows, in order to get access to the channel [22]. It first waits for a random number of backoff periods, chosen uniformly between 0 and $2^{BE} - 1$. Initially, BE, the *backoff exponent*, is equal to the value of the standard parameter *macMinBE* (in the sequel denoted *BE_MIN*), the default value of which is 3, but can range from 0 to 3. The *backoff period*, in the sequel denoted *BO_PERIOD*, is equal to the time needed to transmit 20 symbols with a chosen standard bit rate. After the waiting time, the station performs the *clear channel assessment* (CCA) for the duration of 8 symbols (denoted as *CCA* in the sequel).

If the channel is found to be busy at any time during the CCA period, it is indicated as busy by the CCA at the end of this period, and otherwise as free. If it is indicated as busy, the station increments BE by one if it is smaller than *macMaxBE* (denoted by *BE_MAX*; its default value being 5) and leaves it unchanged otherwise, and again randomly backs off and carries out the CCA. This backoff procedure is repeated until either the CCA succeeds, i.e. indicates that the channel is free, or the number of backoffs reaches *macMaxCSMABackoffs* (denoted by *NB_MAX*; it can range from 0 to 5, its default value being 4). If the backoff procedure is executed *NB_MAX* times without success, the station ends the execution of CSMA-CA by declaring a *channel access failure*.

If the CCA succeeds, the CSMA-CA is taken to be finished successfully, the station switches from the *receive* mode to the *transmit* mode (*RX-to-TX*) and transmits the frame. As usual in the literature, we shall assume that any RX-to-TX or TX-to-RX turnaround lasts exactly 12 symbols, which is the maximal possible value allowed by the standard (*aTurnaroundTime*, denoted as *TURNAROUND* in this paper). Optionally, the sending station can require an acknowledgement within the data frame. If the destination station receives such a frame, it performs a RX-to-TX turnaround and immediately after this transmits the acknowledgement. After the data frame transmission, the sending station performs a TX-to-RX turnaround and waits for *macAckWaitDuration* symbols, denoted *ACK_TIMEOUT* in the sequel and equal to the sum of the acknowledgement length in symbols, *TURNAROUND* and *BO_PERIOD*. If the acknowledgement does not come within this time, the sending station sets BE to *BE_MIN* and repeats the CSMA-CA algorithm as if sending the data frame for the first time. If the acknowledgement does not come even after repeating the algorithm *aMaxFrameRetries* times (denoted as *MAX_RETRIES* in the sequel; 3 by default, but can range from 0 to 7), the sending station declares a *collision failure* and stops trying. If the acknowledgement is received, the sending station declares *success*. If the sending station does not use the acknowledgement option, it declares *success* immediately after the data frame transmission.

After the transmission of an acknowledgement, the destination station performs a TX-to-RX turnaround. If a data frame starts to arrive at the destination during this time, the destination ignores it [26].

The length of the acknowledgement frames is 11 octets. The data frames can be from 15 to 133 octets in length. The rest of this paper supposes that one octet corresponds either to 8 symbols (this is the case if the standard bit rate of 20 or 40 kbps is used for transmission) or 2 symbols (this is the case if the bit rate is 250 kbps).

Unless otherwise stated, in the sequel we consider formal models for the MAC using acknowledgements. The models for the MAC without acknowledgements can easily be obtained from them.

4. Improved formal model for two pairs of stations

In this section, we introduce a PTA model of an IEEE 802.15.4 network consisting of sending stations s_i, $i = 1{,}2$, and receiving stations r_i, $i = 1, 2$, all being within radio range of each other. As in [7,8], we suppose that the air is an ideal medium except for the possibility of collisions, that station s_i sends one message to station r_i by using the unslotted CSMA-CA and the acknowledgements as described in Section 3, and that both sending stations have a message to send at the same time. Throughout this paper, we assume that whenever two stations actually transmit a frame at the same time, a collision occurs, and that propagation delays are negligible.

In [7,8], the model of this network is a parallel composition of a PTA representing the common communication channel between the stations and of two PTAs representing the pair s_1, r_1, and, respectively the pair s_2, r_2. In fact, the PTAs are given only in graphical form. In the PRISM language, they are represented as Markov decision processes. Our aim is to write the PTA model of the network in PRISM. The PTA model given in [7] (and repeated in [8]) contains come features unsupported by the PRISM language. It contains urgent states and transitions. An *urgent state* is one in which time must not advance and must, therefore, be left immediately. It can easily be represented in PRISM by introducing a clock variable which is set to 0 on all its ingoing transitions and by requiring in its invariant that the clock be less or equal to 0. For example, state $s = 0$ in Fig. 1 is urgent. An *urgent transition* forbids the execution of delay transitions in its starting state if its guard is true, i.e. the transition must be taken as soon as it is enabled [15]. Suppose, for instance, that in Fig. 1 we would want the transition from state $s = 1$ to be urgently executed at $x = 2$. Then, it would be insufficient to only write $x = 2$ instead of $x \geqslant 2$ in its guard. The transition should additionally be labelled as urgent, but PRISM does not support this. A solution would be to impose the invariant $x \leqslant 2$ on the starting state of this transition.

Since PRISM neither supports the description of systems in graphical form nor the simulation of PTAs, we used UPPAAL as an auxiliary tool for the preparation of PRISM PTA descriptions. UPPAAL is a tool for the modelling, simulation, and verification of timed automata [2]. These are basically like PTAs, except that they do not support probabilistic choice in transitions. Since in our case, the only effects with probabilities less than 1 are those for choosing the *backoff* values, we can model them in UPPAAL by nondeterministic choice (cf. [17]). Figure 2 shows the timed automaton, drawn in UPPAAL, which corresponds to our PTA model of a pair of stations s_i, r_i, written in PRISM. It is very similar to the PTA drawn in [7,8], except that inaccuracies are eliminated and that urgent states and transitions are represented as explained above. UPPAAL, unlike PRISM, allows writing of state names. The latter and the invariants are written in bold at the nodes. Although UPPAAL contains a special label to indicate a state as urgent, we modelled urgent states with help of invariants. UPPAAL descriptions also contain declarations, but they are not shown in this paper. For instance, the constants ($DATlen$, *BO_PERIOD*, ...) and the variables (*be*, *backoff*, ...) from Fig. 2 are declared there. Variable *be* is initialised to *BE_MIN*. UPPAAL uses the C-style notation for equality and assignment.

The transitions in Fig. 2 between the initial state *SET_BACKOFF* and *BACKOFForNOT* represent the setting of variable *backoff* to a random number as prescribed by the CSMA-CA algorithm. The states containing letter C are so-called committed states, which are known in UPPAAL, but not in PTAs in PRISM. For example, if be is equal to 2 in state *SET_BACKOFF*, then the automaton passes to the committed state in which the outgoing transition nondeterministically chooses an integer i between including 0 and 3, and assigns it to *backoff*. The committed state guarantees that this transition is executed immediately after the transition to this state and with no other transition (of another automaton in the parallel composition) inbetween. It should be noticed that some of the transitions in the automaton are

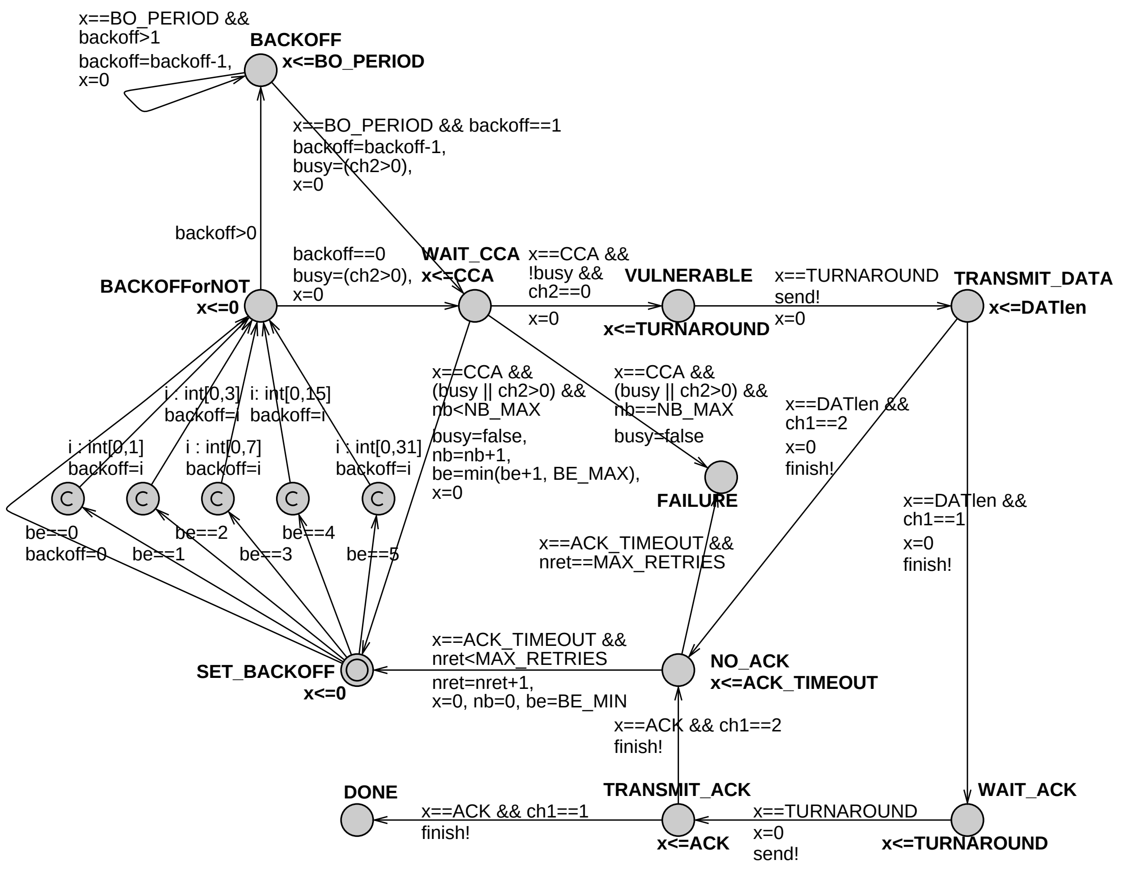


Fig. 2. Timed automaton model of a sending/receiving station pair in case of two sending stations.

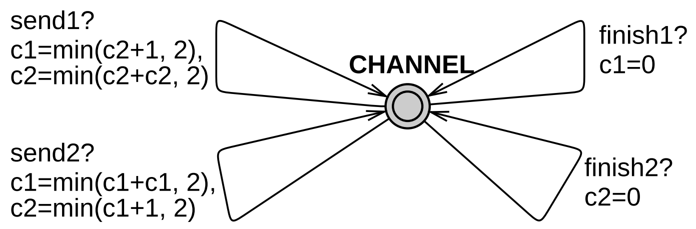


Fig. 3. Model of medium for two sending stations.

labelled with $send!$ and, respectively, $finish!$. These labels have a role similar to the $send$ mentioned in Section 2, except that in UPPAAL, every label ends with ! ('output') or ? ('input'). In contrast to PRISM, in a parallel composition of timed automata in UPPAAL, there is binary synchronisation of transitions with complementary labels: a transition with a label ending with ! is executed simultaneously with a transition labelled with the same label, but ending with ?.

The automaton shown is a so-called template. Let us call it *Station*. $ch1$, $ch2$, $send$, and $finish$ are its parameters. The model of the pair of stations s_1, r_1 (respectively, s_2, r_2) is obtained by setting $ch1$ to $c1$ (respectively, $c2$), $ch2$ to $c2$ ($c1$), $send$ to $send1$ ($send2$), and $finish$ to $finish1$ ($finish2$). The network model is the parallel composition of these two models, let us call them *Station1* and *Station2*, and of the timed automaton representing the communication channel. The latter is shown in Fig. 3. Let us call it *Medium*. Except for the uses of ! and ? indications, it is the same as the graphical representation of the PTA in [7], with the exception that in the latter its only state should not be indicated as urgent. $c1$ and $c2$ are (in UPPAAL global and in PRISM local) variables initialised to 0.

The idea is that the values of $c1$ and $c2$ can be either 0, 1, or 2. $c1 = 0$ (and analogous for $c2$) indicates that nothing is being sent by *Station1* (i.e. between s_1 and r_1). $c1 = 1$ (note that in the text, we use the PRISM notation for equality and similar) means that a (data or acknowledgement) frame is being sent by *Station1*, but not by *Station2*, and that the frame has not been garbled by a collision. $c1 = 2$ means that a frame is being sent by *Station1* and that it has been garbled by a collision with a frame sent by *Station2*.

The transitions labelled with $send1!$ and, respectively, $finish1!$ starting in states *VULNERABLE* and, respectively, *TRANSMIT_DATA* represent the start and, respectively, the end of sending the data frame by s_1 as described in Section 3, whereas the similarly labelled transitions from state *WAIT_ACK* and, respectively, *TRANSMIT_ACK* represent the start and, respectively, the end of sending the acknowledgement frame by r_1 (and at the same time receiving it by s_1). The analogous holds for sending and receiving in *Station2*. Every such transition is executed simultaneously with the transition of *Medium* with the complementary label, which properly sets variables $c1$ and $c2$.

The representation of the pair s_1, r_1 (and analogously for $i = 2$) with one automaton can be justified as follows. Once the last bit of a data frame sent by s_1 arrives (successfully, i.e. with no collision during the frame sending) at r_1, it shall inevitably send an acknowledgement after switching from RX to TX within a *TURNAROUND* interval because it is impossible that it ignores this frame. As told in Section 3, it would ignore it if it was in the course of turning from the TX mode to RX after sending an acknowledgement. It is, however, impossible for a new data frame to arrive during that time because for every data frame sent, the sending station waits longer than the time needed to completely send the acknowledgment, and after this waiting does not send the same data frame or a new one for at least *CCA* + *TURNAROUND* time units. Please note, that in this paper we model only the sending of one fresh data frame per station but nevertheless take into account the possibility of sending more frames when reasoning about the validity of the models. Consequently, all the models for sending one data frame in this paper can be readily used to build models for sending more than one data frame by adding appropriate transitions leading from the success/failure states back to the beginning of the CSMA-CA algorithm. Different delays before the latter could also easily be introduced in different stations.

It suffices to represent the start (respectively, the end) of sending of a data frame from s_i to r_i and the start (respectively, the end) of sending of the acknowledgement in the other direction with the same labels in the (P)TAs representing the stations and with the same transitions in the (P)TA representing the medium for the following reason. s_i waits a sufficient time before (re)sending a data frame so that the acknowledgement for the previous one from this station does not occupy the medium anymore, or shortly, because it is impossible for a data frame from s_i and an acknowledgement from r_i to be transmitted simultaneously.

There are four inaccuracies in the PTA model of the pair of stations given graphically in [7,8]. The model in Fig. 2 does not contain them anymore. Three of them are also no longer present in the PRISM code for Markov decision processes in [8]. One of the latter is that the length of the data frame to be sent is chosen anew each time the sending is retried because of the absence of an acknowledgement. This is not in accordance with the standard. We use a constant value $DATlen$. The length of the fresh data frame in the model could, of course, be chosen nondeterministically and remembered for retransmission as in the code in [8]. Please note, that for convenience throughout this paper it is assumed that all the stations send data frames of equal length.

Another inaccuracy in [7] is the following. The length of each data frame is chosen nondeterministically between *DATA_MIN* and *DATA_MAX* units, and assigned to a variable $data$. The state which is analogous to state *TRANSMIT_DATA* in Fig. 2 has the invariant $x \leqslant data$, which is analogous to the invariant in Fig. 2. On the transitions from this state, in the model for the station pair i, ci ($ch1$ in Fig. 2) is checked to see if the data frame sent has collided ($ci = 2$) or not ($ci = 1$). However, the transition which leads to state *NO_ACK*, meaning that a collision has occurred, has the time guard $x \geqslant$ *DATA_MIN* in [7]. This, together with the invariant, means that the station can stop sending the data frame as soon as the *DATA_MIN* units are sent, even if the frame is longer than that. This is not in accordance with the standard. In the CSMA-CA algorithm, the station sends the complete frame even if a collision has occurred, because it detects collisions from the absence of acknowledgements. In [7], the time guard should be $x = data$ (or equivalently, $x \geqslant data$, as in the code in [8]). In our model, the analogous guard $x = DATlen$ is applied.

One inaccuracy in [7] is intentional: the data frame is resent until an acknowledgement arrives instead of only *MAX_RETRIES* times as in the standard and optionally in the PRISM code in [8].

The following inaccuracy, however, is unsolved in [8]. In the PTA graphs in [7,8], testing whether the channel is clear or not is carried out only at the end of the CCA period. As mentioned in Section 3, it should be assessed from the beginning to the end of the CCA period and declared as clear at the end of the period iff it was clear during the whole period. This inaccuracy has already been detected and corrected in the model of nonbeacon-enabled MAC written in ns-2. In [25], it is written that in that model the channel is tested at the end of the first symbol of the CCA period, but its status is reported after the last symbol. It is unclear whether the channel is also tested at the end of the last symbol as in the previous version. According to [5], in the OMNeT++ model the channel is tested at the beginning and at the end of the CCA period and reported busy iff it is busy at least at one of these moments. We came to the same solution independently for the PTA model. In Fig. 2, the channel is tested on both transitions to state *WAIT_CCA*, as well as in this state when *CCA* time units pass. This is equivalent to testing for the whole CCA period because the minimal lengths of the data and acknowledgement frames are greater than the CCA period duration. Consequently, if a CCA period overlaps with a data or acknowledgement frame sending period, at least the start or the end of the CCA period lies within the latter.

It is easy to obtain the description of PTAs in PRISM from the timed automata in UPPAAL by following the pattern shown in Fig. 1. In PRISM, there are no templates, but a given module can be used to define another one which differs from it only in the module name and the names of some variables, constants, and actions by calling the first one and specifying the renaming of the original names. Another difference from UPPAAL is in the fact that those variables and constants meant to be local to a module must be named differently than variables and constants in other modules. Naturally, the PRISM module representing *Station1* gets all of them, as well as actions $send$ and $finish$ indexed with 1. The module representing *Station2* can be obtained by replacing them with the names indexed with 2:

```
module station2=station1[x1=x2, s1=s2, c1=c2, c2=c1, backoff1=backoff2,
                         be1=be2, nb1=nb2, nret1=nret2, busy1=busy2,
                         send1=send2, finish1=finish2]
endmodule
```

There, $s1$ and $s2$ are the variables used to represent the states of *Station1* and, respectively, *Station2* by integer values in a similar way as the states are represented in Fig. 1.

The essential difference between *Station*s in UPPAAL and the station modules in PRISM is, of course, that the latter represent *probabilistic* timed automata. Therefore, the nondeterministic choice of the *backoff* value in the initial state is modelled by specifying a uniform probabilistic distribution. Suppose that $s1 = 2$ denotes state *SET_BACKOFF* and $s1 = 3$ state *BACKOFForNOT* from Fig. 2 for *Station1*. The setting of *backoff1* in the PTA in PRISM is specified by requiring equal probabilities for all the possible values of *backoff1* for a given *be1* as follows (notice that we do not show all the updates for space reasons) [8,24]:

```
[] s1=2 & be1 = 0 -> (s1'=3) & (backoff1' = 0);
[] s1=2 & be1 = 1 ->
      1/2 : (s1'=3) & (backoff1' = 0) + 1/2 : (s1'=3) & (backoff1' = 1);
[] s1=2 & be1 = 2 ->
      1/4 : (s1'=3) & (backoff1' = 0) + 1/4 : (s1'=3) & (backoff1' = 1)
    + 1/4 : (s1'=3) & (backoff1' = 2) + 1/4 : (s1'=3) & (backoff1' = 3);
[] s1=2 & be1 = 3 ->
      1/8 : (s1'=3) & (backoff1' = 0) + ... + 1/8 : (s1'=3) & (backoff1' = 7);
[] s1=2 & be1 = 4 ->
      1/16 : (s1'=3) & (backoff1' = 0) + ... + 1/16 : (s1'=3) & (backoff1' = 15);
[] s1=2 & be1 = 5 ->
      1/32 : (s1'=3) & (backoff1' = 0) + ... + 1/32 : (s1'=3) & (backoff1' = 31);
```

Here we also give the PRISM guarded commands representing the transitions of PTA *Station1* (cf. Fig. 2) from states *BACKOFForNOT* and *BACKOFF* ($s1 = 4$) that start the CCA period, as well as the transitions from *WAIT_CCA* ($s1 = 5$) because these (and the analogous ones for any other *Station*) will mainly be the ones changed in the sequel:

```
// start of CCA
[] s1=3 & backoff1=0 -> (s1'=5) & (busy1'=(c2>0)) & (x1'=0);
[] s1=4 & x1=BO_PERIOD & backoff1=1 ->
    (s1'=5) & (backoff1'=backoff1-1) & (busy1'=(c2>0)) & (x1'=0);

// end of CCA
[] s1=5 & x1=CCA & (busy1 | c2>0) & nb1<NB_MAX ->
    (s1'=2) & (busy1'=false) & (nb1'=nb1+1) & (be1'=min(be1+1,BE_MAX)) & (x1'=0);
[] s1=5 & x1=CCA & (busy1 | c2>0) & nb1=NB_MAX -> (s1'=12) & (busy1'=false);
[] s1=5 & x1=CCA & !busy1 & c2=0 -> (s1'=6) & (x1'=0);
```

Since the PTA representing the medium has only one state (one node in UPPAAL), no variable is needed to represent it within the PRISM language. Its transitions can either be specified by using the expressions from Fig. 3 (with guards set to true) or as in Fig. 4 [8,24]. It can easily be seen that the effect of the expressions used in Figs 3 and 4 is equivalent if the medium is used in the parallel composition with the stations.

5. Models for two or more station pairs

In this section, we first generalise the model for two pairs of stations s_i, r_i with the common channel to the case of n pairs, $n \geqslant 2$. The model for two pairs, in particular the model of the medium in Fig. 4, suggests that the PTA model of the network is the parallel composition of n PTAs similar to *Station* in Fig. 2, and of a PTA representing the medium with variable *ci* for messages associated with the pair s_i, r_i, $i = 1, \ldots, n$. The idea is that each *ci* still ranges over 0, 1, and 2, and that the meaning of these values is analogous to what it was before. It follows that for each i, the transition labelled with $sendi$

```
module medium

  // medium status
  c1 : [0..2];
  c2 : [0..2];
  // ci corresponds to messages associated with station i
  // 0 nothing being sent
  // 1 being sent correctly
  // 2 being sent garbled

  // begin sending message and nothing else currently being sent
  [send1] c1=0 & c2=0 -> (c1'=1);
  [send2] c2=0 & c1=0 -> (c2'=1);
  // begin sending message and something is already being sent
  // in this case both messages become garbled
  [send1] c1=0 & c2>0 -> (c1'=2) & (c2'=2);
  [send2] c2=0 & c1>0 -> (c1'=2) & (c2'=2);
  // finish sending message
  [finish1] c1>0 -> (c1'=0);
  [finish2] c2>0 -> (c2'=0);

endmodule
```

Fig. 4. A possible description of medium for two sending stations in PRISM.

should set ci to 1 and all the other c variables should remain equal to 0 if no station is sending. If at least one message is being sent (garbled or not), this transition should leave the c variables of the stations that are not sending equal to 0 and set all the others to 2. For each i, the transition labelled with $finishi$ should set ci to 0.

Figure 5 shows the first proposal for the PTA model of the medium for $n = 3$. The template for a station pair differs from the one in Fig. 2 in that in the states *BACKOFFForNOT*, *BACKOFF*, and *WAIT_CCA*, *ch3* has to be checked besides *ch2* to see if the medium is busy (i.e. $ch2 > 0 \mid\ ch3 > 0$) or not (i.e. $ch2 = 0\ \ \&\ \ ch = 0$). Clearly, for an arbitrary n, all the ch variables except $ch1$ have to be checked in the model. Speaking in terms of PRISM, in module *Station1* the commands for these states for an arbitrary n become as follows:

```
// start of CCA
[] s1=3 & backoff1=0 -> (s1'=5) & (busy1'=(c2>0 | ... | cn>0)) & (x1'=0);
[] s1=4 & x1=BO_PERIOD & backoff1=1 ->
    (s1'=5) & (backoff1'=backoff1-1) & (busy1'=(c2>0 | ... | cn>0)) & (x1'=0);

// end of CCA
[] s1=5 & x1=CCA & (busy1 | c2>0| ... | cn>0) & nb1<NB_MAX ->
    (s1'=2) & (busy1'=false) & (nb1'=nb1+1) & (be1'=min(be1+1,BE_MAX)) & (x1'=0);
[] s1=5 & x1=CCA & (busy1 | c2>0| ... | cn>0) & nb1=NB_MAX ->
    (s1'=12) & (busy1'=false);
[] s1=5 & x1=CCA & !busy1 & c2=0 & ... & cn=0 -> (s1'=6) & (x1'=0);
```

Given the new module *Station1* in PRISM, the module representing the station pair i, $1 < i \leqslant n$, can be obtained by renaming as before, except that now, assuming that we list $c_1, \ldots, c_n$ as the parameters of *Station*$_1$ in this order (from now on we will write the indices subscripted for convenience), c_1 must be renamed to c_i, whereas the other c variables of *Station*$_1$ must be renamed disjointly to those c variables other than c_i. Generally, the ordering does not matter, but could if the kind of symmetry present in this protocol could be exploited by the PRISM model checker. We suggest one of the fol-

```
module medium

  // medium status
  c1 : [0..2];
  c2 : [0..2];
  c3 : [0..2];
  // ci corresponds to messages associated with station i
  // 0 nothing being sent
  // 1 being sent correctly
  // 2 being sent garbled

  // begin sending message and nothing else currently being sent
  [send1] c1=0 & c2=0 & c3=0 -> (c1′=1);
  [send2] c2=0 & c1=0 & c3=0 -> (c2′=1);
  [send3] c3=0 & c1=0 & c2=0 -> (c3′=1);
  // begin sending message and something is already being sent
  // in this case all messages sent become garbled
  [send1] c1=0 & c2>0 & c3>0 -> (c1′=2) & (c2′=2) & (c3′=2);
  [send1] c1=0 & c2>0 & c3=0 -> (c1′=2) & (c2′=2);
  [send1] c1=0 & c2=0 & c3>0 -> (c1′=2) & (c3′=2);
  [send2] c2=0 & c1>0 & c3>0 -> (c2′=2) & (c1′=2) & (c3′=2);
  [send2] c2=0 & c1>0 & c3=0 -> (c2′=2) & (c1′=2);
  [send2] c2=0 & c1=0 & c3>0 -> (c2′=2) & (c3′=2);
  [send3] c3=0 & c1>0 & c2>0 -> (c3′=2) & (c1′=2) & (c2′=2);
  [send3] c3=0 & c1>0 & c2=0 -> (c3′=2) & (c1′=2);
  [send3] c3=0 & c1=0 & c2>0 -> (c3′=2) & (c2′=2);
  // finish sending message
  [finish1] c1>0 -> (c1′=0);
  [finish2] c2>0 -> (c2′=0);
  [finish3] c3>0 -> (c3′=0);

endmodule
```

Fig. 5. A possible description of medium for three sending stations in PRISM.

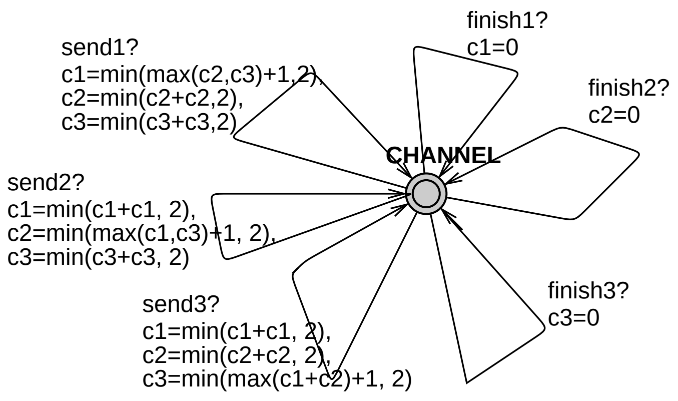


Fig. 6. Simpler description of medium for three sending stations.

lowing (the renaming of c_1 included): *Station*$_i$ = *Station*$_1(c_i, c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n)$ or *Station*$_i$ = *Station*$_1(c_i, c_{i+1}, \ldots, c_n, c_1, \ldots, c_{i-1})$.

We could, of course, derive the description of the medium for an arbitrary n from the one in Fig. 5. We do, however, leave this to the reader and strive for a short description similar to the one in Fig. 3.

From Fig. 5, it can be seen that to properly set c_i in action $send_i$, the largest from all the other c variables has to be taken and incremented by 1, but the result must not be larger than 2. The PTA in Fig. 6 is an equivalent of the PTA in Fig. 5 if used together with the PTAs representing the stations adapted as just explained. For the network with n sending stations, all the commands of the medium can have guards equal to true and the following effect for $i = 1, \dots, n$:

- the effect of *send*$_i$:
  - $c'_i = \min(\max(c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_n) + 1, 2)$,
  - $c'_j = \min(c_j + c_j, 2)$, for $j = 1, \dots, n$, $j \neq i$;
- the effect of $finish_i$: $c'_i = 0$.

Now, we propose a simpler model for the network consisting of n station pairs. In [10], a beacon-enabled network without acknowledgements is modelled by using timed stochastic automata. An integer variable $sending$ initialised to 0 is used. Every station increments it by 1 when it starts to send a data frame and decrements it by 1 when it finishes. This variable suffices for the purpose of CCA – the channel is busy iff it is greater than 0. It, however, does not suffice for the detection in all situations as to whether a frame has collided. The detection of collisions is needed because we have to know whether the acknowledgment will be received successfully or not. In the current models, this is checked in the outgoing transitions of states *TRANSMIT_DATA* and *TRANSMIT_ACK* by checking c_i for *Station*$_i$ (see Fig. 2). Clearly, if $sending$ is greater than 1, it indicates a collision, but it cannot be used instead of c_i because if $sending$ is equal to 1, there are two possibilities. One is that exactly one station is sending and that the frame is not garbled because it started to send when the channel was clear and no other station has started to send thereafter. Another one is that only one station is sending, but the frame is garbled. This could happen as follows. Suppose that only two stations are sending, meaning that the frames being sent have collided. When one of them finishes sending, the variable $sending$ is decremented to 1.

We therefore introduce boolean variable $colind$, which is set by the model of the medium in transitions labelled with $send_i$ to indicate whether the sending of a new (data or acknowledgement) frame by *Station*$_i$ causes a collision (in this case, it is set to true). A similar idea is used in [3]. This suffices in order for the model to properly represent the protocol. However, it is our wish that $colind$ be true exactly when there are garbled messages in the channel. For this reason, we also set it to true in every transition labelled with $finish_i$ if at the time of its execution more than one station is sending, i.e. if $sending > 1$, and to false otherwise. Variable $colind$ is checked instead of c_i in the outgoing transitions of states *TRANSMIT_DATA* and *TRANSMIT_ACK*. For space reasons, we only show the template for *Station*s (Fig. 7) and the PTA representing the medium for the network with three station pairs (Fig. 8) drawn in UPPAAL. The modules in PRISM can be obtained easily. In principle, the transitions of the medium can be the same as in UPPAAL. However, at least the version of PRISM we have used requires a limiting of the possible values of variables in them. We did it by adding the conditions on the values of $sending$ in their guards ($sending > 0$ in the transitions labelled with $finish_i$ and $sending < n$ in the transitions labelled with $send_i$). The advantage of this model is that the common channel for any n is represented by only two variables instead of n variables. Consequently, the model of the medium for n has transitions labelled with $send_i$, $i = 1, \dots, n$, all with the same effect, and likewise for $finish_i$. All the n modules representing the station pairs are instances following the template in Fig. 7. They all check the variables $sending$ and $colind$ of the medium in the same way, and differ only in the indices of the other variables and the action labels.

In fact, UPPAAL allows even a simpler model. The same label, for example *send*! (respectively, *send*?) can be used in the automata representing the stations (respectively, the medium) instead of *send*$_i$! (respectively, *send*$_i$?) for $i = 1, \dots, n$, and likewise for $finish_i$. Consequently, the medium in UPPAAL

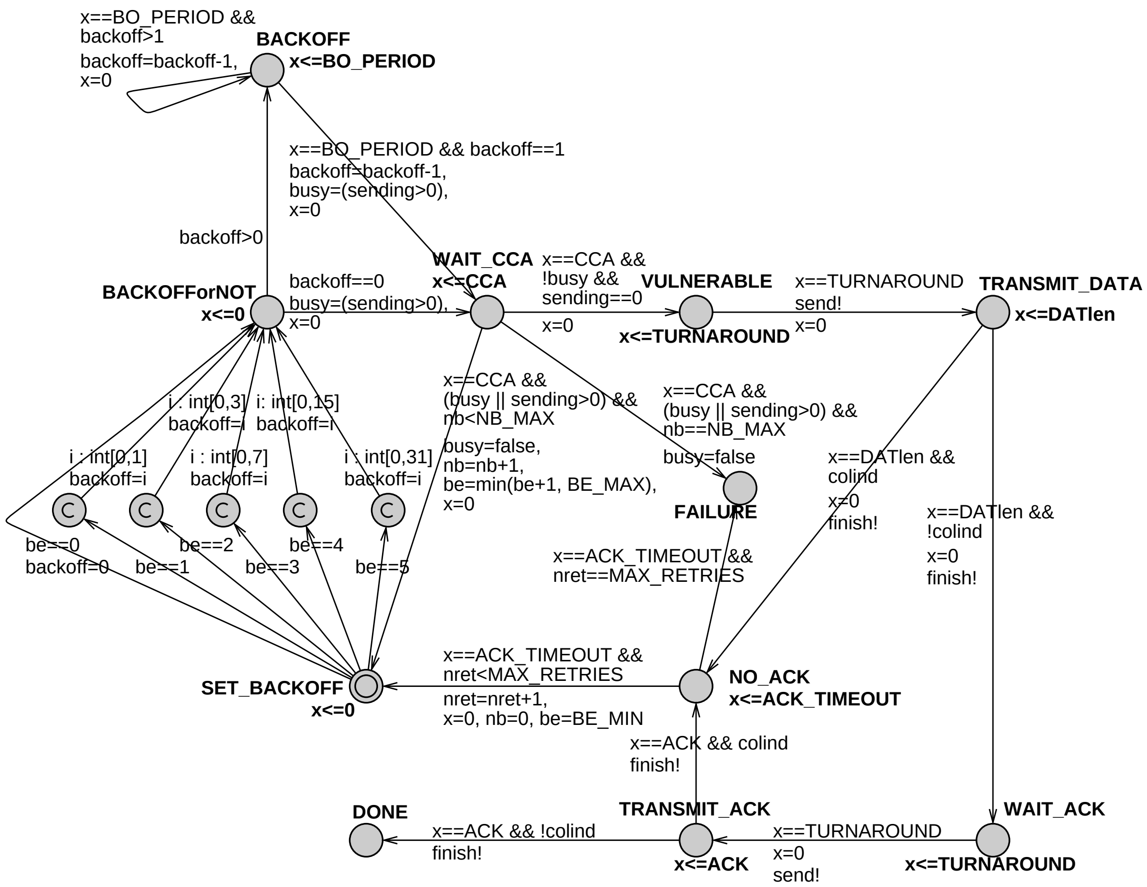


Fig. 7. Model of a station pair using variable $sending$.

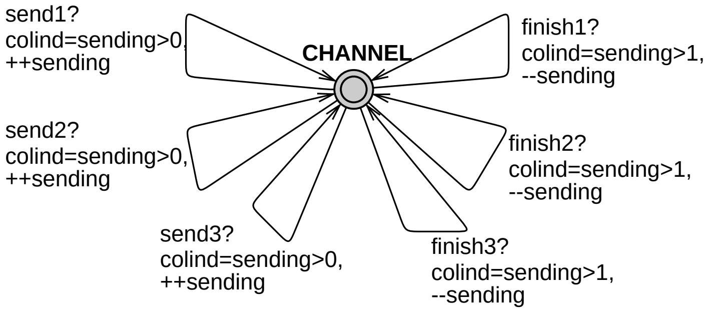


Fig. 8. Medium for three sending stations using variable $sending$.

can be represented with only two transitions. This simplification is possible because of the binary synchronisation in the parallel composition in UPPAAL. In PRISM, by default, all the transitions with the same label must synchronise. It is possible to achieve other kinds of synchronisations with special operators, but we could not find a solution that would allow only labels *send* and $finish$ in the model in PRISM.

6. Models for star topology

6.1. Models for star topology networks with no hidden stations

The IEEE 802.15.4 standard is often used in star-shaped WSNs in which there is a central node (a coordinator called 'receiving station' and denoted by r in the sequel) collecting sensor data sent directly to it by nodes ('sending stations', denoted as s_i) located around it [4]. Suppose that all the stations are within radio range of each other and that the sending stations simultaneously start to execute the nonbeacon-enabled version of the IEEE 802.15.4 MAC protocol with acknowledgements in order to send a data frame (all the stations the frames of equal length) to the receiving station, which only sends acknowledgements back. This could, for example, be the case in a WSN if the coordinator sent a request for sensor data to all the stations [4].

We claim that the models proposed in the previous sections are also the models of such networks. In all of them, automaton $\mathit{Station}_i$ represents the sending of a data frame to r by s_i and of the acknowledgement from r to s_i. The argument for the validity of modelling the sending of the data frame and the acknowledgement with the same transitions in the *Medium* automaton is the same as in Section 4. The question remains whether it is true that once s_i successfully (i.e. without a collision) sends a data frame to r, the latter necessarily sends an acknowledgement to it as in the model. In order for the answer to be positive, we must eliminate the following three possibilities:

- Is it possible for r to receive a data frame from s_i when it is in the course of switching from TX to RX after sending an acknowledgement for the previous data frame from s_i? (As already mentioned, we also want our modelling approach to be valid if several data frames are sent by s_i.)
  This possibility has been eliminated in Section 4.
- Is it possible for r to receive a data frame from s_i when it is in the course of switching from TX to RX after sending an acknowledgement for a data frame received from some s_j, $j \neq i$?
  The answer is no because even the minimal length of the data frames is larger than the *TURNAROUND* interval. Consequently, it is impossible for the last bit of the data frame from s_i to be received by r inside the *TURNAROUND* interval without part of that frame overlapping and thus colliding with the acknowledgement sent by r until the start of that interval.
- Is it possible for a data frame from s_i to come complete and not garbled to r when the latter has just received another data frame and is in the course of switching from RX to TX in order to send an acknowledgement for it?
  It is impossible for a data frame from s_i to come immediately after another data frame from the same station because s_i does not send a new data frame before the acknowledgement for the previous one is finished. It is also impossible for a data frame from s_i to come immediately after a data frame from some station s_j, $j \neq i$, because the minimal length of the data frames is larger then the *TURNAROUND* interval, and consequently, the data frame from s_i ending in that interval would necessarily collide with the data frame from s_j.

6.2. Models for star topology networks with hidden stations

In contrast to Section 6.1, we now suppose that not all the sending stations in a star-shaped network are within the range of each other and propose accordant adaptions of the models. Suppose a network as sketched in Fig. 9 consisting of the receiving station r, which receives data frames from and sends acknowledgements to the sending stations s_i, $i = 1, \dots, n$. They are all within the range of r and

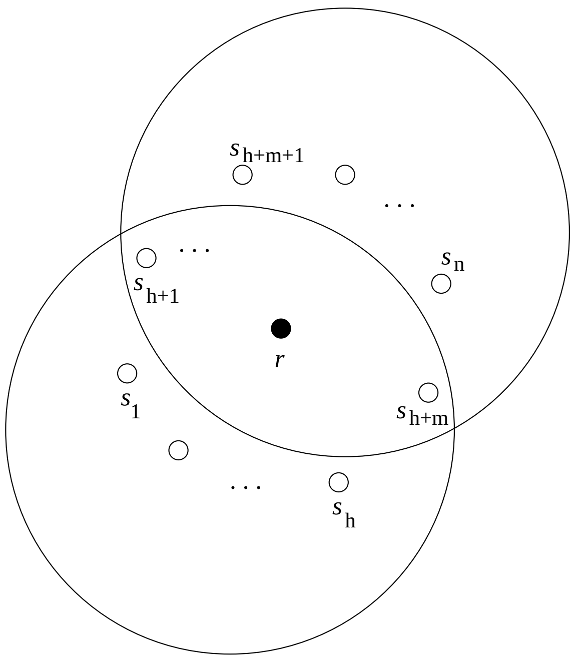


Fig. 9. A star-shaped network with hidden stations.

vice versa. There are three different kinds of sending stations. Stations s_i with the lowest indices, $i = 1, \ldots, h$, are within the range of each other, but outside the range of stations s_i with the highest indices, $i = h+m+1, \ldots, n$. Likewise, the latter are within the range of each other, but outside the range of the former. Stations s_i, $i = h+1, \ldots, h+m$, are in a common range with all the others, like r. It follows that the stations with the lowest indices are *hidden* from the stations with the highest ones and vice versa.

The problem with the hidden stations with the lowest indices is in that when they perform CCA, they do not "hear" whether anything is being sent by the stations with the highest ones, and vice versa. A hidden station, therefore, might send a data frame after the *TURNAROUND* time from the end of CCA even if a hidden station from the other range is sending. This might increase the probability of collisions because once a data frame from a hidden station is sent, it propagates into the other range. As already mentioned, we neglect the propagation delay, which means that such a data frame comes into that range immediately after the start of sending.

We first propose a way of modelling the star-shaped network with the hidden stations by adapting the model for n stations presented in Section 5 in which the medium is represented by variables c_i, $i = 1, \ldots, n$.

Now, the model in PRISM is a collection of a PTA (i.e. module) representing the medium as well as of two kinds of PTAs, the ones representing the hidden stations and the ones representing the sending stations that are within the range of all the others.

The PTA in PRISM representing a hidden station s_i is the same as the PTA representing station s_i in the network with no hidden stations (and similar to the UPPAAL template shown in Fig. 2), except for the following. In the former model, in states *BACKOFForNOT*, *BACKOFF* and *WAIT_CCA*, the variables c_j of all the other *Station*s are tested. However, c_j represents the sending of either a data frame by station s_j or an acknowledgement by r. Now, s_i is only reached by acknowledgements sent by r and by data frames sent by the stations that are within the same range as s_i. We, therefore, introduce a new variable

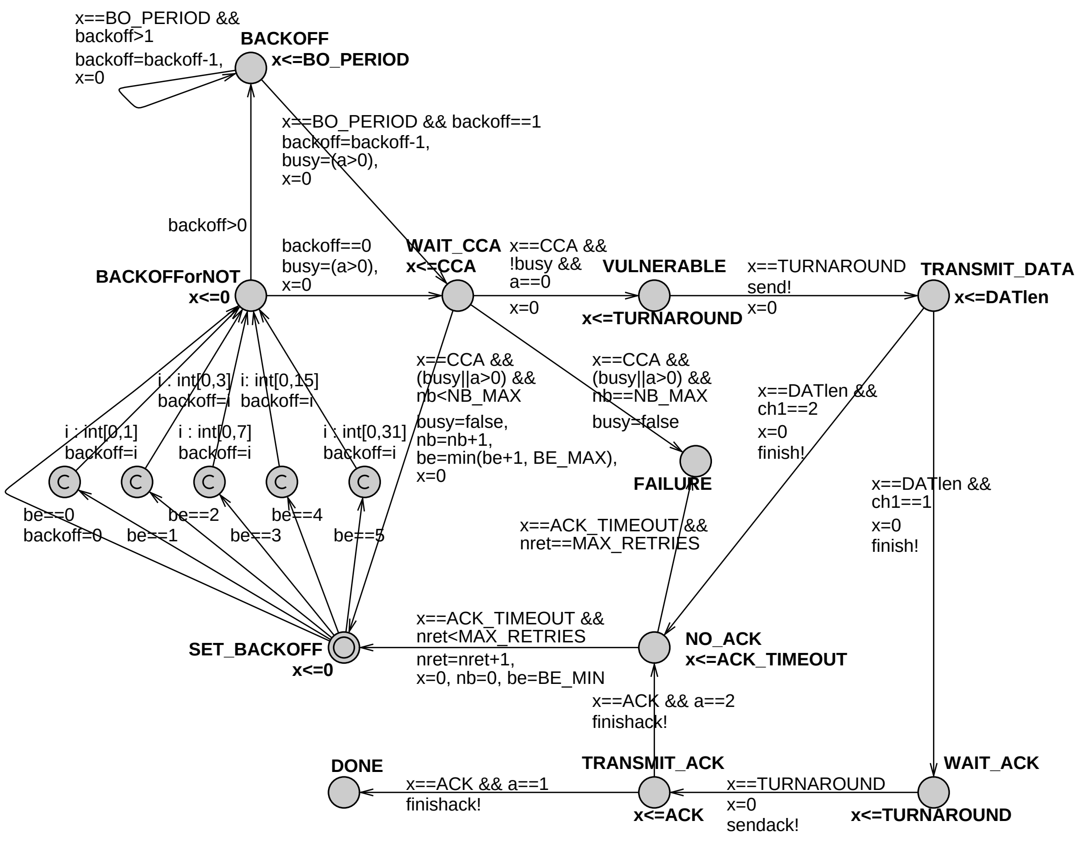


Fig. 10. Model of a hidden station using variable a.

a which is equal to 0 if no acknowledgement is being sent, 1 if being sent but not garbled, and 2 if garbled. Consequently, in states *BACKOFFForNOT*, *BACKOFF* and *WAIT_CCA* of the PTA representing the hidden station s_i, the following variables are tested:

– if $1 \leqslant i \leqslant h$: a and c_j for all j, $1 \leqslant j \leqslant h + m$, $j \neq i$,
– if $h + m + 1 \leqslant i \leqslant n$: a and c_j for all j, $h + 1 \leqslant j \leqslant n$, $j \neq i$.

Furthermore, in the PTA representing any hidden station s_i, i.e. for $1 \leqslant i \leqslant h$ and $h+m+1 \leqslant i \leqslant n$:

– the transition leading from state *WAIT_ACK* is labelled with action *sendack*$_i$,
– in the guards of the transitions from state *TRANSMIT_ACK*, a is tested instead of c_i to see whether the acknowledgement sent has collided or not, and the transitions are labelled with action *finishack*$_i$.

Figure 10 shows the template of the timed automaton model of a sending station which is hidden from all the other stations in the network except the destination one.

The PTAs representing stations s_i which are within the range of all the other stations, i.e. for $h + 1 \leqslant i \leqslant h+m$, are the same as those representing the hidden ones, except that in the guards of the transitions from states *BACKOFFForNOT*, *BACKOFF* and *WAIT_CCA*, a and c_j for all j, $1 \leqslant j \leqslant n$, $j \neq i$, are tested.

If we do not want to write all the (P)TAs representing the hidden stations in PRISM or, respectively, in UPPAAL from scratch, we need a template for the hidden stations with the lower indices and another

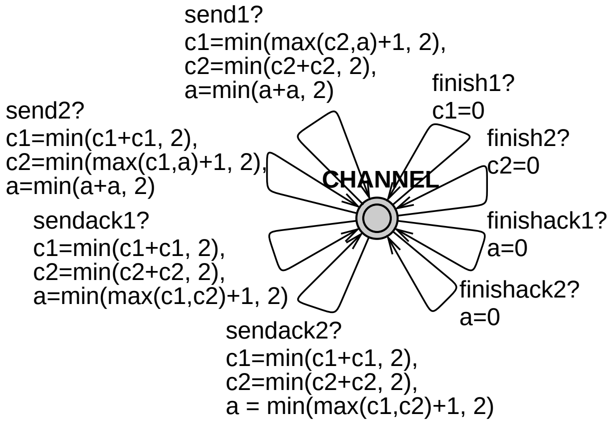


Fig. 11. Model of medium using a for a star network with two sending stations.

one for those with the higher ones if the number of the former is not the same as the number of the latter because in this case the automata differ in the number of the c variables they must check.

The PTA representing the medium differs from the PTA described in Section 5 in that all the transitions take variable a into account besides variables c_i, $i = 1, \ldots, n$, and that transitions labelled with *sendack*$_i$ and *finishack*$_i$ for $i = 1, \ldots, n$ are added. Figure 11 shows the model of the medium for the case $n = 2$.

All the commands of the medium can have the guards equal to true and have the following effect for $i = 1, \ldots, n$:

- the effect of *send*$_i$:
  - $c'_i = \min(\max(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n, a) + 1, 2)$,
  - $c'_j = \min(c_j + c_j, 2)$, for $j = 1, \ldots, n$, $j \neq i$,
  - $a' = \min(a + a, 2)$;
- the effect of $finish_i$: $c'_i = 0$;
- the effect of *sendack*$_i$:
  - $c'_j = \min(c_j + c_j, 2)$, for $j = 1, \ldots, n$,
  - $a' = \min(\max(c_1, \ldots, c_n) + 1, 2)$;
- The effect of *finishack*$_i$: $a' = 0$.

It can be seen that the transitions labelled with *sendack*$_i$ and, respectively, *finishack*$_i$ have the same effect for $i = 1, \ldots, n$. Again, because of binary synchronisation, in the model in UPPAAL built by following this approach it suffices to replace them with one transition labelled with *sendack* and, respectively, *finishack*, and use the same labels in all the PTAs representing the stations.

Since the solution with variable a requires a lot of variables, we next propose one which is an adaption of the solution for n stations from Section 5 with the variables $sending$ and $colind$. The idea is to have the latter indicate whether the currently sent frames are garbled as before, and two variables similar to $sending$, both initialised to 0. Given the network shown in Fig. 9, let variable $sending1$ denote the number of frames being sent at a time by stations r and s_i, $i = 1, \ldots, h + m$, i.e. within the common range denoted by the lower circle in that figure (let us call it range 1). Likewise, let variable $sending2$ denote the number of frames being sent at a time by stations r and s_i, $i = h + 1, \ldots, n$, i.e. within the common range denoted by the upper circle in the figure (let us call it range 2). When a data frame is sent by a station that is only within range 1 (respectively, range 2), only $sending1$ (respectively,

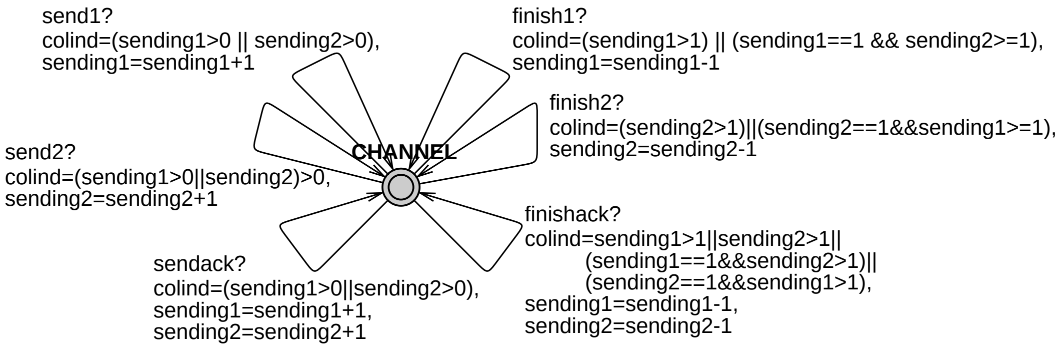

send1?
colind=(sending1>0 || sending2>0),
sending1=sending1+1
finish1?
colind=(sending1>1) || (sending1==1 && sending2>=1),
sending1=sending1-1
finish2?
colind=(sending2>1)||(sending2==1&&sending1>=1),
sending2=sending2-1
send2?
colind=(sending1>0||sending2)>0,
sending2=sending2+1
CHANNEL
finishack?
colind=sending1>1||sending2>1||
(sending1==1&&sending2>1)||
(sending2==1&&sending1>1),
sending1=sending1-1,
sending2=sending2-1
sendack?
colind=(sending1>0||sending2>0),
sending1=sending1+1,
sending2=sending2+1

Fig. 12. UPPAAL model of medium using $sending$ variables for star network with two mutually hidden stations.

$sending2$)) is incremented by one. $colind$ is set to true if something is already being sent in range 1 or range 2 because of the assumption of zero propagation delay. When a data frame or, respectively, an acknowledgement frame is sent by a station which is within both ranges, the variables $sending1$ and $sending2$ are incremented by one and $colind$ is set as in the former case. As in the case with no hidden stations, once $colind$ is true, it must stay so until all the stations involved in the collision finished sending. During the CCA period, those stations that are only within one range only check the $sending$ variable representing the latter. Those stations that are within both ranges check both $sending$ variables before sending a data frame to see if the channel is clear.

Again, the model in PRISM consists of n *Station*$_i$ modules representing PTAs of the same form as the TA in Fig. 2, except for the following, and of a medium module.

For $i = 1, \ldots, n$, in the guards of the transitions from states *TRANSMIT_DATA* and *TRANSMIT_ACK* of the module for s_i, $colind$ is checked as in the solution with variable $sending$ for the network with no hidden stations (Fig. 7). The transition from state *WAIT_ACK* is labelled with action *sendack*$_i$ instead of *send*$_i$, and the transitions from state *TRANSMIT_ACK* with the action *finishack*$_i$ instead of $finish_i$.

For $i = 1, \ldots, h$ (respectively, for $i = h + m + 1, \ldots, n$), $sending1$ (respectively, $sending2$) is checked in states *BACKOFFForNOT*, *BACKOFF* and *WAIT_CCA* of the PTA representing s_i instead of $sending$ as in the model with no hidden stations. For $i = h + 1, \ldots, h + m$, $sending1$ and $sending2$ are checked – if at least one of them is greater than 0, the channel is busy.

The model of the medium differs from the one for the network with no hidden stations (see e.g. Fig. 8) in that it has actions *sendack*$_i$ and *finishack*$_i$ besides *send*$_i$ and $finish_i$ for $i = 1, \ldots, n$. All the guards can be true, but for the reason already mentioned in Section 5, we have added conditions on $sending_1$ and/or $sending_2$ in the guards to prevent the complaints of the version of PRISM we used.

The transitions labelled with *sendack*$_i$ and *finishack*$_i$ have the same effect for any i, $1 \leqslant i \leqslant n$:

– the effect of *sendack*$_i$:
  * $colind' = (sending1 > 0 \mid sending2 > 0)$,
  * $ending1' = sending1 + 1$,
  * $sending2' = sending2 + 1$;
– the effect of *finishack*$_i$:
  * $colind' = sending1 > 1 \mid sending2 > 1 \mid (sending1 = 1 \;\&\; sending2 > 1) \mid (sending2 = 1 \;\&\; sending1 > 1)$,
  * $sending1' = sending1 - 1$,
  * $sending2' = sending2 - 1$.

The transitions labelled with $send_i$ or $finish_i$ for $i = 1, \ldots, h$ have the following effect:

– the effect of $send_i$:
  - $colind' = (sending1 > 0 \mid sending2 > 0)$,
  - $sending1' = sending1 + 1$;

– the effect of $finish_i$:
  - $colind' = sending1 > 1 \mid (sending1 = 1 \ \& \ sending2 \geqslant 1)$,
  - $sending1' = sending1 - 1$.

The effect of the transitions labelled with $send_i$ and, respectively, $finish_i$ for $i = h + m + 1, \ldots, n$ is obtained from the one for $i = 1, \ldots, h$ by substituting $sending2$ for $sending1$ and vice versa in the above expressions.

The transitions labelled with $send_i$ and, respectively, $finish_i$ for $i = h+1, \ldots, h+m$ have the same effect as the transitions labelled with $sendack_i$ and, respectively, $finishack_i$ for $i = h + 1, \ldots, h + m$.

Figure 12 shows such a model of the medium for a star topology network with two sending stations hidden from each other, i.e. for $n = 2$, $h = 1$, and $m = 0$ in Fig. 9, but note that there is only one *sendack* and, respectively, *finishack* transition. For the reason already mentioned, this is only possible in UPPAAL.

7. Some results of probabilistic model checking

In UPPAAL, the properties to be verified by model checking are expressed in its query language based on Computation Tree Logic (CTL). In PRISM, for PTAs they are expressed in Probabilistic CTL (PCTL). For a quick overview of the property specification, please consult [2,24]. Within the framework of this paper, the experiments were carried out by using the default values of *BE_MAX*, *NB_MAX*, and *MAX_RETRIES*, and the values from 0 to 3 for *BE_MIN*. The values for the other constants (note that they are all related to time) were first set to the standard values mentioned in Section 3, but we soon realised that we had to scale them down by using timescale abstraction in order for the PRISM model checker not to run out of memory for larger values of *BE_MIN* and $DATlen$ at least for networks with two sending stations. We performed the model checking with PRISM 4.0.3 by using the MTBDD engine and the PTA model checking method with digital clocks [15] on a computer with 2 GB of memory. We focused on the networks with bit rate 250 kbps. This means the minimal (respectively, maximal) length of data frames 30 (respectively, 266) symbols (i.e. time units, speaking in terms of time in (P)TA models), the length of acknowledgement frames 22 symbols, and the length of *ACK_TIMEOUT* 54 symbols. An exact timescale abstraction consists in choosing a new unit of time such that it is a common divisor of all constants in clock constraints in the model and dividing all these constants by it [7]. Such an abstraction preserves the properties we want to verify. We wanted to carry out model checking for the minimal and maximal lengths of data frames. We, therefore, chose the new time unit to be 2, giving the new *BO_PERIOD* equal to 10, *CCA* equal to 4, *TURNAROUND* to 6, $DATlen$ to 15 for the shortest frames and 133 for the longest, *ACK* to 11, and *ACK_TIMEOUT* to 27. Since the new time unit is the time needed to transmit two symbols, i.e. 8 bits, it is equal to 32 μs for the bit rate 250 kbps.

This scaling enabled us to verify networks with no hidden stations, as well as with hidden ones, for *BE_MIN* and $DATlen$ up to their maximal values (with a few exceptions for the latter) for $n = 2$ and $n = 3$. We carried out experiments for $DATlen$ equal to 15, 45, 75, 105, and 133. However, for $n = 3$, the verification was feasible only for the MAC without acknowledgements.

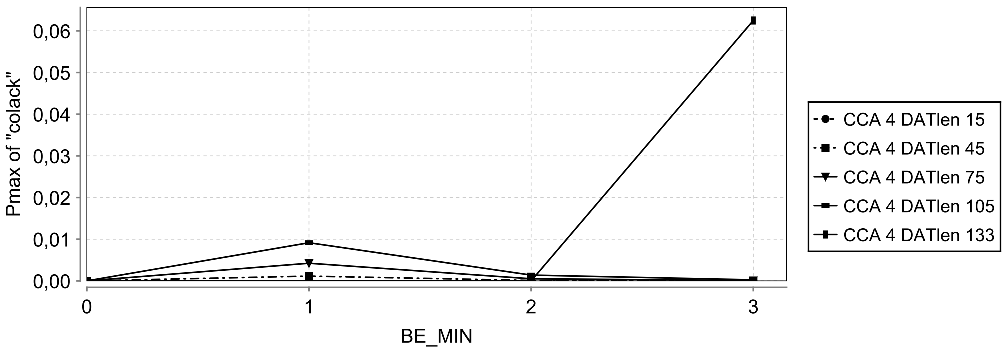


Fig. 13. Probability of collisions involving an acknowledgement frame for standard *CCA* in network with two sending stations and acknowledgements.

We usually first checked the prepared network model for validity with UPPAAL and/or PRISM, typically by verifying whether certain combinations of states or values of variables are possible. For instance, for the models with two stations with acknowledgements, we checked whether it is possible that two acknowledgements are transmitted simultaneously. In UPPAAL, we checked, for example, that formula `E<>(Station1.TRANSMIT_ACK and Station2.TRANSMIT_ACK)` (`E` meaning 'there exists a run' and `<>` meaning 'eventually') was false. In PRISM, the same was expressed with formula `P>0[F((s1=9)&(s2=9))]` (`P>0` meaning 'non-zero probability' and `F` meaning 'eventually').

For all the networks considered, PRISM reported the same number of global states and transitions for the model using the c (and possibly a) variables and the one using variable(s) $sending$. As this is a good indication that these models are equivalent (and valid), we used them interchangeably, and shall, therefore, not indicate which results were obtained using which of them.

7.1. Model checking of networks with no hidden stations

We first experimented with the models of IEEE 802.15.4 MAC with acknowledgements for two stations. In [12], it is claimed that the length of the CCA period being greater than the turnaround time eliminates the possibility for a data frame to collide with an acknowledgement frame from the coordinator. By model checking, we found that this claim cannot be confirmed in the models given in [7,8] because of inaccurate modelling of the clear channel assessment. In our models with no hidden stations, the claim was found to be true. We first checked the probability of collisions involving a data frame and an acknowledgement for the standard value of *CCA*, i.e. 8 symbols or 4 new time units. The results are shown in Fig. 13. This probability is greater than zero for all the combinations of parameter values considered except those with *BE_MIN* $= 0$ and those with the maximal data frame length, for which it is greater than zero only at *BE_MIN* $= 3$. For *CCA* equal to 16 symbols (i.e. 8 new units of time) and to 14 symbols (i.e. 7 units), the result is zero in all the cases.

Further experiments were performed for *CCA* equal to 8 and 16 symbols. We were interested in the probability that both (i.e. all) sending stations successfully send the data frame (let us denote it Pd), expressed, for example, as `Pmin=?[F "done"]` with `"done"` defined to mean that all the sending stations are in state *DONE* (please, note that for all our PTA models, `Pmin` is equal to `Pmax` because they do not contain nondeterminism [15]). For all the models for which we present the verification results in the rest of this paper, Pd for *BE_MIN* $= 0$ is 0 (cf. [8]).

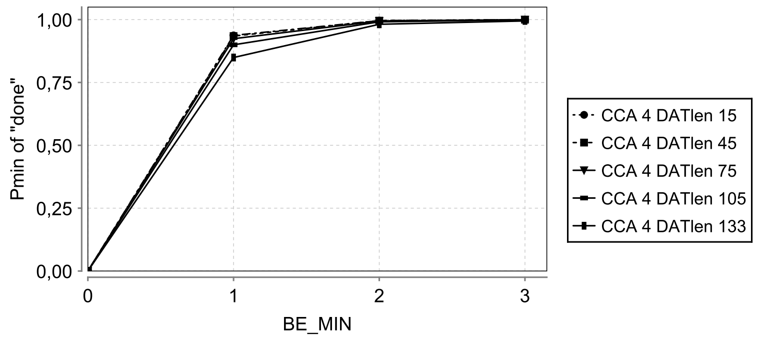


Fig. 14. Probability of "`done`" for standard *CCA* in network with two sending stations and acknowledgements.

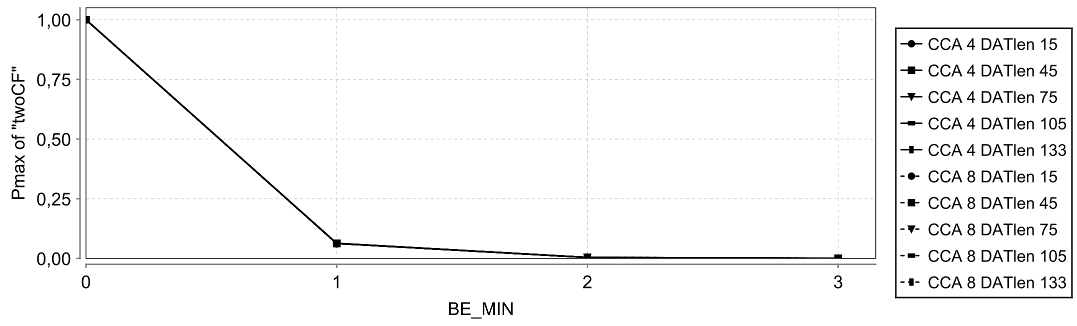


Fig. 15. Probability of a collision failure in network with two sending stations and acknowledgements.

For the network with two sending stations and *CCA* $= 4$, Pd at *BE_MIN* $= 1$ is from around 90 (respectively, 85) percent for $DATlen = 105$ (respectively, 133) to 93 percent for $DATlen = 15$ (Fig. 14). At *BE_MIN* equal to 2 and 3 it is more than 99 percent for all these frame lengths, the largest being at *BE_MIN* $= 3$. Although *CCA* $= 8$ prevents collisions of data with acknowledgements, it improves Pd for all the considered data frame lengths by at most a few tenths of a percent, except at *BE_MIN* $= 1$ and $DATlen = 133$, where Pd increases by about 1.6 percent. This might seem surprising because for the longest data frames and the standard *CCA*, the probability of collisions involving an acknowledgement at *BE_MIN* $= 1$ is zero, whereas it is more than 6 percent at *BE_MIN* $= 3$ (Fig. 13).

Consequently, in both (P)TAs representing the stations, we introduced two states instead of the *FAILURE* state, one representing the channel access failure and pointed to by the transition that tests the condition $nb =$ NB_MAX, and another one representing the collision failure and being the target state of the transition that tests the condition *nret* $=$ *MAX_RETRIES*. By using UPPAAL and PRISM, we verified that for all the combinations of parameter values, for *CCA* equal to 4 and 8, the executions can terminate in one of the following three ways: either both station s_1 and s_2 are successful (denoted "`done`"), both stations declare a collision failure (denoted "`twoCF`" in the sequel), or one of the stations declares success and another one a channel access failure (denoted "`oneCAFoneD`"). Figures 15 and 16 show the probability of the latter two cases. Interestingly, *CCA* $= 8$ causes a little decrease (by at most about a tenth of a percent) in the probability of a collision failure at all the values of *BE_MIN* greater than 0 and $DATlen$ except at *BE_MIN* $= 1$ and $DATlen = 133$, where that probability does not change. For the latter combination, the probability of a channel access failure drops by about 1.6 percent,

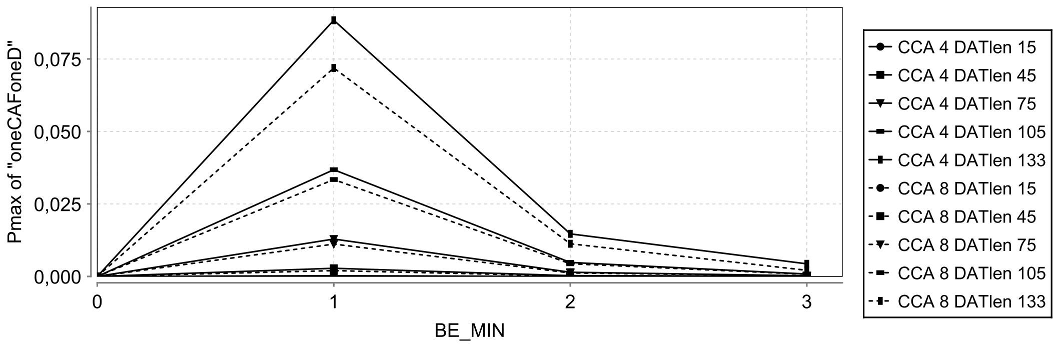


Fig. 16. Probability of a channel access failure in network with two sending stations and acknowledgements.

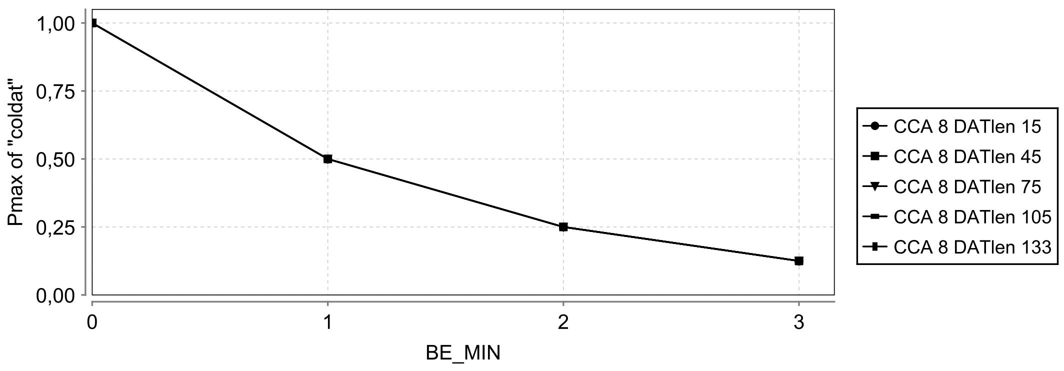


Fig. 17. Probability of collisions of data frames for *CCA* equal to 16 symbols in network with two sending stations and acknowledgements.

which explains the increase in Pd. By looking at (the results used to draw) Figs 15 and 16, it can be seen that generally, the increase of *CCA* to 8 mostly affects the probability of a channel access failure. It should also be mentioned that for $CCA = 8$, at each value of *BE_MIN* the probabilities in Fig. 15 are the same for all the data frame lengths considered. The same holds for the probability that two data frames collide: it is equal to 1 at $BE\_MIN = 0$ and drops by a half for every subsequent *BE_MIN* (Fig. 17).

$CCA = 8$ might prolong the expected time needed to successfully finish the transmission from both stations. Unfortunately, PRISM allows for verification of the expected time needed to reach a certain global state only if the probability of reaching it is 1. So, we could only query about the maximal expected finishing time in the models with limits *NB_MAX* and *MAX_RETRIES* removed as suggested in [7]. To compute the expected time, the reward structure

```
rewards "time"
        true : 1;
endrewards
```

was added to the models, which causes the so-called reward in each state to increase by 1 for each 1 unit of time elapsed [14].

The maximal expected finishing times returned by PRISM upon the query `R{"time"}max=?[F "done"]` for *CCA* equal to 8 (i.e. 256 μs) are from 7 to 14 units (i.e. from 224 to 448 μs) longer than

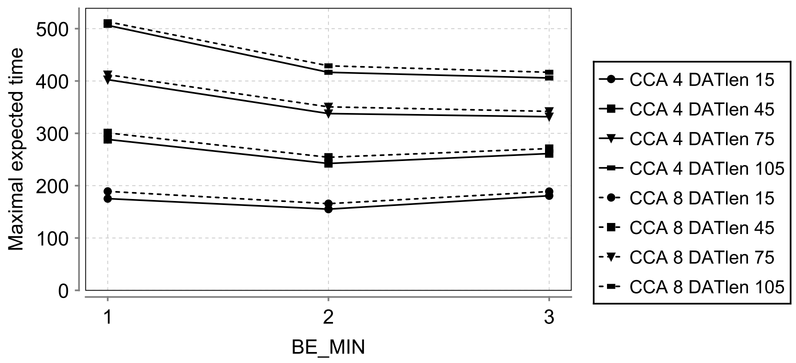


Fig. 18. Maximal expected time to reach `"done"` for *CCA* equal to 8 and 16 symbols in network with two sending stations and acknowledgements.

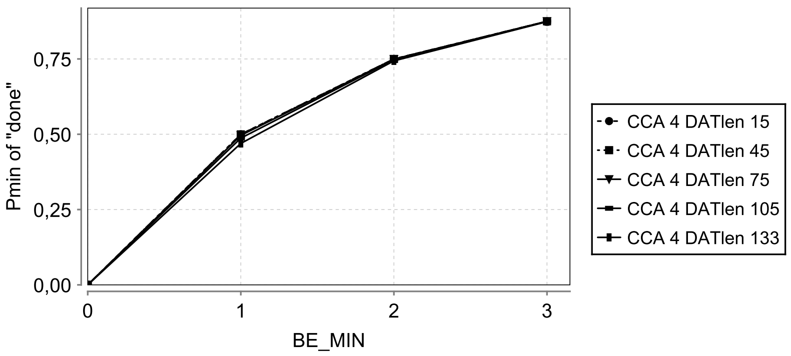


Fig. 19. Probability of `"done"` for standard *CCA* in network with two sending stations without acknowledgements.

for the standard *CCA* (Fig. 18). In both cases, the shortest time for $DATlen$ equal to 15 and 45 was returned at *BE_MIN* $= 2$. For $DATlen$ equal to 75 and 105 (due to the memory limitation, we could not obtain it for 133), it is the best at *BE_MIN* $= 3$. The absolute maximal expected times for $DATlen =$ 15 (i.e. 480 μs of transmission) are less than 200 units (i.e. 6400 μs) and increase by around 70 (2240 μs) to 110 units (3520 μs, i.e. 11 backoff periods) for every additional 30 units (i.e. representing 960 μs of transmission) in $DATlen$.

From the models proposed in this paper, we constructed the models of the MAC without acknowledgements by removing the states *WAIT_ACK*, *TRANSMIT_ACK*, and *NO_ACK*, and their outgoing transitions. The transitions leading from the state *TRANSMIT_DATA* were redirected to state *FAILURE* (for the case of collision) and, respectively, to state *DONE* (for the case of no collision). Please note, that we wanted to verify the probability that the data frames successfully reach the destination and not the probability of success as defined for the protocol without acknowledgements (see e.g. Section 3).

In the model without acknowledgements with $n = 2$, for both values of *CCA*, Pd for $DATlen = 15$ is 0.5 at *BE_MIN* $= 1$, 0.75 at *BE_MIN* $= 2$, and 0.875 at *BE_MIN* $= 3$ (Fig. 19 shows the probabilities for *CCA* $= 4$). For $DATlen$ from 45 to 105 and also 133, Pd is a few tenths of a percent smaller. Pd is a little better in the case *CCA* is 8 than when it is 4, but of course, not because *CCA* $= 8$ would eliminate the collisions with acknowledgements.

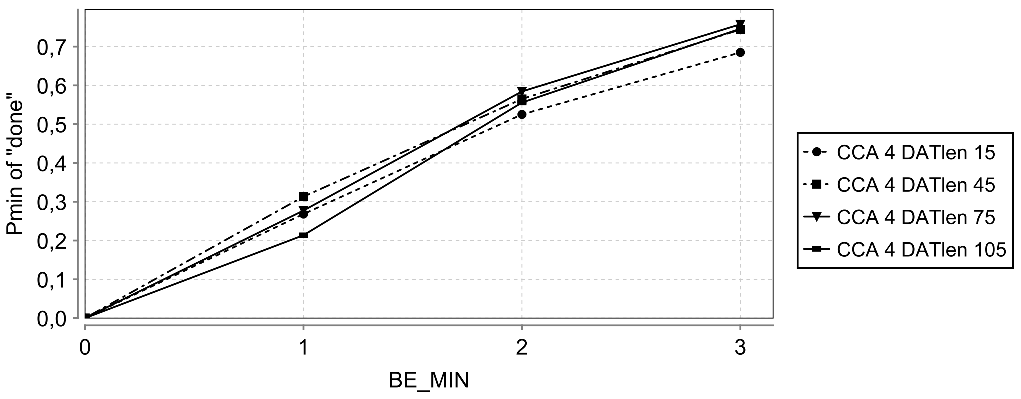


Fig. 20. Probability of `"done"` for standard *CCA* in network with three sending stations without acknowledgements.

Pd is also a little better for *CCA* = 8 than for *CCA* = 4 if $n = 3$. Again, only the values for *CCA* = 4 are shown in Fig. 20 for clarity. It is from around 0.15 to 0.3 at *BE_MIN* = 1, between 0.5 and 0.6 at the latter being 2, and around 0.7 at 3 (we could not verify this for $DATlen = 133$ – the state space contained more than 14 million global states; therefore, the results for $DATlen = 133$ are not shown in the figure). In the networks for $n = 2$ with acknowledgements (Fig. 14) and without them (Fig. 19), the following holds for all the values of *BE_MIN* considered: the greater the value of $DATlen$, the smaller the value of Pd. As evident from Fig. 20, this is not the case for $n = 3$. As expected, at *BE_MIN* = 1, Pd is the worst for $DATlen = 133$ (it is more than 15 percent smaller than the best one, i.e. for $DATlen = 45$), whereas at *BE_MIN* equal to 2 and 3, it is the worst for $DATlen = 15$ (at 3 it is around 6 percent smaller than for the other values of $DATlen$, 133 excluded). The best Pd at 2 and 3 (133 excluded for the latter) is at $DATlen = 75$.

7.2. Model checking of networks with hidden stations

Next, we were interested in Pd for the MAC with acknowledgements in a star network with two sending stations hidden from each other, i.e. in a network of the form shown in Fig. 9 with $n = 2$, $h = 1$, and $m = 0$.

However, we first checked whether *CCA* greater than the turnaround time also prevents collisions of data with acknowledgements in the case of hidden stations. We found that generally, the answer is negative. The answer of PRISM as well as UPPAAL for *CCA* = 8 (i.e. 16 symbols) and *CCA* = 7 (i.e. 14 symbols) was, for example, negative for *BE_MIN* = 1 and $DATlen = 15$ within the mentioned network. Following the negative answer to the query `A[]not(Station1.TRANSMIT_DATA and Station2.TRANSMIT_ACK)` (`A[]` meaning 'on all runs always'), UPPAAL drew a diagnostic trace which confirmed our expectations on how the collision could occur. It can happen that the channel is free and that one station starts to send a data frame. Eventually, the other station finishes the backoff and executes the CCA, but assesses the channel as idle because *it does not hear the transmission*. Before the end of the RX-to-TX turnaround after the CCA at this station, the first station successfully ends the transmission of the data frame. Eventually, the turnaround at the other station is finished and it starts to send a data frame. Soon, the RX-to-TX turnaround at the destination is finished and it starts to send an acknowledgement for the data frame of the first station. It collides with the data frame being sent. Another possibility, also confirmed with UPPAAL, is that the destination starts to send the acknowledgement before the turnaround at the other station is finished. Eventually, the latter starts to send the data

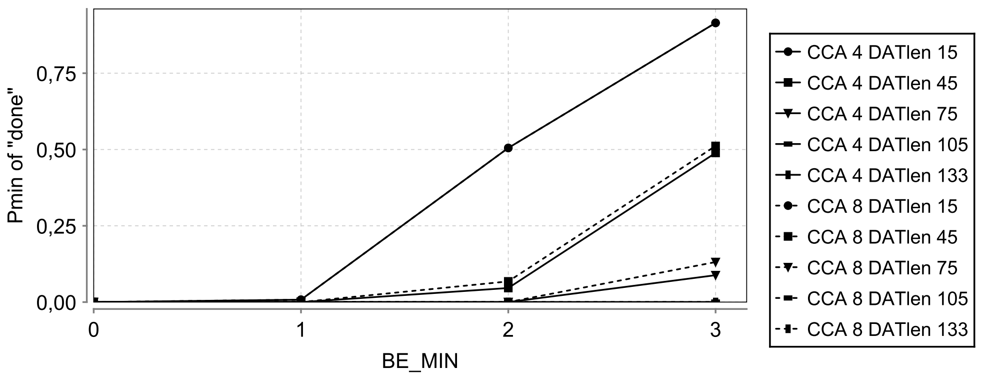


Fig. 21. Probability of "`done`" in star network with two mutually hidden stations and acknowledgements.

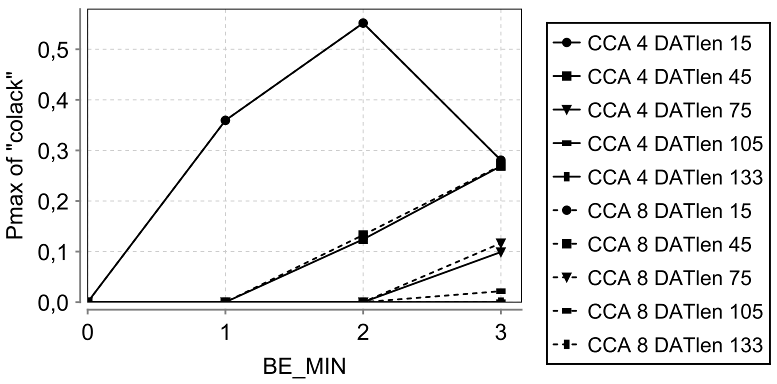


Fig. 22. Probability of collisions involving an acknowledgement frame in star network with two mutually hidden stations and acknowledgements.

frame, which collides with the acknowledgement. It follows that the assumption made in [26], that *CCA* of 16 symbols also ensures an acknowledgement is never involved in a collision in the case of hidden stations, is wrong. In spite of these findings, we conducted further experiments for networks with hidden stations for *CCA* = 4 and *CCA* = 8.

Figure 21 shows that the only non-zero (or close to zero) values of Pd in the network with two hidden stations with acknowledgements were obtained for *BE_MIN* equal to 2 and 3 and $DATlen$ equal to 15, 45, and 75. For the latter, Pd is zero at *BE_MIN* = 2 and around 9 (respectively, 13) percent for *CCA* = 4 (respectively, *CCA* = 8) at *BE_MIN* = 3. Pd for $DATlen$ = 15 is the same for both values of *CCA*: around 50 percent at *BE_MIN* = 2 and 91 percent at *BE_MIN* = 3. For $DATlen$ = 45, Pd is around 5 percent at *BE_MIN* = 2 and around 50 percent at *BE_MIN* = 3, and is circa 2 percent better for *CCA* = 8 than for *CCA* = 4. We can see that *CCA* = 8 improves Pd by around 4 percent for $DATlen$ = 75 and *BE_MIN* = 3.

Figure 22 shows the probability of collisions involving an acknowledgement frame in the network with two hidden stations for both values of *CCA*. For all the combinations of *BE_MIN* and $DATlen$, it either does not change (e.g. in the case of the minimal data frame length) or *increases* for *CCA* = 8. We can see that the changes of Pd in Fig. 21 largely correspond to the changes of the probability in Fig. 22.

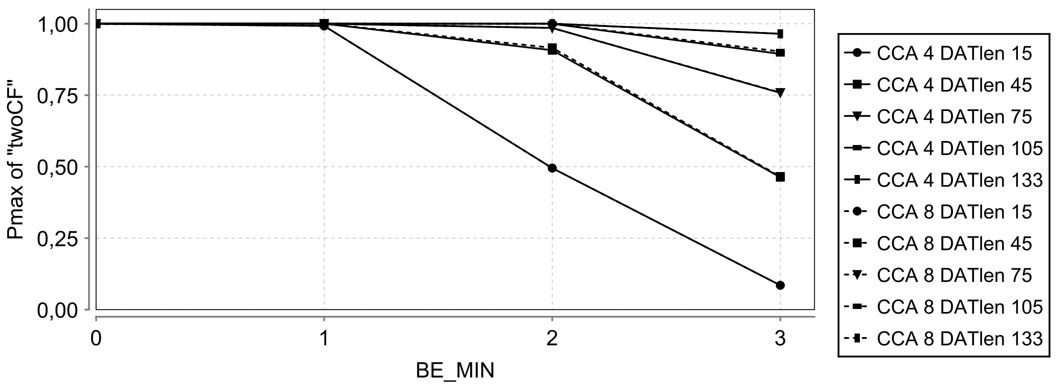


Fig. 23. Probability of a collision failure in both stations in star network with two mutually hidden stations and acknowledgements.

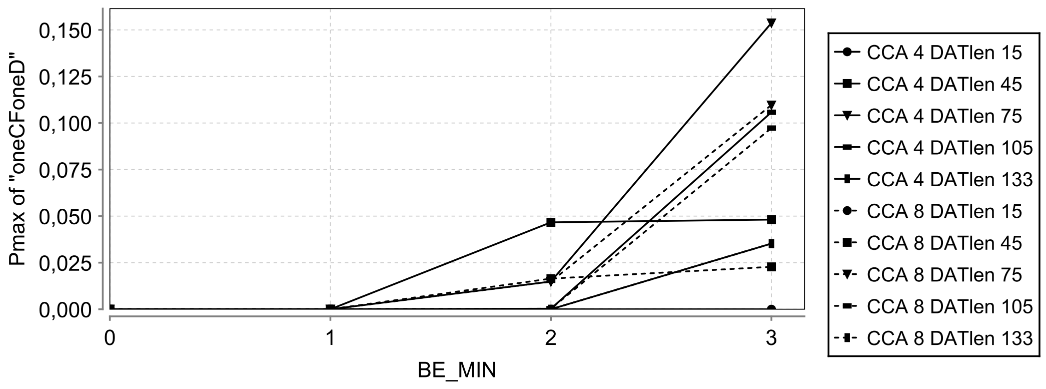


Fig. 24. Probability of a collision failure in only one station in star network with two mutually hidden stations and acknowledgements.

The former improves for $DATlen = 45$ at *BE_MIN* equal to 2 and 3 as well as for $DATlen = 75$ and *BE_MIN* = 3 although the latter increases for these combinations.

In order to find an explanation for this, we introduced two states instead of the *FAILURE* state in the models of both hidden stations in the same way as described in Section 7.1. We verified that for all the combinations of parameter values, for *CCA* equal to 4 and 8, the executions of the network can terminate in one of the following three ways: either both station s_1 and s_2 are successful, both stations declare a collision failure, or one of the stations declares success and another one a collision failure (denoted `"oneCFoneD"`). Figures 23 and 24 show the probability of the latter two cases. Interestingly, there is no chance of a channel access failure, but in contrast to the network with both stations within the same range, it is possible that only one station finishes by a collision failure and another one successfully sends a data frame. By negating this possibility in an UPPAAL query, we obtained a diagnostic trace revealing how this can happen: One station, for example s_1, randomly chooses a smaller *backoff* value than the other, s_2. Station s_1 starts to transmit the data frame first. Eventually, s_2 finishes backoff and since it does not hear s_1, it also starts to transmit the data frame. Eventually, s_1 transmits the complete frame, but does not receive an acknowledgement because its data frame has collided with the data frame

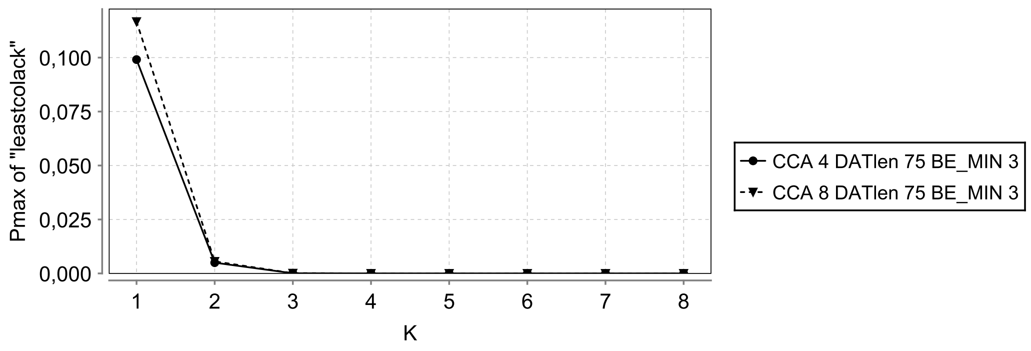


Fig. 25. Probability of at least K collisions involving an acknowledgement in case of two mutually hidden stations, *BE_MIN* equal to 3, data frame length 150 symbols, and *CCA* 8 or 16 symbols.

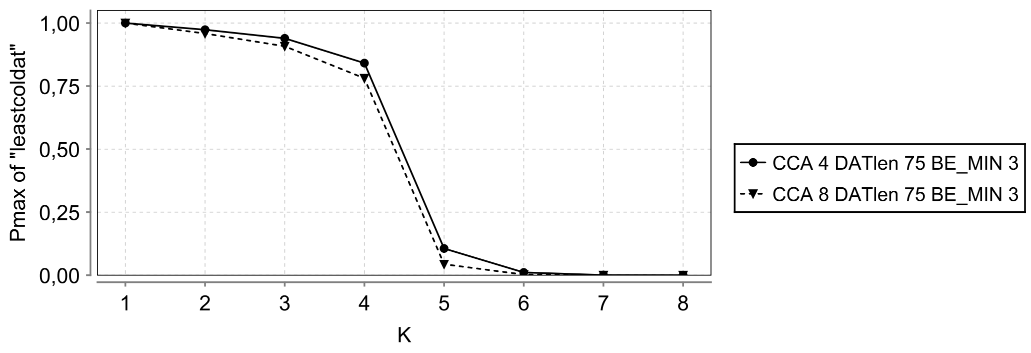


Fig. 26. Probability of at least K collisions not involving an acknowledgement in case of two mutually hidden stations, *BE_MIN* equal to 3, data frame length 150 symbols, and *CCA* 8 or 16 symbols.

of s_2. Hence, s_1 repeats the CSMA-CA algorithm. Before it sends the data frame again, s_2 finishes the transmission and repeats the algorithm for the same reason. A similar situation as at the first attempt of both stations can happen for additional three times (if *MAX_RETRIES* $= 3$). However, the values of *backoff* may be chosen in such a way that when s_1 finishes the data frame transmission for the fourth time, s_2 finishes the data frame transmission for the third time. An acknowledgement is sent neither to s_1 nor to s_2. Station s_1 eventually declares a collision failure, whereas s_2 can successfully accomplish the last allowed attempt.

Altogether, we can see that the improvement of Pd for *CCA* $= 8$ in Fig. 21 is the consequence of a smaller probability of collision failures in the network (please note, that some of the probabilities in Fig. 23 increase, but only by less than a percent). This probability is the consequence of a smaller probability of a larger number of collisions. The fact of the latter in spite of the increased probability of collisions involving acknowledgements can be explained as follows: It is important to notice that Fig. 22 shows the probability of at least one collision involving an acknowledgement. By recording the number of such collisions in the model, up to some limit K (cf. [8]), we found that for all the cases where Pd improves, the increase in the probability of at least K collisions of this kind is negligible for K greater than 1 (see e.g. Fig. 25 for the case of $DATlen = 75$ and *BE_MIN* $= 3$). In a similar way, we found that the probability of at least K collisions involving only data drops for the larger values of K (between 3

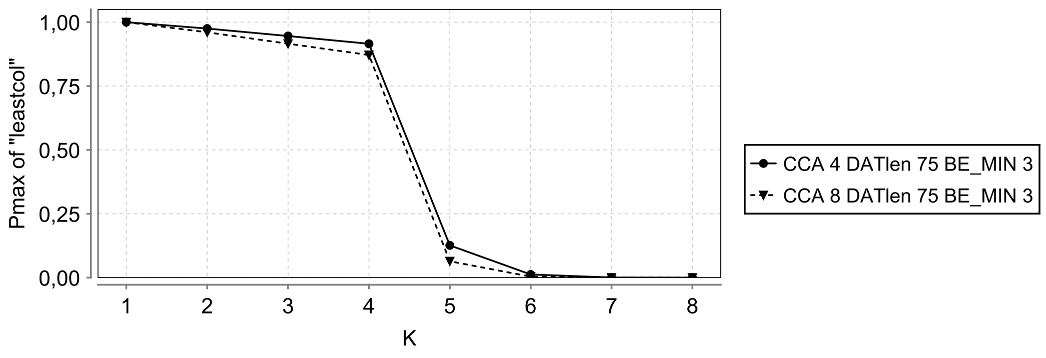


Fig. 27. Probability of at least K collisions in case of two mutually hidden stations, *BE_MIN* equal to 3, data frame length 150 symbols, and *CCA* 8 or 16 symbols.

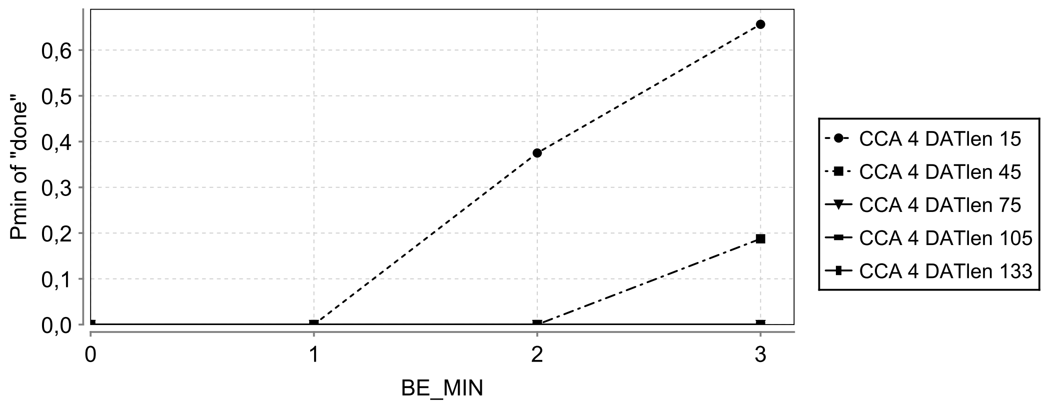


Fig. 28. Probability of `"done"` for standard *CCA* in star network with two hidden stations without acknowledgements.

and 5 or so) by an amount close to the increase of Pd (see e.g. Fig. 26). Consequently, the probability of at least four collisions of either kind (four being the minimal number that causes a collision failure if *MAX_RETRIES* is equal to 3) decreases by a similar amount (see e.g. Fig. 27).

As can be seen from Fig. 28, Pd within the network consisting of two mutually hidden stations not using the acknowledgement option is non-zero only for $DATlen = 15$ in the case of *BE_MIN* $= 2$ and *BE_MIN* $= 3$, as well as for $DATlen = 45$ at *BE_MIN* $= 3$. It is the same for *CCA* equal to 4 and 8.

Finally, we conducted experiments for two networks with 3 sending stations, some of them being hidden, and no acknowledgements. One network contained, besides the destination node, two mutually hidden stations like the last one considered, and an additional one between the two, being within the range of both like the destination. This corresponds to the case with $n = 3$, $h = 1$, and $m = 1$ in Fig. 9. From Fig. 29 we can see that in contrast to the network with $n = 3$ and non-hidden stations (Fig. 20), it holds for any *BE_MIN* greater than zero (with one little exception at 1 and 2), that the smaller the $DATlen$, the greater the Pd. In all the cases, Pd is some tenths of a percent better at *CCA* $= 8$.

The other network was of the form shown in Fig. 9 with $n = 3$, $h = 1$, and $m = 0$. This means a network with the destination and one sending station hidden from a pair of sending stations, the latter being able to hear one another. As shown in Fig. 30, Pd within this network is non-zero only at *BE_MIN*

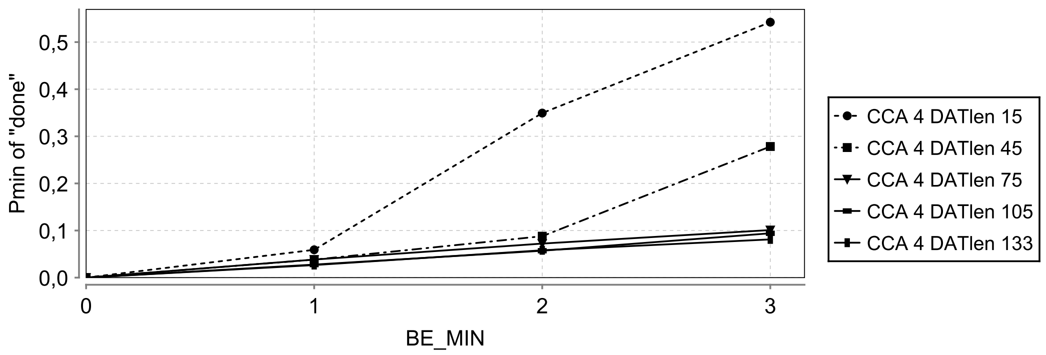


Fig. 29. Probability of "`done`" for standard *CCA* in star network with a sending station between two mutually hidden stations and without acknowledgements.

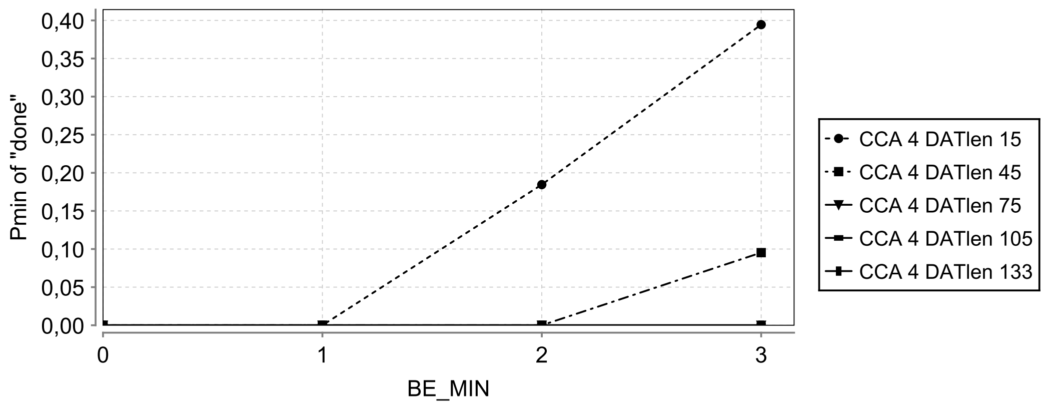


Fig. 30. Probability of "`done`" for standard *CCA* in star network with a sending station hidden from a pair of stations and without acknowledgements.

equal to 2 and 3 for $DATlen = 15$ and at *BE_MIN* = 3 for $DATlen = 45$ as within the network with two mutually hidden stations without acknowledgements (Fig. 28). It is, however, around 10 to 15 percent smaller than in the latter. For *CCA* = 8, it is some tenths of a percent greater than for *CCA* = 4.

By comparing Figs 29 and 30 we can see that the probability that the data frames of all the three stations come through within the network with one sending station within the range of two mutually hidden stations is better than in the network with two stations on one side of the coordinator and the other one hidden from them on the other for all the checked values of $DATlen$. It is around 5 (at *BE_MIN* = 1) to 15 percent (at the larger values of *BE_MIN* and $DATlen$) better. The latter observation as well as the comparison of Fig. 29 with Fig. 28 suggest that the presence of a sending station within the range of two mutually hidden (groups of) sending stations helps that Pd is not zero for the longer data frames.

8. Discussion and conclusions

Based on the model given graphically in [7,8], in this paper we first prepared an accurate PTA model of a nonbeacon-enabled IEEE 802.15.4 network consisting of two sending/receiving station pairs in

PRISM. We subsequently derived the models for networks with n station pairs. We proposed two kinds of models, one with a c variable for every sending station and one with common variables $sending$ and $colind$ for all the stations. The latter is similar to the UPPAAL model recently presented in [3]. The essential difference between them is in that the UPPAAL model consists only of automata representing the stations and no model of the medium. This is achieved in such a way that the action representing the start of sending is equally named (e.g. $busy!$) in all the stations and is declared as a broadcast action, which is allowed by UPPAAL, but not by PRISM. A transition labelled with such an action synchronises with all currently enabled transitions labelled with the complementary action ($busy?$) in other components. All the stations in the model can execute a transition labelled with $busy?$ over the CCA period and during the transmission. In this way, any stations that are interested can hear if another station has started to send.

We further argued that the presented models for n sending/receiving station pairs are also valid models of star topology networks with n sending stations and a coordinator if the latter only sends acknowledgements. Finally, we showed that by relatively small variations, models of typical star-shaped networks with mutually hidden stations can be obtained from them. We could not find formal models of such networks in the literature. As for the star topology networks without hidden stations, MODEST [10] and SPIN [20] have been applied to model the MAC in beacon-enabled mode. A MODEST specification for nonbeacon-enabled mode without acknowledgements can be found at [19]. In all these models, the coordinator is represented separately, and the CCA is represented accurately in neither of them.

We illustrated the usages of the proposed models by presenting verification of the probability that all the stations successfully send a data frame. We were especially interested in the effect of the CCA duration longer than the standard one in those networks using the acknowledgement option. Due to the state-space explosion, we could only verify networks with two and three stations, the latter only without acknowledgements. Nevertheless, we obtained some interesting results on the effect of *CCA*, unknown from the literature. In [9], the effect of *CCA* is studied for nonbeacon-enabled networks with the continuous sending of new data frames and no hidden stations by using classical stochastic modelling and simulation. In [26], the performance of similar networks, but consisting of a destination node and mutually hidden stations located within two different ranges around it, is studied by using simulation. The value of 16 symbols is used for *CCA* because of the conviction that it entirely prevents collisions of acknowledgements.

The main contribution of this paper, besides the proposed modelling approaches, is the evidence obtained by model checking that a longer CCA period does not ensure the absence of collisions involving an acknowledgement frame within those networks with hidden stations, but may nevertheless improve the probability of successful sending. A 'side result' of this paper is the models in UPPAAL. Its capability of interactive simulation and diagnostic trace generation, together with the graphical user interface, proved to be a useful supplement to PRISM.

The modelling approach for networks with hidden stations using the c variables can readily be applied to networks containing sending stations within more than two different ranges around the destination. It only needs to be seen which c variables have to be checked within the CCA period in the models of the individual stations with respect to their ranges. The approach using the $sending$ variables could be generalised to more than two different ranges by introducing as many $sending$ variables as there are ranges.

In the future, we will further investigate the effect of *CCA* in nonbeacon-enabled networks containing hidden stations by using probabilistic model checking as well as classical network simulation tools. Clearly, even with better computer equipment it cannot currently be expected that with the model checking of PTAs the networks containing tenths of nodes could be handled. Analytical performance results are usually validated by comparing them with the results of simulation tools. We, however, intend to check to what degree the analytical results in [4] for nonbeacon-enabled networks in which all the stations send one data frame simultanously to the coordinator, as in the presented models, could be confirmed by model checking using PRISM.

References

[1] P. Baronti, P. Pillai, V.W.C. Chook, S. Chessa, A. Gotta and Y. Fun Hu, Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards, *Computer Communications* **30** (2007), 1655–1695.

[2] G. Behrmann, A. David and K.G. Larsen, A tutorial on UPPAAL 4.0 (Updated November 28, 2006), http://www.uppaal.org/.

[3] P. Bulychev, A. David, K.G. Larsen, A. Legay and M. Mikučionis, Computing Nash equilibrium in wireless ad hoc networks: a simulation-based approach, in: *Proc. 2nd International Workshop on Interactions, Games and Protocols (IWIGP2012)*, J. Reich and B. Finkbeiner, eds, EPTCS 87, 2012, pp. 1–14.

[4] C. Buratti and R. Verdone, Performance analysis of IEEE non beacon-enabled mode, *IEEE Transactions on Vehicular Technology* **58** (2009), 3480–3493.

[5] F. Chen and F. Dressler, A simulation model of IEEE 802.15.4 in OMNeT++, in: *6. Fachgespräch Sensornetzwerke der GI/ITG Fachgruppe "Kommunikation und Verteilte Systeme"*, Technischer Bericht der RWTH Aachen AIB 2007-11, Distributed Systems Group, RWTH Aachen University, 2007, pp. 35–38.

[6] L. de Alfaro, *Formal verification of probabilistic systems*, Ph.D. Dissertation, University of Stanford, 1997.

[7] M. Fruth, Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol, in: *Proc. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, 2006, pp. 290–297.

[8] M. Fruth, *Formal methods for the analysis of wireless network protocols*, Ph.D. Dissertation, University of Oxford, 2011.

[9] M. Goyal, W. Xie and H. Hosseini, IEEE 802.15.4 modifications and their impact, *Mobile Information Systems* **7** (2011), 69–92.

[10] C. Groß, H. Hermanns and R. Pulungan, Does clock precision influence ZigBee's energy consumptions? in: *Proc. 11th International Conference on Principles of Distributed Systems (OPODIS 2007)*, E. Tovar, P. Tsigas and H. Fouchal, eds, LNCS 4878, Springer-Verlag, 2007, pp. 174–188.

[11] J.-P. Katoen, Perspectives in probabilistic verification, in: *Proc. 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*, IEEE CS Press, 2008, pp. 3–10.

[12] T.O. Kim, J.S. Park, H.J. Chong, K.J. Kim and B.D. Choi, Performance analysis of IEEE 802.15.4 non-beacon mode with the unslotted CSMA/CA, *IEEE Communications Letters* **12** (2008), 238–240.

[13] A. Koubaa, M. Alves and E. Tovar, A comprehensive simulation study of slotted CSMA/CA for IEEE 802.15.4 wireless sensor networks, in: *Proc. 2006 IEEE International Workshop on Factory Communication Systems*, 2006, pp. 183–192.

[14] M. Kwiatkowska, G. Norman and D. Parker, PRISM 4.0: verification of probabilistic real-time systems, in: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, LNCS 6806, Springer-Verlag, 2011, pp. 585–591.

[15] M. Kwiatkowska, G. Norman, D. Parker and J. Sproston, Performance analysis of probabilistic timed automata using digital clocks, *Formal Methods in System Design* **29** (2006), 33–78.

[16] M. Kwiatkowska, G. Norman, D. Parker and J. Sproston, Verification of Real-time Probabilistic Systems, in: *Modeling and Verification of Real-Time Systems: Formalisms and Software Tools*, S. Merz and N. Navet, eds, John Wiley and Sons, 2008, pp. 249–288.

[17] M. Kwiatkowska, G. Norman and J. Sproston, Probabilistic model checking of the IEEE 802.11 wireless local area network protocol, in: *Proc. 2nd Joint International Workshop on Process Algebra and Probabilistic Methods and Performance Modeling in Verification (PAPMPROBMIV 2002)*, LNCS 2399, Springer-Verlag, 2002, pp. 169–187.

[18] J. Mišić, V.B. Mišić and S. Shafi, Performance of IEEE 802.15.4 beacon enabled PAN with uplink transmissions in non-saturation modes – access delay for finite buffers, in: *Proc. First International Conference on Broadband Networks*, 2004, pp. 416–425.

[19] MoDeST Tutorial, http://depend.cs.uni-sb.de/modesttutorial/index.html.

[20] Z.L. Németh, Model checking of the slotted CSMA/CA MAC protocol of the IEEE 802.15.4 standard, in: *Proc. 2010 Mini-Conference on Applied Theoretical Computer Science (MATCOS-10)*, A. Brodnik and G. Galambos, eds, University of Primorska Press, 2011, pp. 121–126.

[21] T. Park, T. Kim, J.Y. Choi, S. Choi and W. Kwon, Throughput and energy consumption analysis of IEEE 802.15.4 slotted CSMA/CA, *Electronics Letters* **41** (2005), 1017–1019.

[22] 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, IEEE Computer Society, 2006.

[23] Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4-2011, IEEE Computer Society, 2011.
[24] PRISM model checker homepage, http://www.prismmodelchecker.org/.
[25] I. Ramachandran, Changes made to the IEEE 802.15.4 NS-2 implementation, February 7, 2006.
[26] D. Rohm, M. Goyal, H. Hosseini, A. Divjak and Y. Bashir, A simulation based analysis of the impact of IEEE 802.15.4 MAC parameters on the performance under different traffic loads, *Mobile Information Systems* **5** (2009), 81–99.

Tatjana Kapus received her M.Sc. and Ph.D. degrees from the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia, in 1991 and 1994, respectively. She is currently a professor there. She teaches courses on communications networks, formal methods, and programming. Her primary research interest lies in formal methods for the specification and verification of reactive systems, such as communications protocols.