Software and Systems Modeling (2021) 20:505–538
https://doi.org/10.1007/s10270-020-00817-2

REGULAR PAPER

Modeling and simulation of the IEEE 802.11e wireless protocol with hidden nodes using Colored Petri Nets

Estefanía Coronado[1] · Valentín Valero[2] · Luis Orozco-Barbosa[2] · María-Emilia Cambronero[2] · Fernando L. Pelayo[2]

Received: 23 May 2019 / Revised: 17 June 2020 / Accepted: 26 June 2020 / Published online: 26 July 2020

Abstract
Wireless technologies are continuously evolving, including features such as the extension to mid- and long-range communications and the support of an increasing number of devices. However, longer ranges increase the probability of suffering from hidden terminal issues. In the particular case of Wireless Local Area Networks (WLANs), the use of Quality of Service (QoS) mechanisms introduced in IEEE 802.11e compromises scalability, exacerbates the hidden node problem, and creates congestion as the number of users and the variety of services in the network grow. In this context, this paper presents a configurable Colored Petri Net (CPN) model for the IEEE 802.11e protocol with the aim of analyzing the QoS support in mid- and long-range WLANs The CPN model covers the behavior of the protocol in the presence of hidden nodes to examine the performance of the RTS/CTS exchange in scenarios where the QoS differentiation may involve massive collision chains and high delays. Our CPN model sets the basis for further exploring the performance of the various mechanisms defined by the IEEE 802.11 standard. We then use this CPN model to provide a comprehensive study of the effectiveness of this protocol by using the simulation and monitoring capabilities of CPN Tools.

Keywords IEEE 802.11 · QoS · Colored Petri Nets · Simulation · Performance · Hidden terminal

1 Introduction

IEEE 802.11 networks are used in daily operations in both professional and personal spheres, including video streaming, social media and Internet of Things (IoT) applications. The original standard implemented the Distributed Coordination Function (DCF) as basic medium access mode using the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) method to detect collisions. However, DCF is unable to provide traffic differentiation for the variety of services and lacks of efficiency when the number of users increases. To partially mitigate this problem, Quality of Service (QoS) mechanisms in the IEEE 802.11e amendment [1] were introduced through the Enhanced Distributed Channel Access (EDCA) function as an extension of DCF.

Despite Wireless Local Access Networks (WLANs) were initially designed to provide short range connectivity, the tendency of the society to be permanently connected has made this wireless technology being extended to cover mid- and long-range communications and a higher number of devices through amendments such as IEEE 802.11ah [2]. This is especially relevant in outdoor deployments, i.e., urban and rural areas [3], to enable longer transmission coverage. This change in perspective puts the *hidden node* problem in the spotlight on research, in which several users erroneously sense simultaneously the medium as idle, therefore causing collisions and interference [4,5]. To mitigate this effect, the IEEE 802.11 standard introduced the Request to Send / Clear to Send (RTS/CTS) exchange. However, its use is optional

Communicated by Dorina Petriu.

✉ Estefanía Coronado
e.coronado@fbk.eu

Valentín Valero
valentin.valero@uclm.es

Luis Orozco-Barbosa
luis.orozco@uclm.es

María-Emilia Cambronero
memilia.cambronero@uclm.es

Fernando L. Pelayo
fernandol.pelayo@uclm.es

[1] Smart Networks and Services (SENSE), Fondazione Bruno Kessler, Trento, Italy

[2] Albacete Research Institute of Informatics, Computer Science Department, University of Castilla-La Mancha, 02071 Albacete, Spain

due to the significant overhead that it may incur [6]. The hidden terminal problem is intensified when managing QoS requirements of a mixture of services including video streaming [7], VoIP [8], and downloads of high-size files [9], which results also in the coexistence of saturated and intermittent traffic [10,11].

Although all the working groups responsible for the 802.11 amendments have the common objective of optimizing Wi-Fi networks, the specific improvements aimed in each amendment are not necessarily aligned with the remaining ones, especially on a time difference between them. This is precisely the case of the DCF mechanism and the IEEE 802.11e amendment when it comes to long-range WLANs such as in IEEE 802.11ah [12,13]. As a consequence of this misalignment, even a low number of users using QoS differentiation may bring the network to a saturation state in the presence of hidden nodes, thus leading to huge inefficiencies [14]. This heterogeneous ecosystem intensifies even more the necessity of validating in a unified manner the behavior of 802.11e in scenarios covering hidden nodes instead of analyzing each part of the problem separately, i.e., hidden nodes, channel access mechanisms, and traffic prioritization in IEEE 802.11.

Carrying out controlled experiments while taking into account this wide variety of scenarios using experimental testbeds or even high-level simulators is a major challenge. Moreover, relying on real-world testbeds presents the added drawback that commercial wireless cards do not often comply with all the requirements defined in the IEEE 802.11 standards, which makes the validation even more difficult and inaccurate [15]. For example, the difference in performance between the use of burst traffic or interference issues is so small that both phenomena are unable to be distinguished with these methods. A huge body of research has been devoted to study this issue using either simulation and experimentation [16–19], or analytical models aiming at shedding some light to the problem without involving costly and lengthy simulations that may not be able to collect all the hidden insights of the evaluated scenarios. Many of these analytical works are based on stochastic models, and especially on Markov chains [20–24], as well as on Petri Nets (PNs) models [7,25,26].

In this paper, we use Colored Petri Nets (CPNs) [27] to create a flexible and configurable model of the IEEE 802.11 protocol in the presence of hidden nodes with QoS support that allows us to analyze its behavior in various scenarios while jointly considering hidden terminals, various traffic patterns, and different types of services. Colored Petri Nets [28] are a mathematical formalism that enables the description of the behavior of concurrent systems rigorously, with the main advantage of having a graphical and intuitive representation, which allows us to easily create, edit, simulate and analyze the models by using the wide spectrum of existing PN tools. In particular, we have relied on CPNs, which extend the classical model with both data and time, and are supported by a widely-known tool, namely *CPN Tools* [29], which allows us to simulate the model and obtain performance results. Then, we have created a CPN model of the IEEE 802.11e protocol in network deployments considering hidden terminals. In this respect, we first study the effectiveness of the hidden-node mitigation and priority mechanisms in the context of a long-range wireless setup operating under extreme traffic conditions and in the presence of hidden-node issues. Then, we evaluate the performance of wireless network operating under moderate-load conditions and concurrently supporting delay-sensitive and delay-tolerant applications. Our evaluation tool and scenarios set the basis for further evaluation studies in the context of emerging wireless standards, such as the IEEE 802.11ah.

The remainder of the paper is organized as follows. Section 2 provides an overview of the IEEE 802.11 protocol and the main aspects of Colored Petri Nets. In Sect. 3, we present the CPN model proposed for the IEEE 802.11 protocol with priority classes. Section 4 presents the model validation. The evaluation methodology and the results obtained are discussed in Sect. 5. Section 6 describes the related work. Finally, we draw conclusions and lines for future research in Sect. 7.

2 Background

In this section we give an overview of the most relevant properties of the IEEE 802.11 standard, and afterwards we provide an introduction to CPNs.

2.1 IEEE 802.11 protocol

The description of the IEEE 802.11 standard can be divided into two main parts with respect to this work. First, the wireless medium access when introducing traffic priority classes is described. Second, the RTS/CTS message exchange in the presence of hidden terminals is outlined.

2.1.1 Medium access with priority classes

The medium access method most widely used in the 802.11 family of standards is the so-called DCF function, in which all stations compete for the channel with the same priority. In this distributed method, before starting a transmission the stations must sense the medium as free for a period given by the DCF Interframe Space (DIFS). If after this period the medium is still busy, they must start the Backoff algorithm, a channel access arbitration algorithm that estimates the period of time that a station must wait before attempting retransmission due to the deferred operation.

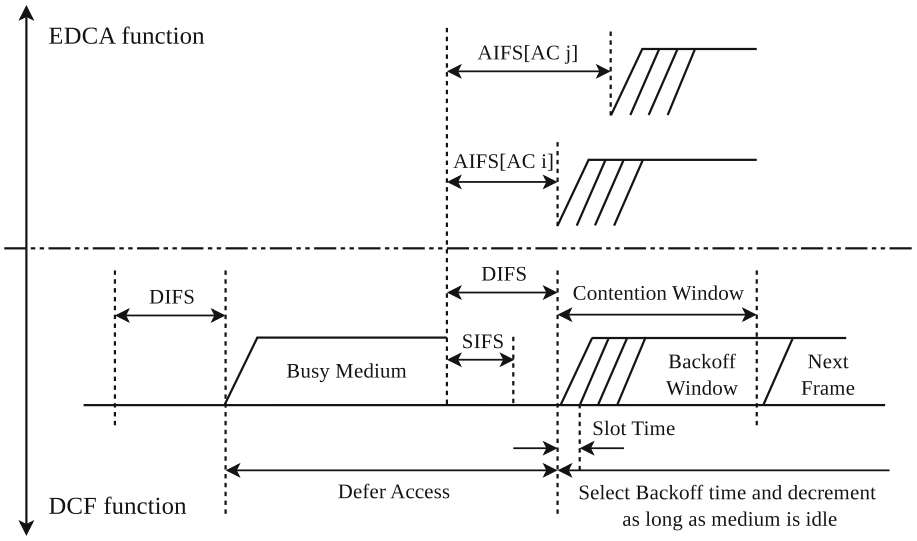


Fig. 1 Interframe spaces in the DCF and EDCA functions

To calculate the waiting time, the Backoff algorithm selects a random value in the interval $[0, CW - 1]$. This interval takes as a parameter the Contention Window (CW), which is given by the physical layer and is represented by the low and upper bounds, CW_{min} and CW_{max}. Initially, the size of CW is set to CW_{min}. After n erroneous transmissions, CW is increased following a sequence of 2^n until reaching the upper bound CW_{max}. If the maximum limit is reached, the frame is silently dropped. Otherwise, the current value is decreased by one every time the channel is sensed as idle for a *DIFS* period. When the counter reaches zero, the transmission begins. Nevertheless, this mechanism gives the same priority to all users and applications. This process is shown in Fig. 1, where the Short Interframe Space *(SIFS)* represents the time used by high priority actions such as *ACKs*. We remind the reader that the standard specifies a two-way handshake protocol for unicast transmissions in which the stations must confirm frame receptions using *ACKs* to ensure transmission reliability [30].

Given the lack of QoS, the IEEE 802.11e amendment [1] introduced EDCA as an extension of DCF while providing differentiated and prioritized traffic access. EDCA tackles this problem by defining four traffic Access Categories (ACs), namely Voice (VO), Video (VI), Best Effort (BE) and Background (BK), from highest to lowest priority. Each AC makes use of its own traffic queue and has its own set of medium access parameters: Arbitration Interframe Space (AIFS), Transmit Opportunity (TXOP) and Contention Window (CW), which is also given by two bounds CW_{min} and CW_{max}.

To ensure compatibility with the stations using DCF while respecting the traffic priorities, the standard establishes a fixed set of values for EDCA. These values are shown in Table 1, where σ represents the duration of the slot time given by the physical layer. Unlike DCF, EDCA specifies different waiting times for each type of traffic. As shown in Fig. 1, a station must sense the channel idle for a period equal to AIFS before attempting a transmission. The AIFS is the translation of the DIFS period but allocated per AC when using QoS mechanisms, i.e., the DIFS is invalidated and four different values are instead used per type of traffic. Conversely, and similarly to DCF, the Backoff algorithm is executed in case of finding the medium busy with the exception of using a specific CW size on an AC basis. Finally, TXOPs enable the transmission of multiple streams without gaining medium access every time that a frame is transmitted and are exclusively used in real-time applications.

2.1.2 RTS/CTS carrier-sensing protocol

Despite the improvements introduced, the two medium access functions described above and the hand-shake pro-

Table 1 Medium access parameters in IEEE 802.11e

AC	CW_{min} (μs)	CW_{max} (μs)	AIFS (μs)	TXOP (ms)
BK	aCW_{min}	aCW_{max}	$SIFS + 7 \cdot \sigma$	–
BE	aCW_{min}	aCW_{max}	$SIFS + 3 \cdot \sigma$	–
VI	$\frac{(aCW_{min}+1)}{2} - 1$	aCW_{min}	$SIFS + 2 \cdot \sigma$	3.008
VO	$\frac{(aCW_{min}+1)}{4} - 1$	$\frac{(aCW_{min}+1)}{2} - 1$	$SIFS + 2 \cdot \sigma$	1.504

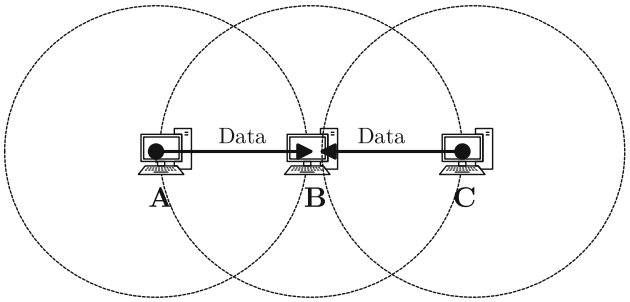


Fig. 2 Hidden node example scenario

tocol for frame acknowledgment have proved not to be enough to guarantee robust transmissions, especially in highly diverse scenarios that can involve hidden nodes. This phenomenon can be observed in the example depicted in Fig. 2. At a given time, station A sends a frame to station B, which cannot be heard by station C, since it is not in the coverage area. Therefore, station C apparently finds the medium idle and starts a new transmission to station B. As a consequence, the two frames collide at the receiver and station B is not able to decode any of them.

With the purpose of addressing this problem, the standard introduces the RTS/CTS protocol (illustrated in Fig. 3), which allows a station to indicate the intention to begin transmission by sending an RTS frame to the Access Point (AP). Upon its reception and if the medium is idle, the AP broadcasts a CTS frame to all the users in the network. Once the source station receives the CTS frame correctly, it can proceed with the transmission. The Network Allocation Vector (NAV), located at the MAC header of the RTS/CTS frames, gives an estimation of the amount of time that the medium will be busy in handling the transmission. On the basis of this information, the remaining stations must update their NAV time during which they are not allowed to sense the status of the channel. Therefore, regardless of their location, this protocol makes it possible to inform all the stations about the beginning of a new transmission, and the waiting time before sensing the channel again. As a result, collisions are reduced and the performance of the network is improved when hidden stations are present.

2.2 Colored Petri Nets

A Petri Net [28] is a directed bi-partite graph with nodes of two types: places (drawn as circles) and transitions (drawn as rectangles). An arc can connect either a place with a transition (pt-arc) or a transition with a place (tp-arc). Let P be the set of places, T the set of transitions, $X = P \cup T$ (nodes) and $F \subseteq (P \times T) \cup (T \times P)$ the set of arcs. For any node $x \in X$ (place or transition), we define the preconditions and postconditions of x, denoted by ${}^\bullet x$ and $x^\bullet$ respectively, as follows: ${}^\bullet x = \{y \in X \mid (y, x) \in F\}$, $x^\bullet = \{y \in X \mid (x, y) \in F\}$.

Places usually represent states or system conditions, while transitions are the actions or events that produce changes in the system state. Places can have an associated *marking*, which is a natural number indicated beside the place (number of *tokens* in it). This number can be used for instance to indicate the number of packets or the number of processes in a queue.

Pt-arcs are also labeled with a natural number (*arc weight*) to indicate the number of tokens required to execute (*fire*) the outgoing transition. The value by default is one, in which case there is no need to explicitly indicate it. The same notation applies to tp-arcs, but in this case the weight indicates the number of tokens to be produced in the outgoing place when the transition is fired. Thus, for a transition to be fireable (*enabling condition*) all its precondition places must have at least as many tokens as the weight of the arc that connects them to the postcondition transition. The firing of a transition t has therefore the following effects:

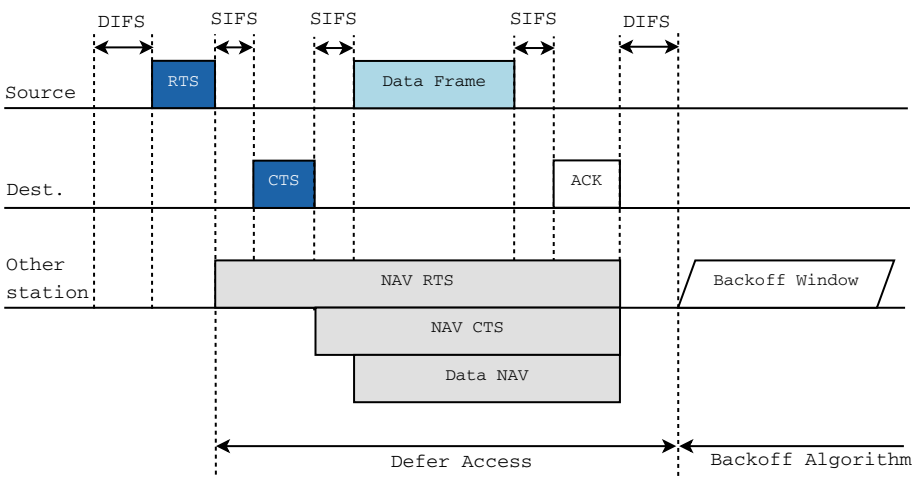


Fig. 3 Working mode of the RTS/CTS protocol exchange

– For each precondition place $p \in {}^\bullet t$, a number of tokens equal to the weight of the pt-arc (p, t) are removed from p.
– For each postcondition place $p \in t^\bullet$, a number of tokens equal to the weight of the tp-arc (t, p) are produced in p.

In the simple Petri net model, no time information is considered and no information can be associated to the tokens and places, which are two important features required to model concurrent systems. CPNs [27] are an extension to the original Petri nets which incorporate data and time, allowing the modeling of complex data structures attached to tokens and time restrictions in the sequence and synchronization of the processes involved. Thus, in CPNs, places have an associated *color set* (a data type), which specifies the set of permitted token colors at a given place. CPNs are supported by a widely-used tool, namely CPN Tools [29], which allows us to create, edit, simulate and analyze CPNs. The notation described below is the one used in this tool.

A place can have no attached information at all, as in the plain model. In this case, we indicate *UNIT* as the color set of the place. But a place can now have as a color set, for instance, the set of integer numbers *INT*, a Cartesian product of two or more color sets as in $INT2 = INT \times INT$, a string (*STRING*), etc. In this case, each token has an attached data value *(color)*, which belongs to the corresponding place color set. Furthermore, we can use the timed features of Colored Petri Nets. In this type of nets, a discrete global clock is used to represent the total time elapsed in the system model; and places can be either timed or untimed. In the case of timed places, their tokens have an associated timestamp, which indicates the time at which they will be available and thus usable to fire a transition. In CPN Tools, the current number of tokens in every place is drawn on a green circle on the right-hand side of the place, and the specific colors of these tokens are indicated by the notation $n`v$, meaning that there are $n$ instances of color $v$. The symbol '++' (respectively '+++' for timed tokens) is used to represent the union of colors in CPN Tools. Thus, a timed integer place (color set *int timed*) with a marking $4`3@5 + + + 6`19@10$ has 4 tokens with value 3 and timestamp 5, and, 6 tokens with value 19 and timestamp 10.

The arc inscriptions are now extended to color set expressions, which are constructed using variables, constants, operators and functions. The arc expressions must evaluate to a color or multiset of colors in the *color set* of the attached place. These colors (tokens) will then be consumed or produced when the corresponding transition is fired, as explained below. Both pt-arc expressions and tp-arc expressions can have time inscriptions, with the syntax $expression@x$. In the case of pt-arcs, these timed expressions are used to allow the corresponding transition to consume the tokens that match the expression in advance, i.e., these tokens can be consumed at a time equal to their timestamp minus the value x indicated in the arc expression. On the other hand, in the case of tp-arcs, the tokens produced by evaluating *expression* have an age equal to the current time increased by x time units (0 by default). Delays can also be indicated in transitions, with the syntax $@ + x$, which means that all tokens produced will have the current time plus x as timestamp. Furthermore, transitions can have guards that restrict their firing, as well as priorities. Guards are Boolean expressions constructed by using the variables, constants, operators and functions of the model, and they must evaluate to true for the transition to be *fireable*. Transitions can also have an associated priority, so in the event of a conflict between two transitions that can be fired at a given time, the one with the highest level of priority is fired first. We only use two levels of priority, namely *P_NORMAL* (by default) and *P_HIGH*, the latter being higher than *P_NORMAL*. Priorities are indicated on the lower left corner of transitions, but in the case of *P_NORMAL*, these are not usually shown.

Let us now see how arc expressions are evaluated and the firing rule for CPNs. For any transition t with variables $x1, x2, \ldots$ in its input and output arc expressions, we call a *binding* of t an assignment of concrete values to each of these variables. A binding of a transition t is then *enabled* if there are tokens in its precondition places matching the values of the corresponding inscriptions. For instance, for the pt-arc expression $2`x + +4`y$ we need to assign variable x a value such that we have 2 tokens with this value in the input place, and a value to y such that we have at least 4 tokens with this value in the input place. Thus, arc expressions are evaluated by assigning values to the variables, and these values are then used to select the tokens that must be removed or added when firing the corresponding transition. When no transition can be fired at the current time, time elapsing occur, but only up to a time at which some transition can be fired.

Example 1 Figure 4 shows a CPN modeling a simple protocol. In this CPN, places *Start, TimeOut, St_fail, Sending* and *St_ok* have the *INTt* color set, which stands for timed integer (*int timed* in CPN Tools). Place *Medium* has the $INT2 = INT \times INT$ color set. Transition *Success* has a guard associated (*s=1*), meaning that it can only be fired when the variable s is bound to 1.

At the initial marking only the places *Start* and *Medium* are marked. A function *M_INIT* is used to produce an initial marking for place *Start* (see Listing 1). Functions in CPN Tools are defined using an extended version of the ML language [31], and *M_INIT* has a natural number as argument, which indicates the number of tokens (stations) to be produced at the *Start* place. Notice that the *dexp* function is used to produce the timestamp for each token, which is generated by using an exponential distribution with rate 1/50. Therefore, the figure depicts the three tokens produced with

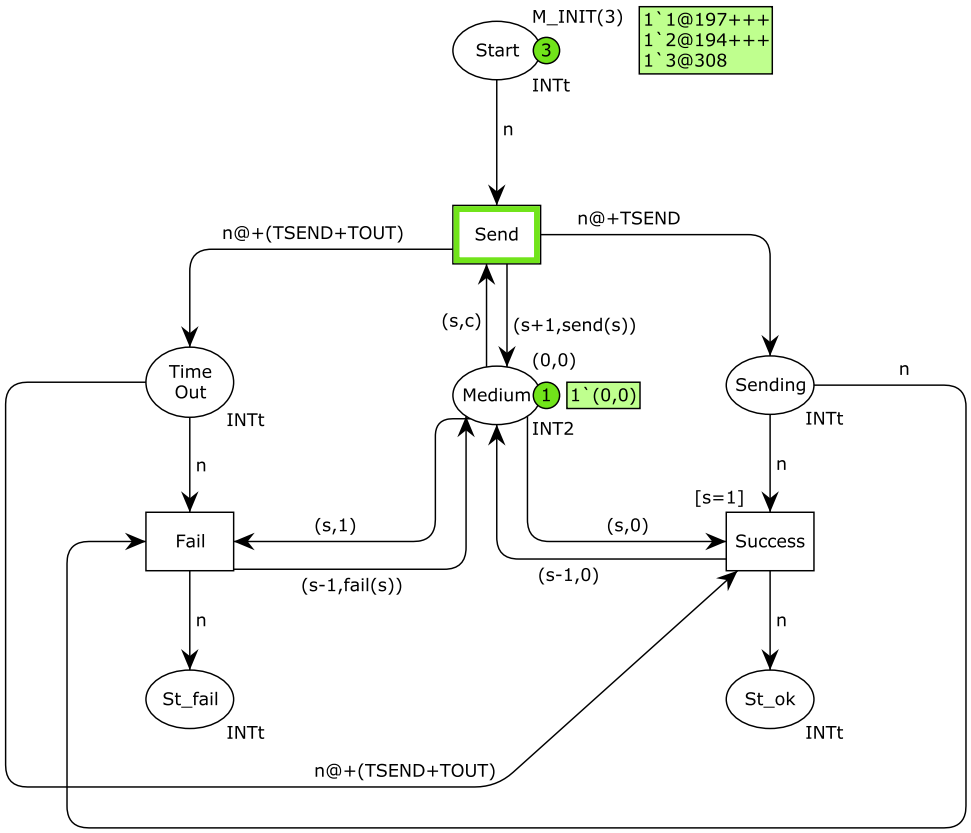


Fig. 4 CPN modeling a simple protocol

Listing 1 Constant definitions and functions.

```
1 val TSEND=20;
2 val TOUT=10;
3 val tmed=50.0;
4 fun dexp(r)=round(exponential(r));
5 fun M_INIT(n)=
6 if (n=0) then nil
7 else 1‘n@+dexp(1.0/tmed)+++M_INIT(n−1);
8 fun send(s)=if (s=0) then 0 else 1;
9 fun fail(s)=if (s=1) then 0 else 1;
```

integer values 1, 2, 3 (representing station identifiers), and timestamps 197, 194 and 308, respectively.

Place *Start* models the initial station state, which consists of three stations which will start to transmit a message at their corresponding timestamps. Place *Medium* stands for the communication medium state. This place will always be marked with a single token. The first field of this token represents the number of stations that are currently trying to transmit a message, while its second field is a Boolean indicating whether there has been a collision. Thus, the initial marking of place Medium is $(0, 0)$.

A station starts the transmission by firing the *Send* transition, which produces one token in place *TimeOut* and one token in place *Sending*. The token in *Sending* will be available after *TSEND* time units, while the token in *TimeOut* will be available after $TSEND + TOUT$ time units. In addition, the medium state is changed, increasing the number of stations trying to transmit and updating the collision state using the *send* function, which assigns the collision state to 1 when several stations are trying to transmit simultaneously.

At time 194 transition *Send* fires, producing the token 1‘2@214 in *Sending* and the token 1‘2@224 in *TimeOut*. Moreover, the token in *Medium* changes to $(1, 0)$. Then, at time 197 station 1 starts its transmission, also producing one token in *Sending*, with value 1‘1@217 and one token in *TimeOut* with value 1‘1@227. Notice that place *Medium* also changes to $(2, 1)$, indicating a collision. Neither of these transmissions can therefore be successful, so transition *Fail* is fired twice (notice the arc inscription $(s, 1)$ from *Medium*), at times 224 and 227. The firing of *Fail* also removes the corresponding token for station n in place *Sending*. Thus, after both firings place *Medium* is marked again with $(0, 0)$ and there is one token in place *Start* with value 1‘3@308.

Finally, station 3 starts at time 308, firing the *Send* transition, producing both tokens in *TimeOut* and *Sending*, and changing the marking of *Medium* to $(1, 0)$. Then, transition *Success* fires at time 328, which also removes the corresponding token from *TimeOut*. Notice the timed inscription used in this pt-arc to consume this token in advance: $n@ + (TSEND + TOUT)$. This pt-inscription, as indicated in Sect. 2.2, means that this token can be consumed

$TSEND + TOUT$ time units before its time stamp. Place *St_fail* collects the failed transmissions, while place *St_ok* collects the successful transmissions.

The CPN in Fig. 4 represents a very simple system. In general, we will have to deal with larger CPNs in which the use of the hierarchical features of CPN Tools will allow the decomposition of the model into various smaller subnets. These subnets, called *pages* in the CPN Tools terminology, should be linked using substitution transitions and fusion sets. Substitution transitions refer to transitions replaced by subnets represented in other pages, while fusion sets are sets of places used in different pages, which are functionally identical and therefore correspond to the same place from a formal viewpoint. In this paper, we have made use of fusion places to split the model into pages. The link of these pages is made through their common places, denoted by a blue fusion label at their left bottom corner.

3 IEEE 802.11 CPN model

In this section we first present the characteristics of the network system based on the IEEE 802.11 standard (and specifically on the IEEE 802.11e amendment) considered in the CPN model, and then we introduce the proposed model by describing its CPN pages.

3.1 IEEE 802.11 network system configuration

The problem analyzed in this paper verses on the impact of the priority mechanisms introduced in IEEE 802.11e. However, the system modeled does not aim to independently evaluate the performance of this mechanism of the standard, but to analyze its behavior when it is enabled in any Wi-Fi network, where we may find a broad set of traffic patterns and services, i.e., intermittent/saturated traffic, different bitrates, etc., channel states, and topologies. Moreover, it is our standpoint that the proposed system model must be as reusable as possible for any Wi-Fi network, i.e., any version of the standard, to pursue innovation by facilitating its modification and adaptability to any new version. With this objective in mind, the following constraints must be taken into account:

- The traffic priority mechanisms introduced in IEEE 802.11e must be a common factor in all the physical layers and can be used across different versions of Wi-Fi networks. Therefore, the variables defining both the physical and the channel access layers must be configurable enough for allowing further analysis of other versions of the protocol. In this respect, we can highlight physical and channel access timings, such as slot time, AIFS, CW_{min} and CW_{max}, as introduced in Sect. 2.1.
- The model must not enforce a specific transmission rate (which depends on the physical layer and is independent on the priority mechanisms) or a exact packet length for any traffic type. In this respect, both parameters must accept modifications regardless of the service type.
- Any Wi-Fi network must be able to manage traffic with different nature: burst or saturated traffic. However, the frequency of burst, i.e., intermittent, traffic can highly differ from each other. In this sense, traffic nature must be modeled using a parameter that allows setting its inter-arrival period and frequency.
- The model must not assume any network topology, user density or range. For that reason, the dimension of the network must be configurable. This constraint is translated into the impossibility of knowing at any moment if a given user is in the coverage range of any other. For that reason, regardless of the network dimensions, the possibility of enabling the RTS/CTS mechanism introduced in Sect. 2.1.2 must be included. It should be noted that one of the main objectives of our analysis is to verify the effectiveness of the priority mechanisms of the IEEE 802.11 protocol in the presence of hidden nodes.

The previous constraints can be specifically translated into network specific parameters to be considered by the CPN model to cover a wide range of representative scenarios. Such a model is based on the following considerations:

- **Network topology.** The network consists of N stations randomly distributed in a way that they cannot overhear each other, with the goal of studying the effect of the hidden terminal problem.
- **PHY/MAC layers.** The network is modeled using configurable timings and transmission rates in order to analyze the behavior of current and future physical layers in the standard.
- **Channel conditions.** Communications take place where transmitted data are never affected by errors in the channel, apart from those caused by deferring time of the stations and frame collisions.
- **Traffic bitrate**. A station is either under *(i) saturated condition*, i.e., it always has a packet to transmit, or follows *(ii) an intermittent transmission*, based on a negative exponential distribution. Moreover, the data can be delivered at a given bitrate regardless of the distribution.
- **Traffic type.** Each station transmits a unique type of traffic as given by the IEEE 802.11e amendment, i.e., *(i) background*, *(ii) best effort*, *(iii) video*, or *(iv) voice* traffic.
- **RTS/CTS control.** The RTS/CTS exchange can be *(i) enabled*, or *(ii) disabled* for all the stations in the network.

Table 2 Parameters used for the CPN model

Variable	Category	Description	Value
`Payload (P)`	PHY/MAC layers	Packet length used for all the traffic	Scenario-dependent
`PHY Header`	PHY/MAC layers	Length of the physical layer header	120 bits
`MAC Header`	PHY/MAC layers	Length of the MAC layer header	28 bytes
`Slot Time` (σ)	PHY/MAC layers	Duration of the channel slot time	20 μs
`SIFS`	PHY/MAC layers	Duration of the Small Interframe Space	10 μs
`AIFS [AC]`	PHY/MAC layers	Duration of the Arbitration Interframe Space	$SIFS + AIFSN[AC] \cdot \sigma$
`CWMIN[AC]`	PHY/MAC layers	Min. contention window per traffic type	(31, 31, 15, 7)
`CWMAX[AC]`	PHY/MAC layers	Max. contention window per traffic type	(1023, 1023, 31, 15)
`ACK Size`	PHY/MAC layers	ACK length of all traffic types	14 bytes
`Bitrate`	Traffic bitrate	Bitrate of the data packets	2 Mbps
`Basic Bitrate`	Traffic bitrate	Bitrate of the control packets	1 Mbps
`Distribution[AC]`	Traffic bitrate	Saturated/intermittent transmissions	Scenario-dependent
`Exponential rate`	Intermittent Traffic Workload	Time between new frames for intermittent stations	1.0/166881.0
`n1p[AC]`	Traffic type	List of the number of stations per traffic type delivering saturated traffic and ordered as (BK, BE, VI, VO). If any value = 0, then the specific traffic type is not present.	Scenario-dependent
`n2p[AC]`	Traffic type	List of the number of stations per traffic type delivering intermittent traffic and ordered as (BK, BE, VI, VO). If any value = 0, then the specific traffic type is not present.	Variable
`Time RTS`	RTS/CTS control	Duration of the RTS frame (enabled if $RTS > 0$)	20 bytes
`Time CTS`	RTS/CTS control	Duration of the CTS frame (enabled if $RTS > 0$)	14 bytes
`Time Send`	All categories	Overall transmission time of each traffic type calculated from the rest of variables	Scenario-dependent

Table 2 indicates the resulting parameters from the constraints described before, which can be adjusted for the analysis of any scenario. In particular, the integer variables shown in the table represent the specific values that have been maintained all along the experiments performed. Note that they have been selected with the aim of being compatible with IEEE 802.11b/g [32] (using long preamble option) given that most common wireless cards on the market follow these standards. However, the same study can be extended to any other physical layer by modifying the values of the PHY/MAC layers. Conversely, the values denoted as *variable* are further described in Sect. 5 given that they are modified across the various experiments to evaluate a wider number of scenarios. Notice how the variable names shown in this table are intended for facilitating human comprehension, therefore being the specific variable names used in the model described in Listing 2. Finally, a particular case is found in the variable *Time Send*, which can be defined as a common denominator across all the categories defined, since its value depends on the ones chosen for the rest of the variables.

The transmission times of a data frame and the corresponding acknowledgment are given by T_{data} and T_{ACK}. The data rate follows the 802.11b/g PHY, taking 1 and 2 Mbps for the basic (R_B) and transmission rate (R_{TX}), respectively. Hence, the transmission time (defined in the model as *TIME_SENDING*) varies depending on the scenario and the type of traffic. Finally, T_{RTS} and T_{CTS} provide the times needed by the RTS/CTS exchange.

$$T_{data} = AIFS + \frac{PHY_{header}}{R_B} + \frac{(MAC_{header} + P) \cdot 8}{R_{TX}} \quad (1)$$

Listing 2 Specific parameters used in the CPN model.

```
val M=10000000; (* A large number *)
val RTS = 0; (* 1: RTS/CTS used. 0: No RTS/CTS *)
val nt1 = 0; (* #st producing consecutive messages *)
val nt2 = 5;(* #st producing messages with exp. *)
val r2 = 1.0/166881.0; (*Exp. rate for nt2 stations.*)
val np = 4; (*Number of priorities *)
val n1p = (0,0,0,0); (* #st_1 by types of priority *)
(* #BK, #BE, #VI, #VO *)
val n2p = (0,0,3,2); (* #st_2 by types of priority *)
val TIME_SENDING=Tdata; (* Eq. (1) *)
val nt = nt1+nt2; (* Total number of stations *)
val TSEND_RTS = T_RTS; (* Time to send RTS message, Eq.
    (3) *)
val TSEND_CTS = T_CTS; (* Time to send CTS message, Eq.
    (4) *)
val TSEND_ACK = T_ACK; (* Time to transmit ACK, , Eq. (2)
    *)
val TTIMEOUT = T_ACK; (* Added to compute time−out *)
val TOUT_CTS = TSEND_RTS + TSEND_CTS + TTIMEOUT;
(* Time−out to detect RTS collision *)
```

$$T_{ACK} = SIFS + \frac{PHY_{header}}{R_B} + \frac{(MAC_{header} + ACK_{size}) \cdot 8}{R_{TX}} \tag{2}$$

$$T_{RTS} = AIFS + \frac{PHY_{header}}{R_B} + \frac{(MAC_{header} + RTS_{size}) \cdot 8}{R_{TX}} \tag{3}$$

$$T_{CTS} = SIFS + \frac{PHY_{header}}{R_B} + \frac{(MAC_{header} + CTS_{size}) \cdot 8}{R_{TX}} \tag{4}$$

3.2 CPN network model

In this section we describe the CPN model for the 802.11 protocol. This is a configurable and flexible model, which allows us to have different versions of the protocol by changing the model parameters, as indicated in Table 2 and more specifically in Listing 2, where we show a configuration example. For instance, a Boolean parameter *RTS* (see Listing 2) allows us to have two versions of the protocol, one using the RTS/CTS mechanism and another one not using this mechanism. In the CPN model, two station types are considered. On the one hand, stations that produce messages consecutively, i.e., as soon as a message has been delivered, a new message is in the queue to be sent, thus producing saturated traffic on the medium; and, on the other hand, stations that produce messages according to an exponential distribution, with the rate $r2$, as indicated in Listing 2.

The *n1p* parameter is an array representing the number of stations of each type (background, best effort, video, and voice) that send messages consecutively, while *n2p* represents the number of stations of each type that send messages by using the exponential distribution.

Thus, we can assign different values for *nt1*, *nt2*, *n1p* and *n2p* in order to model a variety of scenarios in the protocol execution. Listing 2 shows a scenario in which there are no stations sending consecutive messages ($nt1 = 0$) and there are 5 stations sending intermittent traffic using an exponential distribution ($nt2 = 5$). In this scenario, we have 3 stations sending video traffic and 2 stations sending voice traffic, in both cases following an exponential distribution (*n2p*). The other parameters are evident from their names. Notice that *TIME_SENDING, AIFS, CWMIN* and *CWMAX* are also defined depending on the station priority, i.e., on the type of traffic. There are also a significant number of ML functions in the CPN Model, which are required for several purposes, such as to initialize markings, to compute expressions in arc inscriptions, etc. These functions can be found in Listing 5 in Appendix A.

In this CPN, *INTt* is the timed integer color set. In general, *INTn* denotes a product color set of n integer types: $INTn = \overbrace{INT \times \ldots \times INT}^{n}$, and *INTnt* denotes its timed version.

The CPN model consists of eight pages, which are described below. Fusion places are then used to identify the shared places across these pages.

3.2.1 ProdMessages CPN page

In this CPN page depicted in Fig. 5, the initial messages to be sent by each station are produced in the *Last_Arrival_Time* place. These initial messages are generated by the function *M_MSG*, which generates a timed integer token for each station. The value of this token is the station number, and its timestamp is the time at which this message can be sent. As indicated in Listing 2, there are *nt1* stations of type-1 (producing messages consecutively), and *nt2* stations of type-2 (whose messages are generated using an exponential distribution). Therefore, stations are numbered so that the first *nt1* stations are of type-1, and the remaining ones are of type-2.

Transition *New_MSG* is fired when a message can be sent. In particular, in addition to this, for type-2 stations the function *gen_new_time* produces a new token in *Last_Arrival_Time* to prepare the next message for this station, using the exponential distribution with rate $r2$. Place *Total_MSG_(Ev)* is a counter for the messages produced for type-2 stations, so it is updated when *NEW_MSG* is fired. Once *NEW_MSG* has been fired, a new token is written into place *Time_Arrival_MSG*, which represents the input message buffer.

A station starts processing a message by firing the transition *init_st*, which configures the station parameters (place *ST_CONF*). This place contains the station configuration

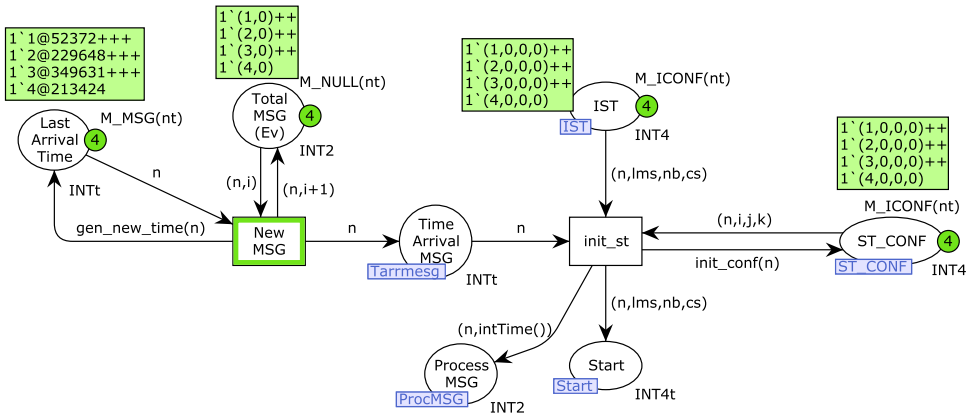


Fig. 5 ProdMessages CPN page

with the following fields: station number, AIFS value, CWMIN, and CWMAX. Thus, the treatment of each station depends on the specific information written in the place *ST_CONF* for that station, which is taken from the different traffic types. The other precondition place of *init_st* is *IST*, which stores the station state information, with the following fields: station number, last medium state known (0 free, 1 busy), Backoff needed (Boolean) and CTS received (Boolean).

Once the transition *init_st* is fired, one token containing the station state information is written into the *Start* place to initiate the protocol. In addition, a token is written into the *Process_MSG* place, which indicates the time at which this station started this transmission (function *intTime()* returns the current simulation time).

3.2.2 Start_Station CPN page

In this CPN page illustrated in Fig. 6, the place *MEDIUM* contains one token in the color set INT2 for each station, with its local medium state information. The first field is the station number and the second field indicates whether the station is transmitting a message or not. We must recall that this message cannot be seen by any other station, except the AP. Function *M_NULL* is used to initialize the marking of place *MEDIUM* to $(n, 0)$ for each station n. Place *AP* stands for the Access Point state information. This place always contains one token, with three integer values. The first field represents the number of incoming transmissions, the second field indicates whether the AP is sending a message (which is visible for all the stations), and the third field is usually 0, but in the event of a collision its value is 1, and it is set to 2 when a CTS message has been sent. The initial marking of place *AP* is therefore 1‘(0, 0, 0).

A station starts a new frame transmission by firing transition *start_st*, which removes the corresponding token from place *Start* (fusion place with place *Start* in page *ProdMessages*), and writes one token in place *TM*, delayed by the associated AIFS value.

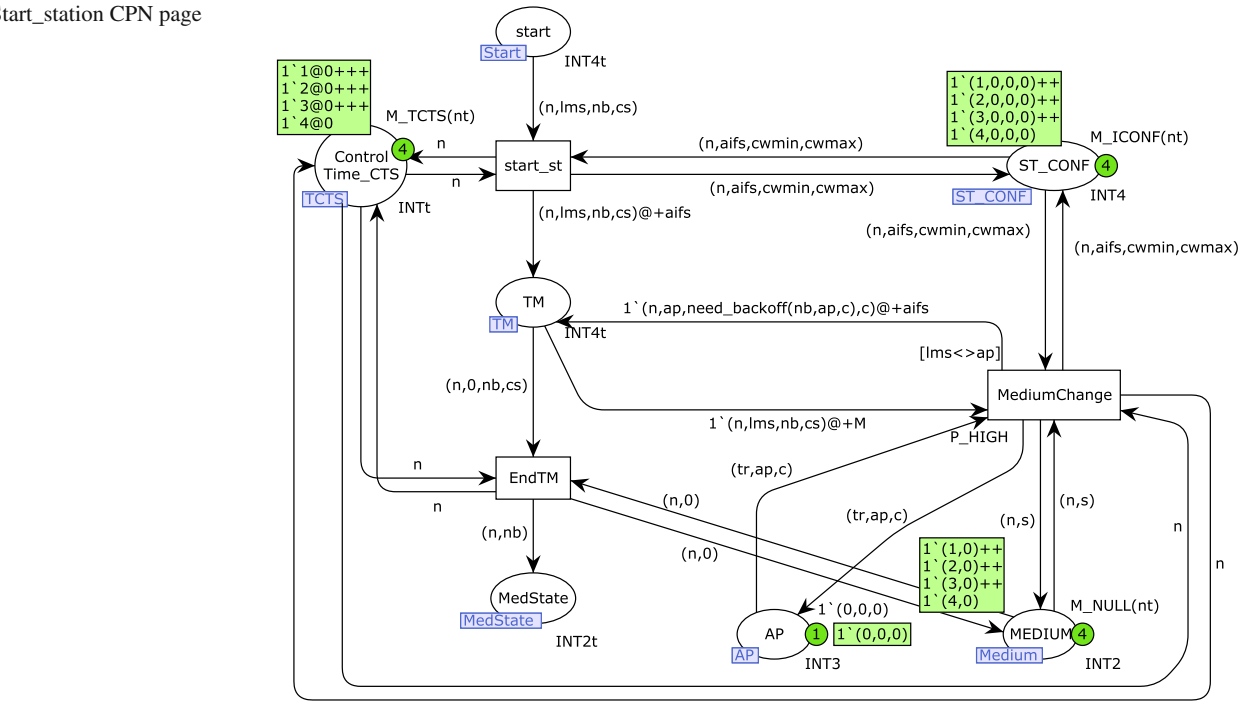
The arc connecting transition *start_st* with place *ST_CONF* (fusion place with *ST_CONF* in page *ProdMessages*) is used to read the specific AIFS value for this station. Thus, we guarantee that the station waits for at least AIFS microseconds before starting the transmission (see Sect. 2.1.1). Actually, when a medium-change condition is detected (transition *MediumChange*, which has high priority), this token in *TM* is updated (delayed by AIFS microseconds), with the newly required Backoff information and CTS-received information, thus guaranteeing that the medium is idle for this time before starting the transmission. Function *need_backoff* returns 1 (backoff required) when the medium is busy or CTS has been received, and function *update_cs* returns 1 when a CTS message has been delivered. Thus, when the last medium state information known by the station *(lms)* differs from the current access point medium state *(ap)*, transition *MediumChange* fires and updates the token in *TM* corresponding to this station.

Transition *EndTM* is fired when the wireless medium has been idle for AIFS microseconds, writing one token in place *MedState*. Note that the value 0 is used in the second field of the pt-arc inscription to indicate this state. Consequently, in this place we indicate the number of the specific station and whether the Backoff algorithm needs to be executed. Finally, place *Control_Time_CTS* is used to stop the stations when a CTS has been delivered, thus establishing a time-out before any other station can start the protocol, except the station indicated in the CTS message.

3.2.3 StartSend CPN page

In this CPN page places *MedState*, *Control_Time_CTS*, *AP* and *Medium* are fusion places, so they are the same as those with the same names in page *Start_station*. The behavior of this CPN page depicted in Fig. 7 depends on the value of the *RTS* constant, which enables and disables the RTS/CTS exchange.

Fig. 6 Start_station CPN page



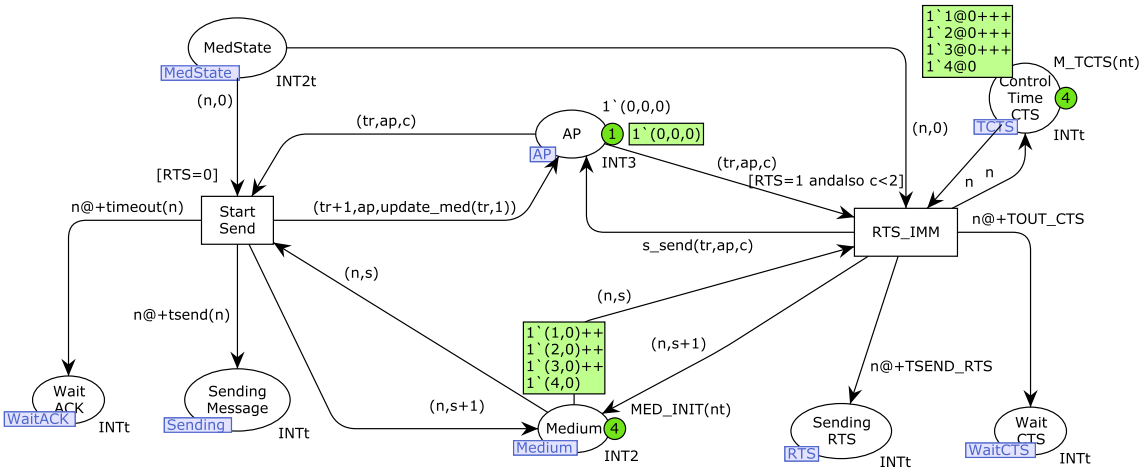


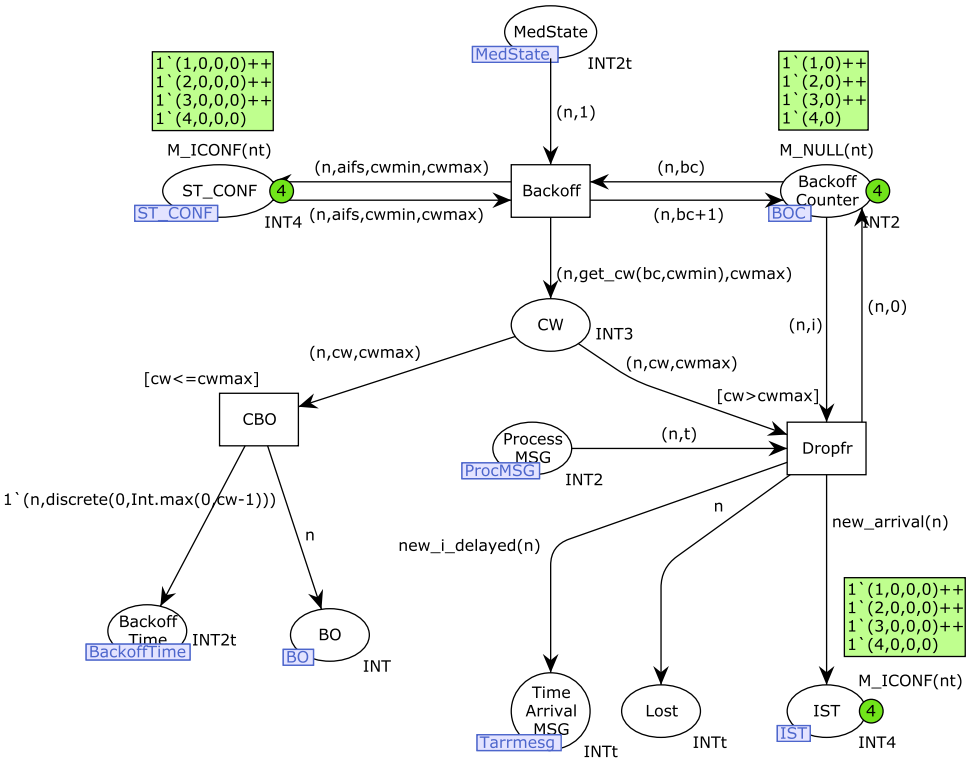
Fig. 7 StartSend CPN page

– **Without RTS/CTS exchange**. Transition *Start_Send* is immediately executed so that the transmission starts. One token is written into the *Wait_ACK* place to activate a time-out (no ACK received), and another one delayed by the transmission time is written into the *Sending_Message* place, to indicate the transmission. In addition, both the local medium state and the AP medium state are changed accordingly (one more station transmitting and collision state updated by function *update_med*, which returns 1 if there were already transmissions in the medium).
– **With RTS/CTS exchange**. In this case, transition *RTS_IMM* is fired, which indicates that an RTS message is being sent. As a result, one token is written with the corresponding timestamp (delayed by the transmission time of RTS message) into the *Sending_RTS* place, and another one into the *Wait_CTS* place in order to activate a time-out (RTS collision).

3.2.4 Backoff CPN page

The *Backoff* CPN page is depicted in Fig. 8. This page contains the first part of the Backoff algorithm, in which either the Backoff time is assigned and the station enters into the Backoff process or a message is dropped when the maximum value of the CW is reached. In this CPN page, places

Fig. 8 Backoff CPN page



ST_CONF, *Time_Arrival_MSG*, *Process_MSG* and *IST* are fusion places, they are the same as those with these names in Fig. 5. In the same way, place *MedState* corresponds to the same place in Fig. 7. Thus, when one token in place *MedState* in the CPN page shown in Fig. 8 indicates that running the Backoff algorithm is required (the second field is 1), the transition *Backoff* fires, which updates the corresponding Backoff counter for this station (place *Backoff_Counter*). When the CW limit is not reached, the firing of transition *CBO* we produce a new token in the *Backoff_Time* place and another one in the *BO* place, which indicate the Backoff times and the stations in Backoff period, respectively. The Backoff time is therefore computed using the algorithm indicated in Sect. 2.1.1. Transition *Dropfr* will only fire in the event that the CW limit has been reached. In this case, the frame is dropped, so it is annotated in the *Lost* place and a new frame is produced for this station using the exponential distribution. Function *new_i_delayed* only produces one token for stations of type 1 (those producing saturated traffic), while *new_arrival* will produce one token in the *IST* place, but indicating that a Backoff period is required in the case of stations of type 1 (Fig. 9).

3.2.5 StartfromBO CPN page

This CPN page contains the second part of the Backoff algorithm and the transitions that start the transmission of either the message or RTS frame, depending on the value of the *RTS* constant. The Backoff time is here decreased to zero by one or more firings of transition *DecrBO*, and then the transmission starts. In this CPN page, places *BackoffTime* and *BO* are the same as those with these names in Fig. 8. In the same way, *AP*, *Medium*, *Wait_ACK*, *Sending_Message*, *Sending_RTS* and *Wait_CTS* are the same as in Fig. 7, and *Control_Time_CTS* is the same place as in Fig. 6.

The Backoff time is updated by firing transition *DecrBO*, which is then fired at each time slot. Furthermore, when a CTS message is received by the station, this transition cannot be fired, since the station must wait for a time-out before restarting its activity. Place *Control_Time_CTS* is used again for this purpose. Once the Backoff time has elapsed, for the RTS/CTS version of the protocol the RTS message is sent (transition *RTS_ABO*), which is analogous to the transition *RTS_IMM* in the *StartSend* CPN page. In the case of the non-RTS/CTS version, transition *Start_SendBO* is fired, which starts the transmission, in the same way as *Start_Send* in the *StartSend* CPN page.

3.2.6 RTS_CTS CPN page

This CPN page illustrated in Fig. 10 is used only if the RTS/CTS exchange is enabled. In this CPN page, places *Sending_RTS*, *Wait_CTS*, *AP* and *Medium* are fusion places, they are the same as those with these same names in Fig. 8. When an RTS message is being sent from any station, a token will be in the *Sending_RTS* place, indicating the sta-

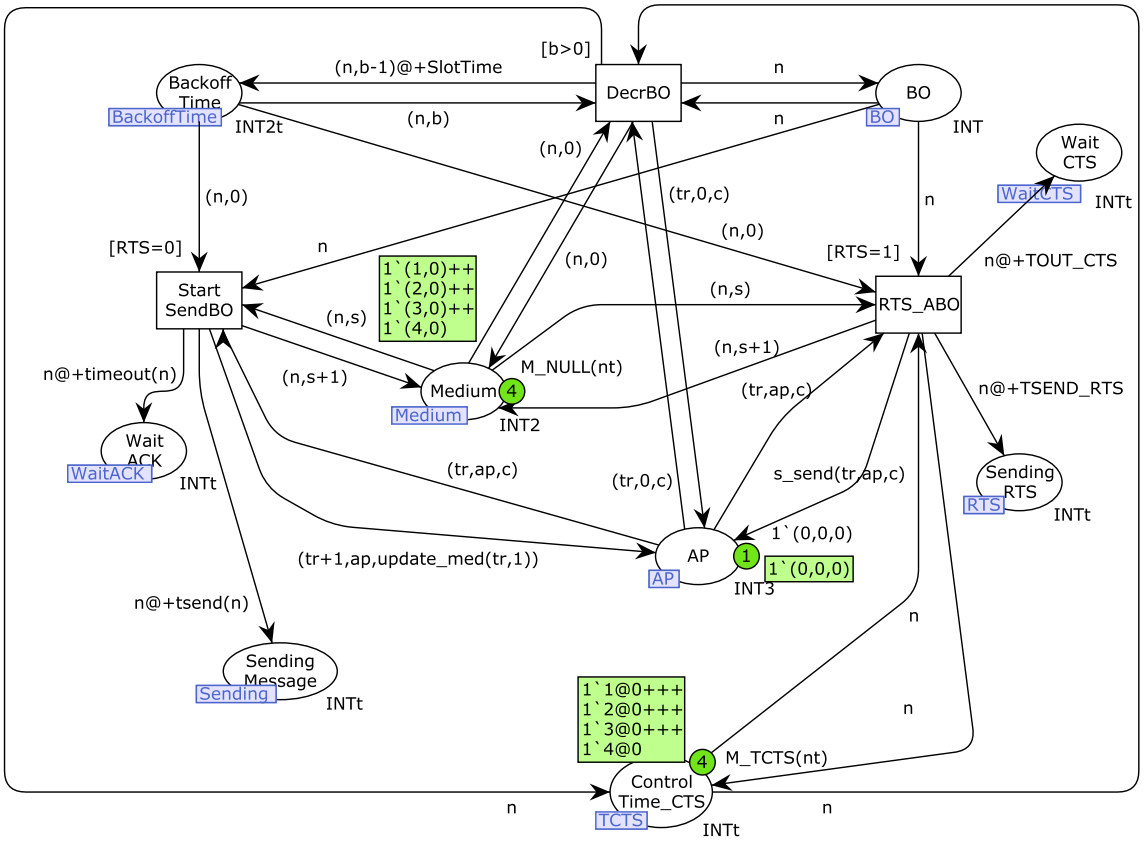


Fig. 9 StartfromBO CPN page

tion number with a timestamp equal to the time at which this transmission finishes. There is also one token for this same station in the *Wait_CTS* place, which is used to detect a collision. Thus, when the RTS transmission is successful (*END_RTS* transition), the *Start_CTS* transition fires, writing one token in the *Sending_CTS* place, with a timestamp equal to the end of the transmission. Otherwise, in the event of an RTS collision, the *Coll_RTS* transition fires, writing a new token in the *Colls_RTS* place, indicating the collision information. In addition, a station collision counter is updated in the *Colls_RTS_ST* place.

Note that after a collision one token corresponding to this station is also written in the *TM* place (see Fig. 6), indicating the last medium information known for this station (*s*) and that a Backoff process is required. Thus, as we saw in the *Start_station* CPN page in Fig. 6, this station will wait for the medium idle for a time equal to AIFS before starting its Backoff period.

Transition *CTS_OK* is fired when the CTS transmission is successful. This transition activates the UpdateTimes CPN page by writing one token in the *Update_times* fusion place. Finally, a CTS collision is detected by transition *Coll_CTS*. This transition can only be fired when there is a collision, when the CTS frame is being sent. Observe the value 1 (collision) in the third field in the expression of the pt-arc from *AP*.

3.2.7 UpdateTimes CPN page

Figure 11 shows the CPN page that updates the time-outs that the stations (except the one that sent the RTS frame) must wait for transmitting once they have received a CTS message. Place *Update_Times* becomes marked with one token when transition *CTS_OK* is fired in the *RTS_CTS* CPN page. The token in the *Stations* place is an integer value that is initially equal to the number of stations. This number is decreased as transitions *Update_other* and *Update_n* are fired, which update the time-outs for the stations in the *Control_Time_CTS* place. Other stations except *n* increase the current simulation time plus a time-out value (function *toutcts*), so they will not be able to transmit further messages until this time-out elapses. Transition *End* fires when the number in the *Stations* place becomes 0, thus resetting this number to its initial value and starting the message transmission for station *n*. For this purpose, one token is written into the *Sending_Message* place, indicating that this station is sending a message, with a timestamp equal to the time at which this transmission finishes. Another token is writ-

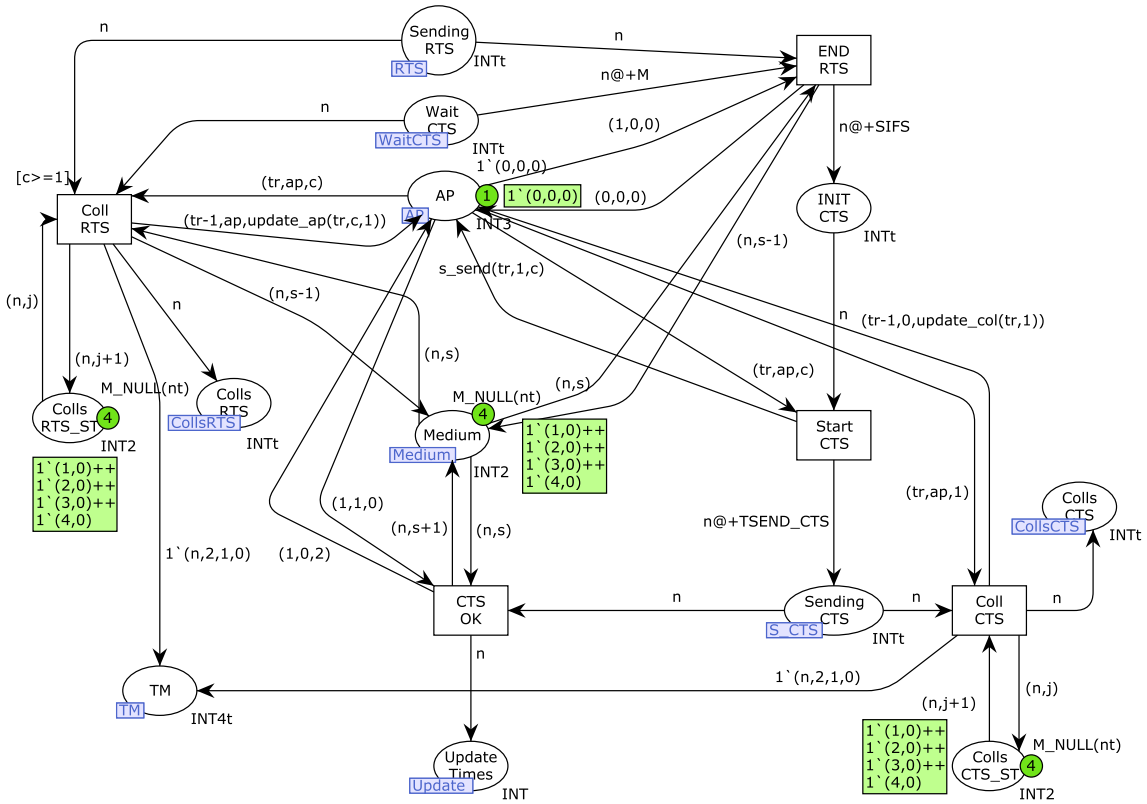
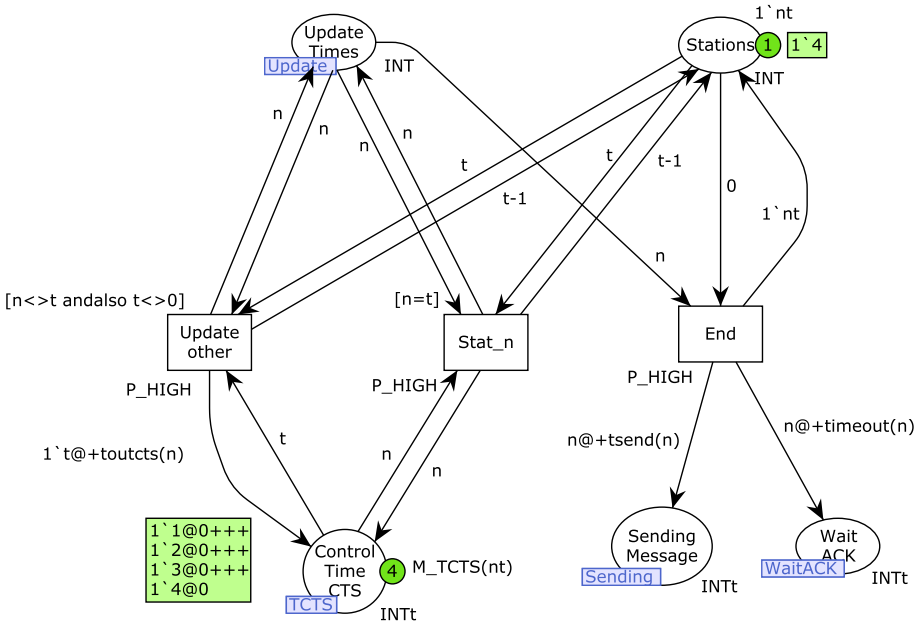


Fig. 10 RTS_CTS CPN page

Fig. 11 UpdateTimes CPN page



ten into the *Wait_ACK* place to activate a time-out for the reception of the ACK message. Notice the high priority of these transitions to enforce their firing before any other station could start a transmission at this same time. Thus, once *CTS_OK* is fired, the time-outs are immediately updated and other stations will be forced to wait.

3.2.8 SendMessage CPN page

In this CPN page, shown in Fig. 12, we have again several fusion places. Places *AP*, *Medium*, *Sending_Message* and *Wait_ACK* are the places with the same names in Fig. 10, *TM* is the same as in Fig. 6, *Backoff_Counter* is the same

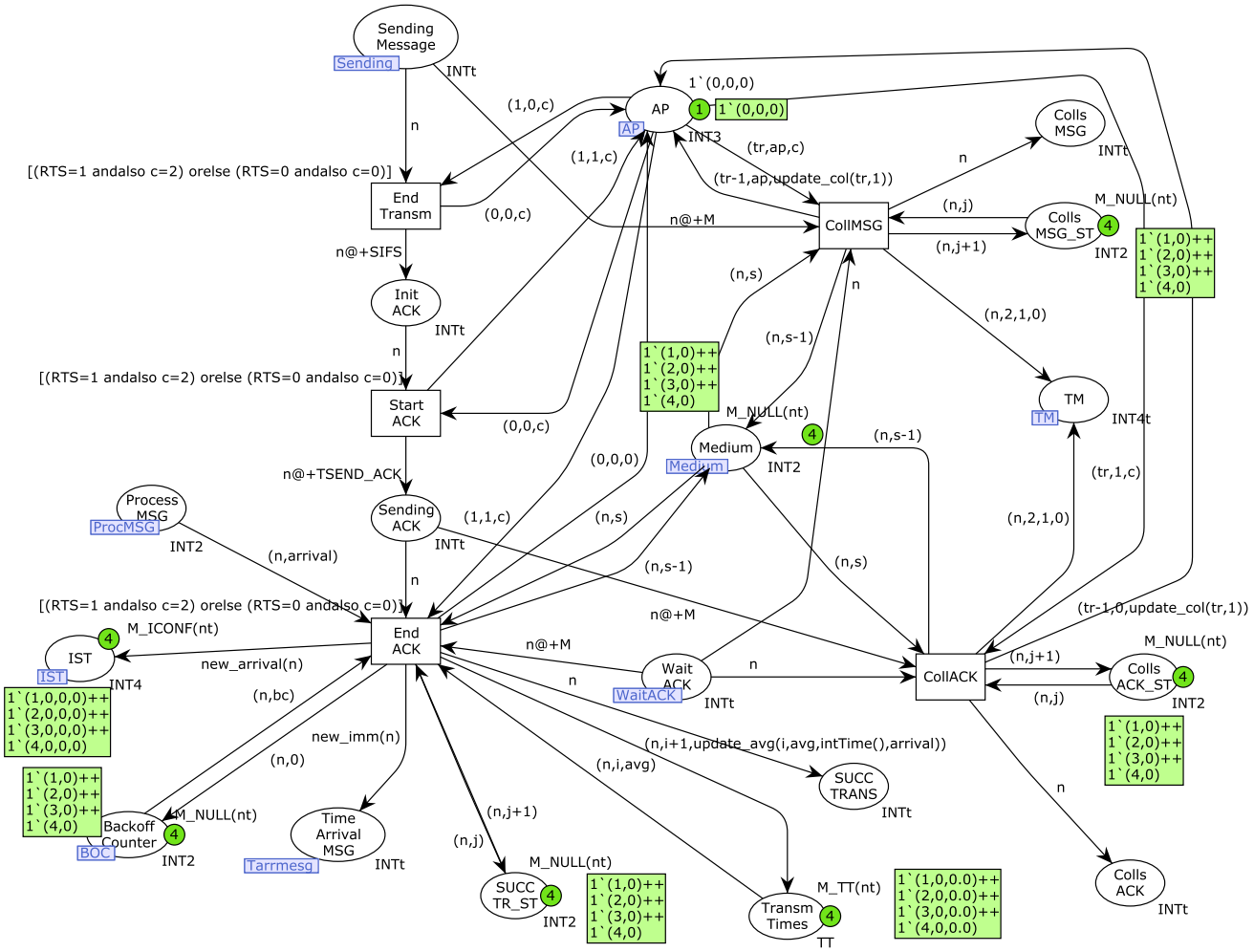


Fig. 12 SendMessage CPN page

place as in Fig. 8, and *Process_MSG*, Time_Arrival_MSG and *IST* are the same places as in Fig. 5.

After a successful transmission transition, *End_Transm* fires. Transition *Start_ACK* then fires after a SIFS delay to start the transmission of an ACK message. In this case, place *Sending_ACK* becomes marked with one token, indicating the station number, with a timestamp equal to the time at which this transmission finishes. In the event of a message collision, transition *CollMSG* fires, updating a counter in the *Colls_MSG_ST* place and writing one new token in the *Colls_MSG* place. In this case, a new token is also written into the *TM* place to retry the transmission, after an AIFS period waiting for the medium to be idle. After this period, the station will start a Backoff period.

Transition *End_ACK* fires when the ACK transmission is correct, whereas transition *CollACK* would fire in the event of an ACK collision, which is a very unlikely event, and which depends on the model parameters. When transition *End_ACK* fires, it means that the message transmission has been successfully performed. Therefore, a new token is written into the place *SUCC_TRANS*, indicating the station number, and with the timestamp at which the transmission terminated. A counter is also updated in the *SUCC_TR_ST* place. In addition, the average transmission time of this station is updated in the place *Trans_Times*. Finally, note that after a successful transmission, the Backoff counter of the station is set to 0, and a new message to be transmitted is produced in place *Time_Arrival_Message* in the case of a station of type-1, i.e., in a saturated state (function *new_imm*). A token is also written in the *IST* place to allow the station to start a new transmission (function *new_arrival*).

4 Model correctness validation

Validation has been done by defining a set of test cases, which are demonstrated by executing and testing the CPN model under the scenarios considered for them. We have identified 18 test cases to validate the behavior of the CPN model. These test cases check the protocol behavior, so they have been

selected to verify that the CPN model behavior conforms the 802.11e wireless protocol, as presented in Sect. 2.1. Each test case is checked by considering several scenarios that cover the test case. Table 3 shows the test cases identified, indicating the CPN pages related with the tests, and thus checked using the specific scenarios associated, which are described in detail in Appendix B.

Most of these test cases can be checked just configuring the CPN model as indicated in the scenarios and running the model using the simulation engine of CPN Tools, but three monitors have also been required for this purpose. Monitors are a mechanism provided by CPN Tools to observe, modify or control a simulation in a CPN model. There are several types of monitor, among them, we have *break point monitors* and *place contents* monitors. Break point monitors are used to stop a simulation when a certain condition is fulfilled, for instance, a transition has been fired or a place reaches a certain marking. In contrast, a place contents monitor stops a simulation when a place is either marked (at least one token in the place) or empty (no token in the place), as indicated in the definition of the monitor.

Listing 3 shows the monitors defined for this purpose. The first one, *BP_NoSevTransmCTSrec* is a break point monitor defined for test case 11 (there cannot be two stations transmitting when CTS was received). This monitor would stop the simulation if the *AP* place reaches the marking 1‘(2, 0, 2) (line 7 of Listing 3), which indicates that two stations are transmitting and CTS was received, so the test in this case consists in checking that simulations are never stopped by the monitor. Monitor *PC_With_RTS_No_CollsMSG* is a place contents monitor, which checks that place *Colls_MSG* is never marked (line 11). This monitor is used to check test case 17 (no message collision when using RTS/CTS), so the test consists in checking that simulations are never stopped by the monitor when variable RTS is set to 1. Notice that message collisions refer to collisions when the message is being sent (after the RTS/CTS exchange). Obviously, we can have RTS and CTS collisions in this case. Finally, monitor *PC_no_ACK_Coll* is also a place contents monitor, which is used to check test case 18 (no ACK collision when using RTS/CTS), so it would stop the simulation in the event of an ACK collision. None of these monitors stopped the simulations throughout the proofs.

Listing 3 Validation break point monitors.

```
1  BP_NoSevTransmCTSrec
2  Type: Break point
3  Nodes ordered by pages
4  AP (place)
5  Predicate
6  fun pred (StartfromBO'AP_1_mark : INT3 ms) =
7  (StartfromBO'AP_1_mark == 1‘(2,0,2));
8
9  PC_With_RTS_No_CollsMSG
10 Type Place Content break point
11 Not If is empty
12 Nodes ordered by pages
13 SendMessage
14 Colls_MSG (place)
15
16 PC_No_ACK_Coll
17 Type Place Content break point
18 Not If is empty
19 Nodes ordered by pages
20 SendMessage
21 Colls_ACK (place)
```

Listing 4 Write-in-file monitor and break point monitor used for performance evaluation.

```
1  Max Time
2  Type: Break point
3  Nodes ordered by pages
4  Predicate
5  fun pred() = IntInf.toInt(time()) > 15000000
6
7  TransmisionT
8  Type: Writein file
9  File extension : txt
10 Nodes ordered by pages
11 SendMessage
12 TransmTimes (place)
13 Init
14 fun init (SendMessage'Transm_Times_1_mark : TT ms) = ""
15 Predicate
16 fun pred (SendMessage'Transm_Times_1_mark : TT ms) = true
17 Observer
18 fun obs (SendMessage'Transm_Times_1_mark : TT ms) = ""
19 Stop
20 fun stop (SendMessage'TransmTimes_1_mark : TT ms) =
21 "Final marking of place Transm Times"^TT.mkstr_ms(SendMessage'
       TransmTimes_1_mark)
```

5 Evaluation

The evaluation presented in this section has two goals: (1) to show the capabilities of CPNs as a performance evaluation and protocol analysis tool; and (2) to evaluate the effectiveness and/or shortcomings of the protocol mechanisms being introduced in the IEEE 802.11 standard.

With regard to the first goal, we have used the simulation and monitoring features of CPN Tools to collect the performance indexes of interest, as the number of successful transmissions, the times for these transmissions, the number of lost packets, etc. As said before, monitors allow us to observe, control and modify a simulation in CPN Tools. In particular, we now use a break point monitor to stop the simulation at the maximum simulation time (15 seconds), several count transition occurrence data collector monitors, which allow us to count the firings of the specified transitions, and two write-in file monitors, which allow us to gather the final markings of the specified places of the CPN, by writing them into text files.

Table 3 Test performed in the CPN model validation

#	CPN page	Test case	Scenarios
1	ProdMessages	Stations are correctly configured in their initialization	1
2	Start_station	Change of medium state is correctly detected by stations and the AIFS delay is applied	2–4
3	StartSend	Either StartSend or RTS_IMM is fired depending on RTS variable, and medium state is correctly updated	5–8
4	Backoff	Backoff is correctly assigned when CW limit not reached	9
5	Backoff	Backoff not assigned and frame dropped when CW limit reached	10
6	StartfromBO	Backoff counter is correctly decremented every *SlotTime* time units when no CTS message has been received	11
7	StartfromBO	When backoff time becomes 0, either *Start_SendBO* or *RTS_ABO* is fired, depending on the RTS variable, and the medium state is correctly updated	12–15
8	StartfromBO	Backoff time cannot be decremented when CTS received until CTS time-out elapses	16
9	RTS_CTS	Collision of *RTS* when medium busy and one station starts RTS transmission	17–18
10	RTS_CTS	Start CTS transmission when RTS sent correctly	19
11	RTS_CTS	There cannot be two stations transmitting when CTS was received	20
12	RTS_CTS	End of CTS transmission if no collision	21
13	RTS_CTS	CTS Collision when a station is sending RTS and CTS being sent for another station	22
14	UpdateTimes	Time-outs updated in place *Control_Time_CTS* when CTS sent	23
15	SendMessage	ACK sent after SIFS time units upon termination of transmission, times and counters updated, medium free and new frame can be sent	24–27
16	SendMessage	Message collision when two stations are transmitting simultaneously (not using RTS/CTS)	28
17	SendMessage	No message collision when using RTS/CTS with the parameters considered	29
18	SendMessage	No ACK collision occurs when using RTS/CTS with the parameters considered	30

Then, we have used the CPN Tools simulator engine to carry out the evaluation of a wireless network consisting of one AP and two or more wireless nodes. For each metric of interest, we have obtained the mean values, typical deviations and confidence intervals by repeating 100 trials of each network setup (scenario). The trials have been automatically launched using the replication function of the CPN Tools software. We have defined a break point monitor to stop the simulation process at the maximum simulation time whose value has been set to 15 seconds, and four count transition occurrence data collector monitors to collect the number of RTS, CTS, ACK and message collisions. These monitors allow us to count the number of times that the corresponding transitions have been fired. We have also implemented one write-in-file monitor to save in a text file the final marking of place *Trans_Times*. This marking provides us with the average station transmission times at the end of each simulation. Finally, a place *MaxCols* has been included to count the maximum number of consecutive collisions before a successful transmission, and a write-in file monitor was then defined to collect this information.

Listing 4 shows the code of two of the monitors implemented in our study. Monitor *Max Time* is the break point monitor to stop the simulation when the simulation time is greater than or equal to 15 seconds, while *TransmissionT* collects the final marking of place *Trans_Times*.

As for the second goal, the evaluation of the effectiveness and shortcomings of the protocol mechanisms being introduced in the IEEE 802.11 standard, we have defined three case studies. The first one sets the basis of our study while the two other cases are developed to further explore the effectiveness of the IEEE 802.11 mechanisms and their potential use in a multimedia scenario.

The first case study explores the performance of the hidden-node mitigation and priority protocol mechanisms under saturation conditions: a setup frequently used to determine the bounds of the capacity allocated to different traffic categories [33]. Under these conditions, we should be able to evaluate the two main performance metrics of interest provided by the priority and hidden-node mitigation mechanisms: throughput and losses. We evaluate individually the performance of the various traffic categories, i.e., a case when all the network nodes belong to the same traffic type, and a

scenario supporting the highest (VO) and lowest (BK) type. The latter aims at exploring the effectiveness of the priority mechanism in the absence of feedback channel conditions, i.e., it emulates a setup as the one described in [26].

The second and third study cases explore scenarios supporting two types of traffic under different network load conditions. The scenarios of these two latter case studies have been defined taking into account the results obtained in the first part. They provide further insights on the choices to be made when setting the parameters of multi-priority scenarios in the presence of hidden nodes. The second scenario focuses on exploring the effectiveness of the priority and hidden-node mitigation mechanisms under hidden-node and non-saturated conditions: it provides further insight on the bandwidth allocated and losses to each priority as a function of the network load.

Regarding the third scenario, our goal is to explore the use of the BE priority as an alternative solution to support voice services. Accordingly, we first set up a scenario consisting of multiple nodes making use of the VO priority setting. We then make use of the BE service to explore the service received by the sources operating under the same conditions as the ones in the previous setting. The main goal is therefore to explore the use of a less-aggressive backoff mechanism, the BE, as a manner to enhance the multiplexing gain at the expense of degradation to the service provided to the VO sources. We also consider two different packet sizes. This latter parameter plays an important role in time-constrained applications, such as VO services.

5.1 Case study I: saturated traffic conditions

This case study allows us to explore the effectiveness and shortcomings of the hidden-node mitigation method and the QoS mechanisms under extreme conditions. The network setup consists of an AP and two stations operating under saturated conditions. All stations can sense the activity of the AP, but they cannot sense the activity of each other due to the distance between them.

Table 4 shows the scenarios considered in this case study, which are classified depending on the use of RTS/CTS or not. In these experiments, we have considered two packet sizes (100 bytes and 1500 bytes), regardless of the traffic priority, since the time wasted by collisions severely depends on the frame transmission time. Moreover, since the length of the Backoff mechanism depends on the traffic priority, each scenario analyzes a single traffic type, except for scenarios 1, 6, 11 and 16. These last scenarios, consisting of one BK and one VO stations, aim at evaluating the effectiveness of the priority mechanism. Notice that the [S] label next to the number of stations indicates the delivery of saturated traffic.

Table 5 shows the average performance results for the scenarios when the RTS/CTS mechanism is disabled, in which we report the total number of packets transmitted, the number of packet transmissions per traffic type, the number of packet losses, the network throughput, the number of collisions and the maximum length of consecutive collisions (collision chain). Notice that when the RTS/CTS handshaking is disabled, the number of collisions refer to the collisions suffered by the data and ACK packets. Table 6 shows the standard deviations and confidence intervals for the main metrics shown in Table 5. From the values of this second table we can observe that the values obtained are a good estimation of the protocol's behavior.

As seen from Table 5, scenarios 1–5, the number of packets successfully transmitted is substantially greater than in scenarios 6–10. We remind the reader that the packet length in the first five scenarios is set to 100 bytes, while in scenarios 6–10 it is set to 1500 bytes. From the results, it is clear that the use of short packets should be preferred in the absence of the RTS/CTS mechanism. Regarding the performance of scenarios 2–5, we notice that the highest priority traffic, i.e., VO, reports the worst results in terms of packets successfully transmitted. The highest throughput is reported for the BE traffic (scenario 2). These results can be explained by the fact that as the priority decreases, a longer Backoff period allows for the competing nodes to successfully solve the channel access conflict. As can be observed, the number of collisions reported for VO traffic is significantly higher than for the BK and BE services. However, it should be noted that the number of successfully transmitted packets of BK traffic is lower than the one reported for the BE traffic (scenarios 2 and 3). The number of lost packets and collisions for these two scenarios are very similar. These results clearly show that the highest priority services, i.e., VO and VI, are unable to solve channel access conflicts when the distance prevents them from sensing the ongoing transmission.

Scenario 1 explores the case when a station of the lowest priority, BK, competes against a VO node. As can be seen, under saturation conditions, the number of successfully delivered BK packets represent less than 1% of the total. This result clearly shows that the high-priority traffic is able to take full advantage of its privilege position. The low priority traffic is heavily penalized in the presence of hidden high-priority nodes.

When long packet sizes are used (Scenarios 6–10), the performance of the IEEE 802.11e standard is heavily penalized. Under these scenarios, the highest priority services are unable to solve channel access conflicts, since the packet transmission time is longer than the Backoff mechanism. As depicted in Fig. 13, when the stations are unable to detect the activity of each other, they transmit their data frames as soon as their corresponding Backoff periods expire. Then they wait for the corresponding ACK frames. As soon as the ACK timeout expires, they attempt to transmit once again their frames following a second Backoff period. However, depending on the

Table 4 Evaluation scenarios for *Case Study I*

Scenario (#)		Packet size (bytes)	Stations distribution (# STAs)			
Without RTS/CTS	With RTS/CTS		BK	BE	VI	VO
1	11	100	1 [S]	–	–	1 [S]
2	12		2 [S]	–	–	–
3	13		–	2 [S]	–	–
4	14		–	–	2 [S]	–
5	15		–	–	–	2 [S]
6	16	1500	1 [S]	–	–	1 [S]
7	17		2 [S]	–	–	–
8	18		–	2 [S]	-	–
9	19		–	–	2 [S]	–
10	20		–	–	–	2 [S]

Table 5 Performance results—*Case Study I* without RTS/CTS

Sc (#)	$CorrectPkts$(#)	Correct Pkts per Station (#)				Lost Pkts (#)	Th. (Kbps)	Collisions (#)		Max colls chain (#)
		BK	BE	VI	VO			Pkts	ACK	
1	7962	72	–	–	7890	2132	414.69	6739	59	20
2	6652	3326	–	–	–	462	346.46	5127	74	13
3	6898	-	3449	–	-	461	359.27	5137	74	13
4	796	-	-	398	-	8283	41.46	16669	52	182
5	770	-	-	-	385	8984	40.10	18035	50	220
6	142	1	-	-	141	1217	110.94	3484	1	291
7	206	103	-	-	-	408	160.94	2876	1	82
8	206	-	103	-	-	411	160.94	2892	1	84
9	0	-	-	0	-	2067	0.00	4137	0	4137
10	0	-	-	-	0	2111	0.00	4226	0	4226

Table 6 Standard deviations and confidence intervals—*Case Study I* without RTS/CTS

Sc (#)	Correct Pkts				Lost Pkts				Collisions			
	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%
1	7962	668	111	176	2132	372	61	96	6739	919	153	243
2	6652	640	107	169	462	27	4	7	5127	382	64	101
3	6898	483	80	128	461	25	4	6	5137	346	58	91
4	796	138	23	37	8283	445	73	115	16669	847	141	224
5	770	161	27	42	8984	85	80	125	18035	914	152	242
6	142	52	9	14	1217	46	8	12	3484	127	21	34
7	206	21	4	6	408	30	5	8	2876	101	17	27
8	206	17	3	5	411	29	5	7	2892	92	15	24
9	0	0	0	0	2067	76	12	19	4137	151	25	40
10	0	0	0	0	2111	25	4	7	4226	50	8	13

timing relation between the data frame transmission time and the Backoff time, their frames may once again collide. This process may repeat up to the number of attempts defined by the standard for each traffic class, see Table 1. As seen from the results of scenarios 9 and 10, the VO and VI services will be severely affected by the presence of hidden nodes when the use of long frames is preferred. For these two services, all the collisions involve data frames. In the case of the lowest priorities, which make use of a longer Backoff period, collisions may also involve ACK packets. From the above results,

Table 7 Performance results—*Case Study I* with RTS/CTS

Sc (#)	Correct Pkts(#)	Correct Pkts per Station (#)				Lost Pkts (#)	Th. (Kbps)	Collisions (#)				Max coll chain (#)
		BK	BE	VI	VO			Pkts	ACK	RTS	CTS	
11	8185	64	–	–	8121	865	426.30	0	0	3205	60	13
12	6542	3271	–	–	–	237	340.73	0	0	3463	133	10
13	6720	–	3360	–	–	242	350.00	0	0	3421	118	10
14	2550	–	–	1275	-	8411	132.81	0	0	17558	236	48
15	2414	–	–	–	1207	10147	125.73	0	0	20605	132	64
16	2001	16	–	–	1985	209	1563.28	0	0	799	19	11
17	1900	950	–	–	–	68	1484.38	0	0	1007	37	9
18	1914	–	957	–	–	68	1495.31	0	0	958	31	9
19	1268	–	–	634	–	4665	990.63	0	0	9688	130	52
20	1150	–	–	–	575	6262	898.44	0	0	12574	95	81

Table 8 Standard deviations and confidence intervals—*Case Study I* with RTS/CTS

Sc (#)	Correct Pkts				Lost Pkts				Collisions			
	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%
11	8185	156	27	43	865	135	24	37	3205	418	74	116
12	6542	38	6	10	237	15	3	4	3463	226	38	51
13	6720	52	9	14	242	17	3	5	3421	303	51	80
14	2550	30	7	10	8411	46	10	15	17558	71	15	24
15	2414	30	20	32	10147	35	24	37	20605	78	52	82
16	2001	134	22	35	209	31	5	8	799	192	32	50
17	1900	5	1	1	68	5	1	1	1007	68	11	17
18	1914	5	1	1	68	5	1	1	958	71	12	18
19	1268	49	8	13	4665	571	94	147	9688	1080	178	278
20	1150	107	17	27	6262	1041	171	268	12574	2038	335	525

it is clear that the use of shorter packets provide much better results. This is simply explained by the fact that when a station is unable to detect the activity of the others, the length of the collision period is equal to the packet transmission time, and thus a longer packet results in a longer collision period. The number of consecutive collisions reported confirms that the ratio between the packet transmission time and the length of the Backoff period plays a major role on solving channel access conflicts. In the case of scenarios 9 and 10, the VI and VO traffic are unable to transmit a single packet, the length of the collision chain is equal to the number of collisions involving data packets. The best results are obtained for short packets and a longer Backoff period (see Table 5). Therefore, the use of RTS/CTS control frames should help to reduce the time to solve conflicts.

Table 7 shows the results for the previous scenarios when the RTS/CTS handshake mechanism is implemented. Table 8 shows the corresponding standard deviations and confidence intervals. Similarly to scenarios 2–5, in scenarios 12–15, the lowest number of transmitted packets is reported for the VO traffic. The best results for a single priority scenario are reported for the BE traffic. A slight decrease in this metric is reported for the BK traffic, confirming that the length of the Backoff period results in a slight bandwidth waste, i.e., idle slots. In scenario 11, BK traffic is again heavily penalized by the presence of the VO node. Although the number of transmitted packets is lower when using long packets (scenarios 16–20) rather than when setting a shorter length, the throughput in bits per second is considerable higher. Take for instance scenarios 15 and 20. In scenario 15, 2414 packets of 100 bytes are transmitted, resulting in a throughput of 125.73 Kbps. By contrast, the throughput for scenario 20, where 1500 bytes packets are used, is 898.44 Kbps. Therefore, the use of RTS/CTS proves being more effective when reserving the channel to transmit long packets. This is an important issue to be considered when setting the network parameters. However, it should be noted that some services,

Table 9 Evaluation scenarios for *Case Study II*

Scenario (#)		Packet Size (Bytes)	Network load (kbps)	Stations Distribution (#)			
Without RTS/CTS	With RTS/CTS			BK	BE	VI	VO
21	31	100	400	1 [I]	–	–	1 [I]
22	32		500	1 [I]	–	–	1 [I]
23	33		600	1 [I]	–	–	1 [I]
24	34		700	1 [I]	–	–	1 [I]
25	35		800	1 [I]	–	–	1 [I]
26	36	1500	400	1 [I]	–	–	1 [I]
27	37		500	1 [I]	–	–	1 [I]
28	38		600	1 [I]	–	–	1 [I]
29	39		700	1 [I]	–	–	1 [I]
30	40		800	1 [I]	–	–	1 [I]

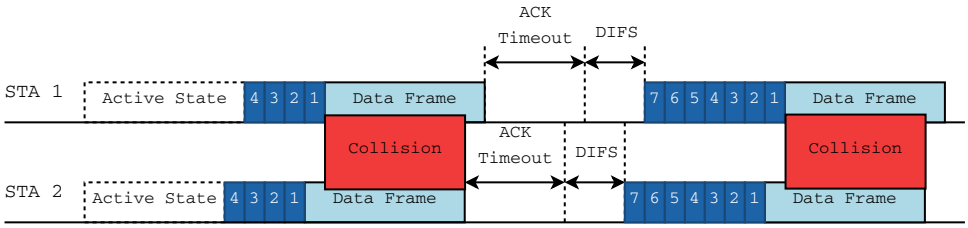


Fig. 13 Timing diagram of two VO sources operating under hidden-node conditions

such as VO traffic, impose heavy constraints in terms of the packet length to be used.

5.2 Case study II: non-saturated traffic conditions

The main goal of this case study is to explore the shortcomings of the priority mechanism defined by the IEEE 802.11e standard in conditions of intermittent traffic in the presence of hidden nodes. Since we are particularly interested on evaluating the priority setting defined by the standard, we consider a system consisting of one AP, one BK station and one VO station. The network load is varied from 400 to 800 Kbps, representing moderate network load conditions. Recall that the nominal network data rate is 2 Mbps. The network load has been set through the average packet inter-arrival time of each station using an exponential distribution. The scenarios considered for this case study are shown in Table 9, once again classified depending on the use of RTS/CTS. We consider again two packet lengths and the five aforementioned network load conditions. Notice that the [I] label next to the number of stations indicates the delivery of intermittent traffic (Tables 10, 11).

Tables 10 and 12 show the results for this case study when disabling/enabling the RTS/CTS mechanism, respectively. In addition, Tables 11 and 13 contain the corresponding standard deviations and confidence intervals for the main metrics. In the absence of RTS/CTS, Table 10 depicts that the priority mechanism can provide a better service for the VO traffic when a short packet is preferred (scenarios 21–25). In all of these scenarios, except for scenario 21, the number of transmitted VO packets is higher than the number of transmitted BK packets. Notice that in the case of scenario 21, the use of a shorter Backoff period plays against the VO traffic. This is due, as explained in the analysis of the first case study, to the ratio between the packet transmission time and the length of the Backoff period.

When a long packet is used (scenarios 26–29 in Table 10), the number of transmitted VO packets is significantly lower than the number of BK packets. Once again, this is mainly due to the lower number of transmission attempts before dropping a VO packet. The results in Table 12 prove the effectiveness of the RTS/CTS mechanism when short packets are used. As seen from the outcomes from scenarios 32–35, the number of VO packets is substantially higher than the number of BK packets. The same trend can be observed in scenario 31 with respect to scenario 21. Scenarios 36–40 demonstrate how the use of a long packet for the voice service should be avoided. As can be observed, the number of VO packets being served is higher than that for the scenarios 26–30 when RTS/CTS is not enabled. However, the number of VO packets transmitted is still lower than the number of BK packets. In other words, the priority mechanisms are not able to provide a better service to the high priority traffic, i.e., the VO packets.

Figure 14 shows the transmission times for BK and VO services for all the scenarios considered in Case Study II. As seen in Fig. 14a, c, the use of RTS/CTS reports a longer

Table 10 Performance results—*Case Study II* without RTS/CTS

Sc (#)	*Correct Pkts*(#)	Correct Pkts per Station (#)				Lost Pkts (#)	Th. (Kbps)	Collisions (#)			
		BK	BE	VI	VO			Pkts	ACK	RTS	CTS
21	5697	3137	–	–	2560	1268	296.72	7805	263	–	–
22	5587	2104	–	–	3483	1324	290.99	8135	190	–	–
23	6031	1451	–	–	4580	1218	314.11	7811	153	–	–
24	6417	917	–	–	5500	1268	334.22	7810	143	–	–
25	7017	545	–	–	6472	1282	365.47	7739	126	–	–
26	414	239	–	–	175	84	323.4	438	23	–	–
27	481	292	–	–	189	140	375.78	717	41	–	–
28	519	335	–	–	184	228	405.47	1114	49	–	–
29	544	377	–	–	167	327	425.00	1564	47	–	–
30	529	399	–	–	130	456	413.28	2125	39	–	–

Table 11 Standard deviations and confidence intervals—*Case Study II* without RTS/CTS

Sc (#)	Correct Pkts				Lost Pkts				Collisions			
	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%
21	5697	474	79	125	1268	251	51	65	7805	529	88	140
22	5587	676	113	179	1324	345	57	89	8135	804	134	212
23	6031	349	58	92	1218	241	40	62	7811	557	93	147
24	6417	579	96	153	1268	319	52	82	7810	750	125	198
25	7017	424	71	112	1282	291	48	75	7739	807	134	213
26	414	34	6	9	84	31	5	8	438	100	17	26
27	481	42	7	11	140	41	7	11	717	142	24	37
28	519	54	9	14	228	57	9	15	1114	197	33	52
29	544	61	10	16	327	68	11	18	1564	250	42	66
30	529	72	12	19	456	77	13	20	2125	309	52	82

Table 12 Performance results—*Case Study II* with RTS/CTS

Sc (#)	*Correct Pkts*(#)	Correct Pkts per Station (#)				Lost Pkts (#)	Th. (Kbps)	Collisions (#)			
		BK	BE	VI	VO			Pkts	ACK	RTS	CTS
31	6077	3144	–	–	2933	832	316.51	0	0	7191	116
32	6127	2201	–	–	3926	780	319.11	0	0	6889	112
33	6403	1459	–	–	4944	736	333.49	0	0	6377	107
34	6903	896	–	–	6007	627	359.53	0	0	5425	96
35	7497	464	–	–	7033	593	390.47	0	0	4700	84
36	493	250	–	–	243	5	385.16	0	0	42	0
37	613	313	–	–	300	10	478.91	0	0	80	1
38	731	376	–	–	355	17	571.09	0	0	133	2
39	847	434	–	–	413	26	661.72	0	0	206	3
40	958	500	–	–	458	38	748.44	0	0	300	4

transmission time for the BK traffic, while the transmission time of the VO traffic remains unchanged. A higher delay is observed for BK packets when RTS/CTS is enabled, due to the greater number of transmitted BK packets in this case. However, this extra delay should not have an impact on the QoS provided to the BK traffic, because the BK traffic is delay tolerant. Figure 14b, d shows the transmission times when a longer packet length is used. In this case, the use of RTS/CTS produces shorter packet transmission times for the

Table 13 Standard deviations and confidence intervals—*Case Study II* with RTS/CTS

Sc (#)	Correct Pkts				Lost Pkts				Collisions			
	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%
31	6077	71	12	19	832	70	12	18	7191	325	54	85
32	6127	71	12	19	780	66	11	17	6889	303	51	79
33	6403	88	15	23	736	134	22	35	6377	574	95	149
34	6903	78	13	20	627	67	11	18	5424	293	49	77
35	7497	118	20	31	593	110	19	29	4700	513	87	136
36	493	20	3	5	5	2	1	1	42	14	2	4
37	613	24	4	6	10	3	1	1	80	22	4	6
38	731	26	4	7	17	4	1	1	133	27	5	7
39	847	28	5	7	26	5	1	2	206	35	6	9
40	958	30	5	8	38	7	1	2	300	45	8	12

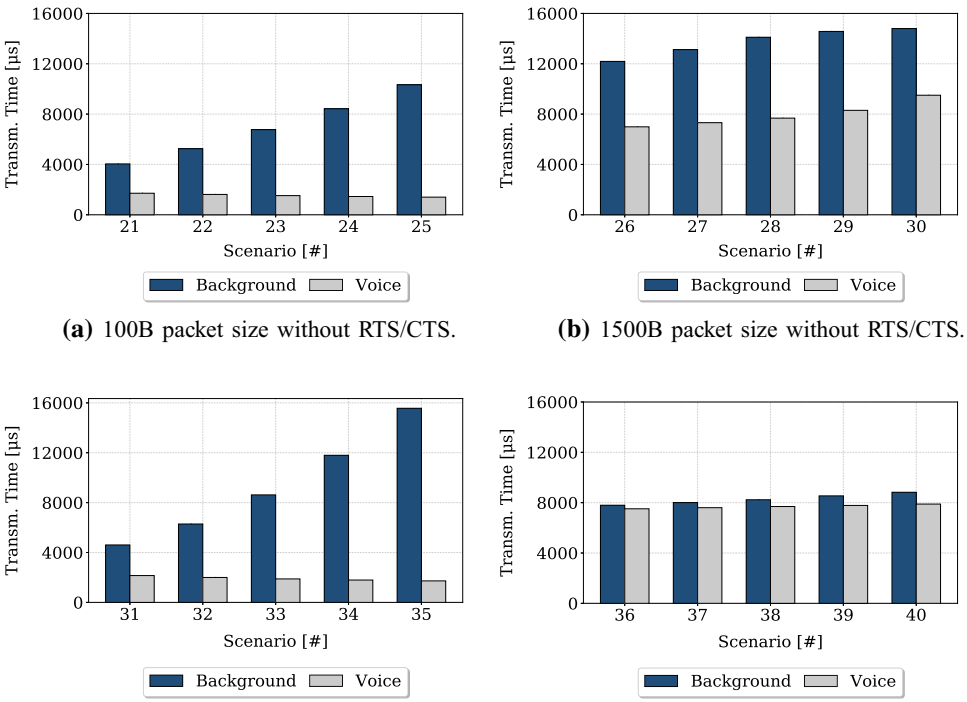


Fig. 14 Transmission time in *Case Study II* for different packet sizes and traffic types

(a) 100B packet size without RTS/CTS.

(b) 1500B packet size without RTS/CTS.

(c) 100B packet size with RTS/CTS.

(d) 1500B packet size with RTS/CTS.

BK traffic. Despite this improvement, the use of long VO packets should be avoided.

5.3 Case study III: multimedia scenario

This case study first presents a network setup consisting of one BK and several VO stations. The VO sources will then be replaced by BE nodes to analyze how in some specific cases we can profit from low traffic priorities to address some shortcomings of the IEEE 802.11 control mechanisms. Table 14 shows the scenarios considered, where in this case the RTS/CTS exchange is always enabled and stations only produce intermittent traffic. Scenarios 41–47 consist of one BK source and a varying number of VO sources. The BK node uses 500 bytes packets, while the VO packet length has been set to 50 bytes. This scenario then reflects a more realistic setup, where VO sources make use of short packets, while BK traffic is used for data traffic. The second part of the tables, i.e., scenarios 48–54, defines a setup consisting of one BK source and a varying number of BE nodes. Notice that BE nodes share the same system parameters than the VO sources in scenarios 41–47. This type of study has been explored in various network contexts with the aim of exploring the use of other network services, e.g., BE, to carry VO traffic.

Table 14 Evaluation scenarios for *Case Study III*

Sc (#)	Station Load (Kbps)				Stations Distribution (# STAs)				Packet Size per Station (Bytes)			
	BK	BE	VI	VO	BK	BE	VI	VO	BK	BE	VI	VO
41	640	–	-	64	1 [I]	–	–	1 [I]	500	–	–	50
42		–	–		1 [I]	–	–	2 [I]		–	–	
43		–	–		1 [I]	–	–	3 [I]		–	–	
44		–	–		1 [I]	–	–	4 [I]		–	–	
45		–	–		1 [I]	–	–	5 [I]		–	–	
46		–	–		1 [I]	–	–	6 [I]		–	–	
47		–	–		1 [I]	–	–	7 [I]		–	–	
48		64	–	–	1 [I]	1 [I]	–	–		50	–	–
49			–	–	1 [I]	2 [I]	–	–			–	–
50			–	–	1 [I]	3 [I]	–	–			–	–
51			–	–	1 [I]	4 [I]	–	–			–	–
52			–	–	1 [I]	5 [I]	–	–			–	–
53			–	–	1 [I]	6 [I]	–	–			–	–
54			–	–	1 [I]	7 [I]	–	–			–	–

Table 15 Performance results—*Case Study III* with RTS/CTS

Sc (#)	*Correct Pkts* (#)	Correct Pkts per Station (#)				Lost Pkts (#)	Collisions (#)			
		BK	BE	VI	VO		Pkts	ACK	RTS	CTS
41	4132	2387	–	–	1745	651	0	0	5381	73
42	3712	1386	–	–	1163	2517	0	0	11943	70
43	3249	726	–	–	841	4860	0	0	18466	50
44	2606	386	–	–	555	7692	0	0	26481	38
45	1874	199	–	–	335	10752	0	0	35349	28
46	1267	103	–	–	194	13716	0	0	43956	21
47	803	54	–	–	107	16567	0	0	52337	15
48	4760	2388	2372	–	–	25	0	0	3228	81
49	6039	1935	2052	–	–	424	0	0	5192	136
50	5527	1297	1410	–	–	846	0	0	8989	184
51	4755	907	952	–	–	1364	0	0	13217	210
52	4004	629	675	–	–	1966	0	0	17660	213
53	3277	445	472	–	–	2641	0	0	22309	207
54	2638	314	332	–	–	3370	0	0	27109	190

Table 15 shows the outcomes of Case Study III, and the corresponding standard deviations and confidence intervals are shown in Table 16. From the results obtained for scenarios 41–47, it is clear that as the number of VO sources operating under hidden-node conditions increases, the performance of both traffic types, i.e., BK and VO, heavily degrades. We also notice that the service received by the VO traffic improves with respect to the one received by the BK traffic. However, the overall performance of the service is useless for the purpose of providing the required QoS guarantees. In scenarios 48–54, the performance offered by BE nodes is considerably higher than the one reported by the VO service in scenarios 41–47. From these results, taking into account that both traffic types, i.e., VO and BE, share the same traffic characteristics in terms of packet size and bitrate, we can conclude that relaxing the Backoff procedure of the VO under hidden-node conditions may prove beneficial. However, such approach proves useful only for a very limited number of nodes operating in the presence of hidden nodes.

Figure 15 shows the transmission delay for the two scenarios. This metric is a key performance index for voice communications. As seen from the figure, the delay reported for BE is higher than the one reported for VO. Despite this increase, the delay experienced by BE traffic is within the QoS requirements of VO traffic. Therefore, further tuning of the Backoff period may be worth to explore in order to accommodate a larger number of VO sources.

Table 16 Standard deviations and confidence intervals—*Case Study III* with RTS/CTS

Sc (#)	Correct Pkts				Lost Pkts				Collisions			
	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%	Avg	SD	Conf 90%	Conf 99%
41	4132	50	8	12	651	35	6	9	5381	279	48	75
42	3712	53	9	14	2517	74	13	20	11943	281	48	75
43	3249	91	16	25	4860	165	29	45	18466	637	112	175
44	2606	105	18	28	7692	181	31	48	26481	659	112	175
45	1874	95	17	26	10752	181	32	51	35349	719	128	201
46	1267	63	11	17	13716	146	25	39	43956	470	80	125
47	803	34	6	9	16567	124	20	32	52337	368	61	95
48	4760	67	12	18	25	5	1	2	3228	87	15	23
49	6039	90	15	24	424	11	2	3	5192	135	23	35
50	5527	86	14	22	846	25	4	7	8989	232	38	60
51	4755	81	13	21	1364	35	6	9	13217	295	49	76
52	4004	68	11	18	1966	25	4	7	17660	199	33	51
53	3277	58	10	15	2641	32	5	8	22309	226	37	58
54	2638	63	10	16	3370	49	9	13	27109	311	51	80

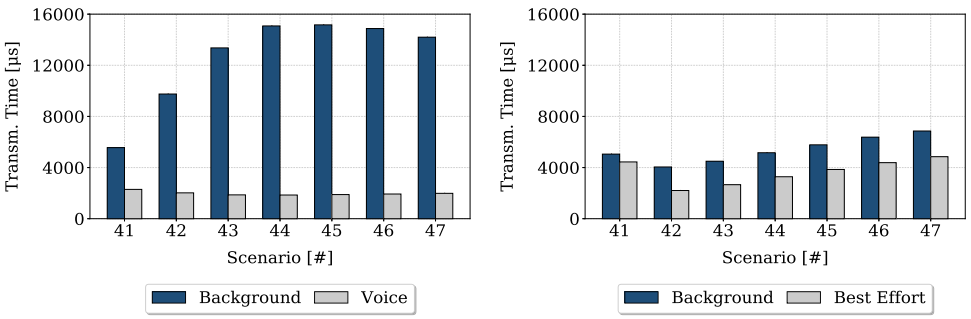


Fig. 15 Transmission time in *Case Study III* for different packet sizes and traffic types

(a) Background Vs. voice traffic.

(b) Background Vs. best effort traffic.

Note: The CPN model and the results obtained for the scenarios considered can be found at the URL http://dx.doi.org/10.17632/d6csk9yvry.3.

6 Related work

Petri Nets have been shown to be very useful in modeling and evaluating communication protocols, and in particular the so-called *High Level Petri Nets* (HLPNs), which extend the basic model by using hierarchic constructions and data/-time extensions, as is the case of CPNs. Thus, we can find many works that use different extensions of PNs to model communication protocols. In the area of wired protocols we can mention the work of Cambronero et. al. [34], in which CPNs and CPN Tools have also been used for the modeling of the 1-wire protocol, which is typically used to communicate a master device with some small inexpensive devices, such as digital thermometers and weather instruments, using low data rates and long ranges over a single shared bus.

With regard to wireless protocols, Hu et al. [35] have also developed a CPN model of the IEEE 802.11 DCF protocol. In particular, their work focuses on the modeling of the original DCF mechanism, including the use of the RTS/CTS exchange in order to address the hidden-node problem. That is to say, since their work does not cover the priority mechanisms introduced in the IEEE 802.11e amendment, the authors focus on evaluating the potential benefits of using RTS/CTS to tackle the hidden station problem. Similarly to our approach, they assume an error-less channel in which all losses are caused by channel access conflicts.

Hu and Jiao [36] present a CPN model of the IEEE 802.15.6 MAC protocol. Similarly to our approach, their aim is to show the benefits of using CPNs in the modeling of the various features of a given protocol. Their model is developed by using hierarchical and symmetrical modeling techniques. However, they leave the tuning and parameter adjustments of the protocol as issues to be considered in the future. That is to say, they do not look into the impact that the various param-

eters may have on the performance of the protocol under different scenarios.

Somappa and Simonsen [37] used a model-based development technique to generate the code of a medium access protocol derived from a CPN model. Their main goal was to show the benefits of deriving the actual code of an operational MAC protocol fully complying with the specifications and expected performance requirements.

Alves and Margy [38] have introduced a novel behavioral approach to the beacon-enabled IEEE 802.15.4 standard using CPNs. Similarly to our work, they test different values of the system parameters of the beacon-enabled MAC protocol of the 802.15.4 standard. They provide numerical results and a comparative analysis with results obtained from an experimental setup.

There are also many works using other extensions of Petri nets. For instance, Perez et al. [26] use the Mobius tool, which is based on Stochastic Petri Nets (SPNs). SPNs extend the basic model by considering that transitions fire after a probabilistic delay determined by a random variable (usually a negative exponential). They focus on the tuning of the EDCA parameters. However, they do not take into account the use of the RTS/CTS mechanism as a major parameter having a big impact on the performance of IEEE 802.11, when facing hidden-node issues and high population (node) densities.

Heindl and German [39] have also used SPNs for the evaluation of the IEEE 802.11 protocol. They propose a detailed model and two compact models of this protocol to analyze the distributed coordination function, which is the fundamental contention-based access mechanism. These models are used to investigate different physical layer options and the influence of several system parameters. These same authors extended this work in [40], where they claim that although most of the performance studies based on SPNs must adopt simplified assumptions (usually not proven to be accurate enough) because of both the expressive and simulating limitations associated to this formalism, they have developed an SPN that captures the main relevant aspects of the system to be modeled. In addition, they use simulation techniques to show a quantification of the influence of features such as Backoff time, Extended Interframe Spaces and Timing synchronization function. Even more, they claim to have identified the conditions under which simplifying parts of the model can be done with the aim of generating a model compact enough to be properly simulated and analyzed. This model, therefore, has the intrinsic limitations derived from the SPN model. In contrast, we propose a richer, flexible and scalable parameterized model based on CPNs, in which the RTS/CTS handshake mechanism can be enabled or disabled, and which also includes four types of prioritized traffic.

Moraes et al. [41] have also proposed a SPN model for the simulation and analysis of the IEEE 802.11e EDCA communication protocol, with the goal to analyze the behavior of the EDCA function associated to Quality of Service (QoS) stations. Specifically, they use Stochastic Activity Networks (SANs) to produce a compact and efficient model, which is then analyzed using the Möbius tool. The model presented in [41], therefore, lacks of some features that we have included in our CPN model, such as four types of prioritized stations, configurable use of RTS/CTS and scalability.

El Masri et al. [24] proposed a pattern-based Markov chain model of the IEEE 802.11e EDCA communication protocol with four types of traffic and including the virtual collision phenomenon. The model is divided into three basic patterns (similar to our CPN pages), and they define a separate Markov chain for each basic pattern and the global model which connects all of them. Performance analysis is then made using the Markov chain at the steady state, thus producing a set of formulas with the performance indicators, such as throughput, probability to access the medium and probability of medium being busy. It is therefore a rigid model, in which the transition probabilities must be set up to analyze the system behavior.

Zairi et al. [42] used HLPNs for formal modeling and analysis of Wireless Sensor Networks (WSN). Specifically, they proposed a model which represents the behavior of each WSN node. The modeled behaviors include the application, the protocols, an abstraction of the hardware energy consumption model and the environment, as viewed by the nodes. The authors claim that the proposed model is symmetrical since the considered WSNs are homogeneous, where all the nodes follow the same behavior. According to the authors, the main advantage of the model is that it is symmetrical and has a modular structure, so they consider that combining modular verification and symbolic reachability graphs will reduce the problem of the state space explosion and reduce the effort required to check properties. In a following paper [43], they completed the model by focusing on energy aspects. Then, they focused on the radio interface, which is controlled by the MAC component, since they consider it consumes the most energy. Then, the proposed model is used to evaluate the lifetime of the network and to estimate energy consumption.

Mokdad et al. [44] have assessed the performance of a multiple-priority (QoS) MAC protocol using two performance evaluation tools, namely Stochastic Automata Networks (SANs) and CPNs. They justify the use of CPNs for evaluating the transmission process. According to the authors, SANs are not designed to assess the performance of the transmission process when considering certain relevant system parameters. In the second part of the study, they show the benefits of using CPNs to model and simulate different scenarios. They obtained numerical results for two scenarios, allowing the modeling and evaluation of the priority mechanisms of the MAC protocol under study.

7 Conclusions

In this paper we have developed a CPN model for evaluating the performance and, more importantly, the effectiveness of the IEEE 802.11e priority mechanism. We have numerically evaluated the performance under various system operating conditions. We have identified some shortcomings of the protocol mechanisms defined by the IEEE 802.11e amendment. Regarding the priority mechanism, we have found out that the proposed parameter setting has serious limitations when operating in the presence of hidden nodes. Our study is particularly relevant in the context of wireless networks based on the IEEE 802.11 standard. Following current trends in the deployment of wireless networks in rural areas, we have considered long-range wireless network scenarios.

One of the main benefits of the model presented is its flexibility. By just changing the configuration parameters, we can analyze a great variety of scenarios that jointly consider several aspects of an IEEE 802.11 network. In addition, as the CPN model is divided into separate pages, it could be easily modified in order to consider possible variations of the protocol or incorporate the new amendments that are introduced in the new versions of the standard. Thus, as future work we intend to further explore the performance and effectiveness of other mechanisms on the IEEE 802.11 standard, and in particular on the IEEE 802.11e amendment and the newest IEEE 802.11ax standard. The study of the latter is of particular interest since it extends the use of collision avoidance mechanisms, namely, carrier sensing threshold. Our future plans will focus on exploring the performance of a protocol architecture integrating both mechanisms, i.e., a dynamic RTS/CTS mechanism and the carrier sensing threshold. The selective use of RTS/CTS may be performed on the basis of network operating conditions: number of active stations and priorities, overall network load, variable number of hidden stations and the carrier sensing area. Regarding the changes required in order to consider a dynamic RTS/CTS mechanism and overall network load, these do not require significant modifications in the CPN model. These changes mainly affect the guards in the transitions *Start_Send*, *RTS_IMM* and modify the functions *gen_new_time* and *tsend*, which produce new packets and set the transmission times, respectively. The same scenarios that have been considered in this paper could then be used in order to analyze the impact of these changes. However, introducing the effect of a variable number of hidden stations and the carrier sensing area will require further changes in the CPN model to capture the required information. Thus, the scenarios here considered should be extended with the appropriate configuration information in order to establish the stations that are visible from each station and the carrier sensing area.

Acknowledgements This work was supported by the Spanish Ministry of Science and Innovation (co-financed by European Union FEDER funds) projects "DArDOS (Formal development and analysis of complex systems in distributed contexts: foundations, tools and applications)", reference TIN2015-65845-C3-02-R and project "FAME (Formal modeling and advanced testing methods. Applications to medicine and computing systems)", reference RTI2018-093608-B-C32. There was also support from the Junta de Comunidades de Castilla-La Mancha project SBPLY/17/180501/000276/01 (cofunded with FEDER funds, EU).

Appendix A

Listing 5 contains the main function declarations used in the CPN model.

Appendix B

Tables 17, 18, 19, 20 and 21 contain the scenarios and the expected output for them that have been identified for the model validation. These scenarios allow us to check the test cases presented in Table 3. For these proofs, we remove the initial marking *M_MSG(nt)* from place *Last_Arrival_Time*. In addition, unless we say otherwise, we assume that the tokens on the place *Control_Time_CTS* have a timestamp smaller than or equal to the current time of simulation (*Time*). All other places keep their initial markings, unless we say otherwise in the scenarios. In some cases, some places and transitions have been included in the tests to enforce the marking scenarios at specific times, in order to have the configuration required for the test case in an easy way.

Table 17 Test cases used for the model validation (part I)

	Scenario	Expected actions/output
1	4 stations to start. $RTS = 1$, $M(Time\_Arrival\_MSG) = 1`1@0 + + + 1`2@0 + + + 1`3@0 + + + 1`4@0$, $nt1 = 0$, $nt2 = 4$, $n2p = (1, 1, 1, 1)$	Firing of *init_st* 4 times. Configuration correct: $M(ST\_CONF) = 1`(1, 73, 31, 1023) + + 1`(2, 37, 31, 1023) + + 1`(3, 28, 15, 31) + + 1`(4, 28, 7, 15)$
2	1 station ready to start protocol. $RTS = 1$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 0, 1, 0)$, $M(start) = 1`(1, 0, 0, 0)@0$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$	Medium free, station ready to transmit after AIFS: *start* fires at time 0, then *EndTM* at time 28 (AIFS for video)
3	1 station detects AP transmission: $RTS = 1$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 0, 1, 0)$, $M(TM) = 1`(1, 0, 0, 0)@28$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $Time = 20$, $M(Medium) = 1`(1, 0)$, $M(AP) = 1`(0, 1, 0)$	Medium change detected (AP transmitting): *MediumChange* fires at time 20 and $M(TM) = 1`(1, 1, 1, 0)@48$
4	1 station detects end of transmission from AP. $RTS = 1$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 0, 1, 0)$, $M(TM) = 1`(1, 1, 1, 0)@28$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $Time = 20$, $M(Medium) = 1`(1, 0)$, $M(AP) = 1`(0, 0, 0)$	Medium change detected (end of transmission from AP), Back-off required: *MediumChange* fires at time 20 and $M(TM) = 1`(1, 0, 1, 0)@48$
5	1 station ready to transmit RTS frame. $RTS = 1$, $nt1 = 0$, $nt2 = 1, n2p = (0, 0, 1, 0)$, $M(MedState) = 1`(1, 0)@0$, $M(AP) = 1`(0, 0, 0)$, $M(Medium) = 1`(1, 0)$, $Time = 0$	*RTS_IMM* fires immediately, station 1 sending RTS frame: $M(Medium) = 1`(1, 1)$, $M(AP) = 1`(1, 0, 0)$, $M(Medium) = 1`(1, 0)$, $M(Sending\_RTS) = 1`1@352$ and $M(Wait\_CTS) = 1`1@818$
6	2 stations, st2 sending some frame and st1 starts RTS transmission. $RTS = 1$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(MedState) = 1`(1, 0)$, $M(AP) = 1`(1, 0, 0)$, $M(Medium) = 1`(1, 0) + + 1`(2, 1)$, $Time = 0$	*RTS_IMM* fires immediately, both stations transmitting, collision: $M(Medium) = 1`(1, 1) + + 1`(2, 1)$, $M(AP) = 1`(2, 0, 1)$, $M(Sending\_RTS) = 1`1@352$ and $M(Wait\_CTS) = 1`1@818$
7	No RTS/CTS, 1 station that starts transmission of message. $RTS = 0$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 0, 1, 0)$, $M(MedState) = 1`(1, 0)$, $M(AP) = 1`(0, 0, 0)$, $M(Medium) = 1`(1, 0)$, $Time = 0$	Station starts transmission of message (no RTS/CTS): *StartSend* fires immediately, $M(Medium) = 1`(1, 1)$, $M(AP) = 1`(1, 0, 0)$ $M(Sending\_Message) = 1`1@6258$ and $M(Wait\_ACK) = 1`1@6906$
8	No RTS/CTS, 2 stations, st2 sending, st1 starts RTS transmission. $RTS = 0$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(MedState) = 1`(1, 0)$, $M(AP) = 1`(1, 0, 0)$, $M(Medium) = 1`(1, 0) + + 1`(2, 1)$, $Time = 0$	Station 1 starts transmission of message (no RTS/CTS), both sending, collision: *StartSend* fires immediately, $M(Medium) = 1`(1, 1) + + 1`(2, 1)$, $M(AP) = 1`(2, 0, 1)$, $M(Sending\_Message) = 1`1@6258$ and $M(Wait\_ACK) = 1`1@6906$
9	1 video station starts Backoff counter when CW limit not reached. $RTS = 1$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 0, 1, 0)$, $M(MedState) = 1`(1, 1)$, $M(AP) = 1`(0, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 1)$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Process\_MSG) = 1`(1, 0)$, $M(IST) = \emptyset$, $M(Medium) = 1`(1, 0)$, $Time = 0$	Repeat several times to validate: *Backoff* fires: $M(Backoff\_Counter) = 1`(1, 2)$ and $M(Backoff\_Time) = 1`(1, x)$, with $x \in [1, 31]$

Listing 5 Function declarations.

```
fun M_NULL(n)=if (n>0) then 1`(n,0)++M_NULL(n-1) else nil;
fun M_JCONF(n)=if (n>0) then 1`(n,0,0,0)++M_JCONF(n-1) else nil;
fun M_TCTS(n)=if (n=1) then 1`1@0 else 1`n@0++M_TCTS(n-1);
fun M_TT(n)=if (n>0) then 1`(n,0,0.0)++M_TT(n-1) else nil;
fun gen_new(n)=if (n>nt1) then dexp(r2) else dexp(r1);
fun gen_new_time(n)=if (n>nt1) then 1`n@+dexp(r2) else nil;
fun M_MSG(n)=if (n>0) then 1`n@+gen_new(n)++M_MSG(n-1) else nil;
fun intTime()= IntInf.toInt(time());
fun get_AIFS(n)=case n of
1 => #1 AIFS | 2 => #2 AIFS | 3 => #3 AIFS | 4 => #4 AIFS;
fun s_send(s,ap,c)=
if ((c=2) orelse (s=0)) then (s+1,ap,c) else (s+1,ap,1);
fun update_med(s,c)=if (s=0) then 0 else c;
fun update_col(s,c)=if (s<=1) then 0 else c;
fun need_backoff(nb,s,c)=
if (c=2 orelse nb=1 orelse s>0) then 1 else 0;
fun update_avg(i,avg,t,arrival)=
((avg*(real i)+ real (t-arrival)))/(real (i+1));
fun update_ap(s,c1,c2)=
if (c1=2) then c1 else if (s=1) then 0 else c2;
fun new_imm(n)=if (n<=nt1) then 1`n else nil;
fun get_type(n)=if (n<=nt1) then 1 else 2;
fun get_n1p(i)=case i of
1 => #1 n1p |2 => #2 n1p |3 => #3 n1p |4 => #4 n1p;
fun get_n2p(i)=case i of
1 => #1 n2p |2 => #2 n2p |3 => #3 n2p |4 => #4 n2p;
fun get_CWMIN(i)=case i of
1 => #1 CWMIN |2 => #2 CWMIN |3 => #3 CWMIN |4 => #4 CWMIN;
fun get_CWMAX(i)=case i of
1 => #1 CWMAX |2 => #2 CWMAX | 3 => #3 CWMAX |4 => #4 CWMAX;
fun get_prio_aux1(n,i)=
if (n<=get_n1p(i)) then i else
get_prio_aux1(n-get_n1p(i),i+1);
fun get_prio_aux2(n,i)=
if (n<=get_n2p(i)) then i else
get_prio_aux2(n-get_n2p(i),i+1);
fun get_prio(n)=
if (n<=nt1) then get_prio_aux1(n,1) else get_prio_aux2(n-nt1,1);
fun tsend(n)=case get_prio(n) of
1=> #1 TIME_SENDING | 2 => #2 TIME_SENDING |
3 => #3 TIME_SENDING | 4 => #4 TIME_SENDING;
fun timeout(n)=tsend(n)+2*SIFS+TSEND_ACK+TTIMEOUT;
fun toutcts(n)=1+tsend(n)+2*SIFS+TSEND_ACK;
fun p2(bc)=if (bc=0) then 1 else
if (bc>10) then 1024 else 2*p2(bc-1);
fun get_cw(bc,cwmin)=(cwmin)*p2(bc);
fun new_arrival(n)=if (n<=nt1) then 1`(n,0,1,0) else 1`(n,0,0,0);
fun new_i_delayed(n)=if (n<=nt1) then 1`n@+dexp(r1) else nil;
fun init_conf(n)=(n,get_AIFS(get_prio(n)),get_CWMIN(get_prio(n)),
get_CWMAX(get_prio(n)));
```

Table 18 Test cases used for the model validation (part II)

10	1 video station drops frame when CW limit reached. $RTS = 1, nt1 = 0, nt2 = 1, n2p = (0, 0, 1, 0)$, $M(MedState) = 1`(1, 1)$, $M(AP) = 1`(0, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 2)$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Process\_MSG) = 1'(1, 0)$, $M(IST) = \emptyset$, $M(Medium) = 1`(1, 0)$, $Time = 0$	*Backoff* fires, frame dropped and new frame can be sent. $M(Backoff\_Counter) = 1`(1, 0)$, $M(Lost) = 1`1@0$, $M(IST) = 1`(1, 0, 0, 0)$
11	Decrement of Backoff time, when CTS not received. $RTS = 1, nt1 = 0, nt2 = 1, n2p = (0, 0, 1, 0)$, $M(Control\_time\_CTS) = 1`1@0$, $M(Backoff\_Time) = 1`(1, 4)@0$, $M(BO) = 1$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Medium) = 1`(1, 0)$, $M(AP) = 1`(0, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 2)$, $Time = 0$	*DecrBO* fired 4 times, every *SlotTime* time units
12	With RTS/CTS, start RTS transmission when Backoff time becomes 0. $RTS = 1, nt1 = 0, nt2 = 1, n2p = (0, 0, 1, 0)$, $M(Control\_time\_CTS) = 1`1@0$, $M(Backoff\_Time) = 1`(1, 0)@0$, $M(BO) = 1$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Medium) = 1`(1, 0)$, $M(AP) = 1`(0, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 2)$, $Time = 0$	*RTS_ABO* fires immediately, and tokens from *BO* and *Backoff_Time* removed. $M(Medium) = 1`(1, 1)$, $M(AP) = 1`(1, 0, 0)$, New tokens on *Sending_RTS* (1`1@352) and *Wait_CTS* (1`1@818)
13	Without RTS/CTS, start message transmission when Backoff time becomes 0. $RTS = 0, nt1 = 0, nt2 = 1, n2p = (0, 0, 1, 0)$, $M(Control\_time\_CTS) = 1`1@0$, $M(Backoff\_Time) = 1`(1, 0)@0$, $M(BO) = 1$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Medium) = 1`(1, 0)$, $M(AP) = 1`(0, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 2)$, $Time = 0$	*StartSendBO* fires immediately, and tokens from *BO* and *Backoff_Time* removed. $M(Medium) = 1`(1, 1)$, $M(AP) = 1`(1, 0, 0)$, New tokens on *Sending_Message* (1`1@6258) and *Wait_ACK* (1`1@6906)
14	With RTS/CTS, two stations, st2 sending and st1 in Backoff, st1 with Backoff time 0 can start RTS transmission if no CTS received. $RTS = 1, nt1 = 0, nt2 = 2, n2p = (0, 0, 1, 1)$, $M(Backoff\_Time) = 1`(1, 0)@0$, $M(ST\_CONF) = 1`(1, 28, 15, 31)++1`(2, 28, 7, 15)$, $M(Medium) = 1`(1, 0) + +1`(2, 1)$, $M(AP) = 1`(1, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 1) + +1`(2, 0)$, $M(BO) = 1$, $M(Control\_time\_CTS) = 1`1@0 + + + 1`2@0$, $Time = 0$	*RTS_ABO* fires immediately for st1, tokens from *BO* and *Backoff_Time* removed, both stations sending and collision: $M(Medium) = 1`(1, 1) + +1`(2, 1)$, $M(AP) = 1`(2, 0, 1)$, New tokens in *Sending_RTS* (1`1@352) and *Wait_CTS* (1`1@818)
15	Without RTS/CTS, two stations, st2 sending and st1 in Backoff, st1 with Backoff time 0 can start message transmission. $RTS = 0, nt1 = 0, nt2 = 2, n2p = (0, 0, 1, 1)$, $M(Backoff\_Time) = 1`(1, 0)@0$, $M(ST\_CONF) = 1`(1, 28, 15, 31)++1`(2, 28, 7, 15)$, $M(Medium) = 1`(1, 0) + +1`(2, 1)$, $M(AP) = 1`(1, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 1) + +1`(2, 0)$, $M(BO) = 1$, $M(Control\_time\_CTS) = 1`1@0 + + + 1`2@0$, $Time = 0$	*StartSendBO* fires immediately for st1, tokens from *BO* and *Backoff_Time* removed, both stations sending and collision: $M(Medium) = 1`(1, 1) + +1`(2, 1)$, $M(AP) = 1`(2, 0, 1)$, New tokens in $M(Sending\_Message)$ (1`1@6258) and $M(Wait\_ACK)$ (1`1@6906)

Table 19 Test cases used for the model validation (part III)

	Scenario	Expected actions/output
16	With RTS/CTS, 2 stations, st2 sending, st1 in Backoff period with CTS received can decrement Backoff time when CTS time-out elapses. $RTS = 1$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(Backoff\_Time) = 1`(1, 1)@0$, $M(BO) = 1$, $M(ST\_CONF) = 1`(1, 28, 15, 31) + +1`(2, 28, 7, 15)$, $M(Medium) = 1`(1, 0) + +1`(2, 1)$, $M(AP) = 1`(1, 0, 0)$, $M(Backoff\_Counter) = 1`(1, 2) + +1`(2, 0)$, $M(Control\_time\_CTS) = 1`1@818 + + + 1`2@0$, $Time = 0$	*DecrBO* fires at time 818
17	With RTS/CTS, 2 stations, st2 sending, st1 starts RTS transmission (collision). $RTS = 1$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(ST\_CONF) = 1`(1, 28, 15, 31) + +1`(2, 28, 7, 15)$, $M(Medium) = 1`(1, 1) + +1`(2, 1)$, $M(AP) = 1`(2, 0, 1)$, $M(Control\_time\_CTS) = 1`1@0 + + + 1`2@0$, $M(Sending\_RTS) = 1`1@352$, $M(Wait\_CTS) = 1`1@818$, $Time = 0$	Time-out elapses and no CTS received (collision). St1 restarts protocol with Backoff: *CollRTS* fires at time 818. New token $1`(1, 1, 1, 0)@818$ in place *TM*, $M(Medium) = 1`(1, 0)+1`(2, 1))$, $M(AP) = 1`(1, 0, 1)$, Updated *Colls_RTS_ST* and new token in *Colls_RTS*
18	With RTS/CTS, 2 stations in collision, st1 restarted protocol, st2 sending RTS. $RTS = 1$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(ST\_CONF) = 1`(1, 28, 15, 31) + +1`(2, 28, 7, 15)$, $M(Medium) = 1`(1, 0) + +1`(2, 1)$, $M(AP) = 1`(1, 0, 1)$, $M(Control\_time\_CTS) = 1`1@0 + + + 1`2@0$, $M(Sending\_RTS) = 1`2@900$, $M(Wait\_CTS) = 1`2@1366$, $Time = 818$	Collision detected for st2, medium free: *CollRTS* fires at time 818. New token $1`(2, 1, 1, 0)@818$ in place *TM*, $M(Medium) = 1`(1, 0) + +1`(2, 0)$, $M(AP) = 1`(0, 0, 0)$. Updated *Colls_RTS_ST* and new token in *Colls_RTS*
19	With RTS/CTS, 1 station. AP starts CTS transmission when RTS received. $RTS = 1$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 0, 1, 0)$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Medium) = 1`(1, 1)$, $M(AP) = 1`(1, 0, 0)$, $M(Control\_time\_CTS) = 1`1@0$, $M(Sending\_RTS) = 1`1@352$, $M(Wait\_CTS) = 1`1@818$, $Time = 0$	*End_RTS* fires at time 352, then *Start_CTS* fires after SIFS time units (time 362). New token in place *Sending_CTS* ($1`1@514$), and tokens for station 1 removed from *Sending_RTS* and *Wait_CTS*. $M(Medium) = 1`(1, 0)$, $M(AP) = 1`(1, 1, 0)$
20	With RTS/CTS. There cannot be several transmissions in parallel when CTS received	Defined a break point monitor *"BP_NoSevTransmCTSrec"* to check that $M(AP)$ cannot be $1`(2, 0, 2)$. Experiment performed with 40 intermittent stations. The monitor did not stop the simulation
21	With RTS/CTS, 2 stations, st1 sent RTS and now waiting for CTS. $RTS = 1$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(Medium) = 1`(1, 0) + +1`(2, 0)$, $M(AP) = 1`(1, 1, 0)$, $M(ST\_CONF) = 1`(1, 28, 15, 31) + +1`(2, 28, 7, 15)$, $M(Control\_time\_CTS) = 1`1@0 + 2`@0$, $M(Sending\_CTS) = 1`1@514$, $M(Start) = 1`(2, 0, 0, 0)@600$, $Time = 514$	End of CTS transmission when no collision. *CTS_OK* fires at time 514 and $M(Update\_Times) = 1`1$

Table 20 Test cases used for the model validation (part IV)

22	With RTS/CTS, 2 stations, CTS collision when st1 sending RTS, st2 waiting for CTS and AP sending CTS. $RTS = 1$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(Medium) = 1`(1, 0) + +1`(2, 0)$, $M(AP) = 1`(1, 1, 0)$, $M(ST\_CONF) = 1`(1, 28, 15, 31) + +1`(2, 28, 7, 15)$, $M(Control\_time\_CTS) = 1`1@0 + 2`@0$, $M(Sending\_CTS) = 1`2@514$, $M(MedState) = 1`(1, 0)@450$, $Time = 450$	*RTS_IMM* fires at time 450. Places *Sending_RTS* and *Wait_CTS* marked (st1 sending RTS). Transition *COL_CTS* fires at time 514. St2 restarts protocol with Backoff required with new token $1`(2, 2, 1, 0)@514$ in place *TM*. Updated counter in *Colls_CTS_ST* for st2 and new token in place *Colls_CTS*. $M(AP) = 1`(1, 0, 1)$ (collision detected) and $M(Medium) = 1`(1, 1) + +1`(2, 0)$
23	With RTS/CTS, 3 stations, st2 sent CTS. $RTS = 1$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 1, 1, 1)$, $M(Update\_Times) = 1`2$, $M(Control\_Time\_CTS) = 1`1@0 + + + 1`2@0 + + + 1`3@0$, $Time = 0$	Sequence of transitions *Update_Other*, *Stat_n Update_Other* and *End* is fired to reach the marking $M(Control\_Time\_CTS) = 1`1@793 + + + 1`2@0 + + + 1`3@793$, $M(Sending\_Message) = 1`2@458$, $M(Wait\_ACK) = 1`2@1106$ and $M(Stations) = 1`3$
24	With RTS/CTS, 1 intermittentstation that terminates message transmission. $RTS = 1$, $nt1 = 0$, $nt2 = 1$, $n2p = (0, 0, 1, 0)$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Medium) = 1`(1, 1)$, $M(AP) = 1`(1, 0, 2)$, $M(Sending\_Message) = 1`1@7174$,$M(Backoff\_Counter) = 1`(1, 1)$, $M(Wait\_ACK) = 1`1@7822$, $M(Process\_MSG) = 1`(1, 374)$, $Time = 7174$	ACK sent and new frame can be sent from st1: Transition *End_Transm* fires at time 7174 and $M(AP) = 1`(0, 0, 2)$, $M(Medium) = 1`(1, 1)$. After SIFS delay transition *StartACK* fires, producing new token $1`1@7498$ in place *Sending_ACK*. $M(AP) = 1`(1, 1, 2)$ and $M(Medium) = 1`(1, 1)$. *EndACK* then fires at time 7498, removing the tokens from *Process_MSG* and *Wait_ACK*, $M(AP) = 1`(0, 0, 0)$, $M(Medium) = 1`(1, 0)$, $M(Backoff\_Counter) = 1`(1, 0)$, $M(IST) = 1`(1, 0, 0, 0)$, new token $1`1@7498$ in place *SUCC_TRANS*, no token produced in *Time_Arrival_MSG* and updated the transmission times in *Trans_Times*

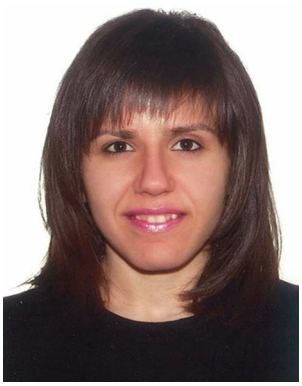
Table 21 Test cases used for the model validation (part V)

25	With RTS/CTS, 1 saturated station that terminates message transmission. $RTS = 1$, $nt1 = 1$, $nt2 = 0$, $n1p = (0, 0, 1, 0)$, $M(ST\_CONF) = 1`(1, 28, 15, 31)$, $M(Medium) = 1`(1, 1)$, $M(AP) = 1`(1, 0, 2)$, $M(Sending\_Message) = 1`1@7174$, $M(Backoff\_Counter) = 1`(1, 1)$, $M(Wait\_ACK) = 1`1@7822$, $M(Process\_MSG) = 1`(1, 374)$, $Time = 7174$	ACK sent and new frame can be sent from st1: Transition *End_Transm* fires at time 7174 and $M(AP) = 1`(0, 0, 2)$, $M(Medium) = 1`(1, 1)$. After SIFS delay transition *StartACK* fires, producing new token $1`1@7498$ in place *Sending_ACK*. $M(AP) = 1`(1, 1, 2)$ and $M(Medium) = 1`(1, 1)$. *EndACK* then fires at time 7498, removing the tokens from *Process_MSG* and *Wait_ACK*, $M(AP) = 1`(0, 0, 0)$, $M(Medium) = 1`(1, 0)$, $M(Backoff\_Counter) = 1`(1, 0)$, $M(IST) = 1`(1, 0, 0, 0)$, new token $1`1@7498$ in place *SUCC_TRANS*, new token $1`1@7498$ produced in *Time_Arrival_MSG* and updated the transmission times in *Trans_Times*
26	Same as 24, but without RTS/CTS (RTS=0 and $M(AP) = 1'(1, 0, 0)$)	Same effects as 24
27	Same as 25, but without RTS/CTS (RTS=0 and $M(AP) = 1'(1, 0, 0)$)	Same effects as 25
28	Without RTS/CTS, 2 stations sending. $RTS = 0$, $nt1 = 0$, $nt2 = 2$, $n2p = (0, 0, 1, 1)$, $M(ST\_CONF) = 1`(1, 28, 15, 31) + +1`(2, 28, 7, 15)$, $M(Medium) = 1`(1, 1) + +1`(2, 1)$, $M(AP) = 1`(2, 0, 1)$, $M(Wait\_ACK) = 1`1@7289 + + + 1`2@7631$, $M(Sending\_Message) = 1`1@6641 + + + 1`2@6983$, $Time = 7289$	Message collision detected for st1. New frame can be sent after Backoff period. Collision for st2 annotated in *AP*: *CollMSG* fires at time 7289 for st1. The corresponding tokens are removed from *Sending_Message* and *Wait_ACK*, $M(Medium) = 1`(1, 0) + +1`(2, 1)$, $M(AP) = 1`(1, 0, 1)$. New token $1`(1, 1, 1, 0)@7289$ produced in place *TM* and collision annotated in *Colls_MSG* and *Colls_MSG_ST*. *CollMSG* fires for st2 at time 7631
29	No message collision when using RTS/CTS (only RTS and CTS collisions are possible)	Place content monitor *"PC_With_RTS_No_CollsMSG"* defined, which checks that place *Colls_MSG* is never marked. Experiment performed with 40 intermittent stations. The monitor did not stop the simulation
30	No ACK collision when using RTS/CTS	Place content monitor *PC_No_ACK_Coll"* defined to check that place *Colls_ACK* is never marked. Experiment performed with 40 intermittent stations. The monitor did not stop the simulation.

References

1. LAN/MAN Standards Committee of the IEEE Computer Society: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 7: Medium Access Control (MAC) Quality of Service (QoS). IEEE 802.11e (2005)
2. LAN/MAN Standards Committee of the IEEE Computer Society: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 2: Sub 1 GHz License Exempt Operation. IEEE 802.11ah (2016)
3. Simó, J., Figuera, C., Seoane, J., Martinez, A.: Distance Limits in IEEE 802.11 for Rural Networks in Developing Countries. In: Proceedings of IEEE WRECOM. pp. 1–5. Rome, Italy (2007)
4. Wu, H., Zhu, F., Zhang, Q., Niu, Z.: WSN02-1: Analysis of IEEE 802.11 DCF with Hidden Terminals. In: Proceedings of IEEE Globecom 2006. pp. 1–5 (2006)
5. Chang, Z., Alanen, O., Huovinen, T., Nihtila, T., Ong, E.H., Kneckt, J., Ristaniemi, T.: Performance Analysis of IEEE 802.11ac DCF with Hidden Nodes. In: Proceedings of IEEE VTC Spring. pp. 1–5 (2012)
6. Tsertou, A., Laurenson, D.I.: Insights into the Hidden Node Problem. In: Proceedings of ACM IWCMC. pp. 767–772 (2006)
7. Perez, S., Facchini, H., Dantiacq, A., Cangemi, G., Campos, J.: An Evaluation of QoS for Intensive Video Traffic over 802.11e WLANs. In: Proceedings of CONIELECOMP. pp. 8–15 (2015)
8. Khiat, A., El Khaili, M., Bakkoury, J., Bahnasse, A.: Study and Evaluation of Voice over IP Signaling Protocols Performances on MIPv6 Protocol in Mobile 802.11 Network: SIP and H.323. In: Proceedings of ISNCC. pp. 1–8 (2017)
9. Maity, M., Raman, B., Vutukuru, M.: TCP download performance in dense WiFi scenarios: analysis and solution. IEEE Trans. Mob. Comput. **16**(1), 213–227 (2017)
10. Pastrav, A.E.I., Puschita, E., Palade, T.: HCCA support in IEEE 802.11 networks QoS and QoE performance evaluation. In: Proceedings of IEEE ISETC. pp. 139–142 (2012)
11. Talebi, M., Papatsimpa, C., Linnartz, J.M.G.: Dynamic Performance Analysis of IEEE 802.15.4 Networks under Intermittent Wi-Fi Interference. In: Proceedings of IEEE PIMRC. pp. 1–7 (2018)
12. Gomez, J., Riggi, A.: Region DCF: a self-adapting CSMA/Round-Robin MAC for WLAN. Wirel. Pers. Commun. **85**(4), 2169–2190 (2015)
13. Aust, S., Prasad, R.V., Niemegeers, I.G.: Outdoor long-range WLANs: a lesson for IEEE 802.11ah. IEEE Commun. Surv. Tutor. **17**(3), 1761–1775 (2015)
14. Damayanti, W., Kim, S., Yun, J.H.: Collision chain mitigation and hidden device-aware grouping in large-scale IEEE 802.11ah networks. Comput. Netw. **108**(C), 296–306 (2016)
15. Bianchi, G., Di Stefano, A., Giaconia, C., Scalia, L., Terrazzino, G., Tinnirello, I.: Experimental assessment of the backoff behavior of commercial IEEE 802.11b network cards. In: Proceedings of IEEE INFOCOM. pp. 1181–1189 (2007)
16. Ahemd, F.U., Sarma, S.K.: QoS and admission controller in IEEE 802.11e WLAN. In: 2013 4th International Conference on Intelligent Systems, Modelling and Simulation, pp. 468–471 (2013)
17. Syed, I., Roh, B.: Delay analysis of IEEE 802.11e EDCA with enhanced QoS for delay sensitive applications. In: Proceedings of IEEE IPCCC. pp. 1–4 (2016)
18. Cranley, N., Davis, M.: An experimental investigation of IEEE 802.11e TXOP facility for real-time video streaming. In: Proceedings of IEEE GLOBECOM, pp. 2075–2080 (2007)
19. Ramakristanaiah, C., Reddy, P.C., Sam, R.P.: Evaluation of starvation problem under saturated loads in IEEE 802.11e. In: 2016 International Conference on Emerging Technological Trends (ICETT), pp. 1–7 (2016)
20. Yang, X., Liu, R.P., Hedley, M.: A Channel access cycle based model for IEEE 802.11e EDCA in unsaturated traffic conditions. In: 2012 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1496–1501 (2012)
21. Tang, J., Cheng, Y., Zhuang, W.: Real-time misbehavior detection in IEEE 802.11-based wireless networks: an analytical approach. IEEE Trans. Mob. Comput. **13**(1), 146–158 (2014)
22. Vijayasankar, K., Taufique, A., Kannan, L.N., Tacca, M., Fumagalli, A.: An analytical model with improved accuracy of IEEE 802.11 protocol under unsaturated conditions. In: 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 385–387 (2010)
23. Prakash, G., Thangaraj, P.: Throughput analysis of IEEE 802.11e EDCA under non saturation condition. In: Proceedings of ICECT, vol. 6, pp. 117–121 (2011)
24. Masri, M., Juanole, G., Abdellatif, S.: Revisiting the Markov chain model of IEEE 802.11E EDCA and introducing the virtual collision phenomenon. In: Proceedings of ACM WINSYS. Barcelona, Spain, pp. 76–84 (2007)
25. Chakraborty, S.: Analyzing peer specific power saving in IEEE 802.11s through queuing Petri Nets: some insights and future research directions. IEEE Trans. Wirel. Commun. **15**(5), 3746–3754 (2016)
26. Perez, S., Facchini, H., Bisaro, L., Campos, J.: Tuning mechanism for IEEE 802.11e EDCA optimization. IEEE Lat. Am. Trans. **11**(4), 1134–1142 (2013)
27. Jensen, K., Kristensen, L.M.: Coloured Petri Nets—Modelling and Validation of Concurrent Systems. Springer, Berlin (2009)
28. Peterson, J.: Petri Net Theory and the Modeling of Systems. Prentice-Hall, Upper Saddle River (1981)
29. CPN Tools (2019). http://cpntools.org/. accessed on 10.06.2020
30. Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function. IEEE J. Sel. Areas Commun. **18**(3), 535–547 (2000)
31. Milner, R., Tofte, M., Macqueen, D.: The Definition of Standard ML. MIT Press, Cambridge, MA (1997)
32. LAN/MAN Standards Committee of the IEEE Computer Society: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band. IEEE 802.11g (2003)
33. Boujnoui, A., Orozco-Barbosa, L., Haqiq, A.: Performance evaluation and tuning of an IEEE 802.11 audio video multicast collision prevention mechanism. Wirel. Netw. (2020) (in press)
34. Cambronero, M.E., Macia, H., Valero, V., Orozco-Barbosa, L.: Modeling and analysis of the 1-wire communication protocol using timed colored Petri Nets. IEEE Access **6**, 27356–27372 (2018)
35. Hu, X., Jiao, L., Li, Z.: Modelling and performance analysis of IEEE 802.11 DCF using coloured Petri Nets. Comput. J. **59**(10), 1563–1580 (2016)
36. Hu, X., Jiao, L.: Efficient modeling and performance analysis for IEEE 802.15.4 with coloured petri nets. In: Proceedings of the IEEE/ACM IWQoS, pp. 1–6 (2017)
37. Somappa, A.A.K., Simonsen, K.I.F.: Model-based development for MAC protocols in industrial wireless sensor networks. In: Proceedings of IEEE PNSE. Toruń, Poland, pp. 193–212 (2016)
38. Alves, R., Margi, C.: Behavioral model of IEEE 802.15.4 beacon-enabled mode based on colored Petri net. ACM Trans. Model. Perform. Eval. Comput. Syst. **2**, 1–31 (2017)
39. German, R., Heindl, A.: Performance evaluation of IEEE 802.11 wireless LANs with stochastic Petri Nets. In: Proceedings 8th International Workshop on Petri Nets and Performance Models, pp. 44–53 (1999)
40. Heindl, A., German, R.: Performance modeling of IEEE 802.11 wireless LANs with stochastic Petri Nets. Perform. Eval. **44**(1–4), 139–164 (2001)

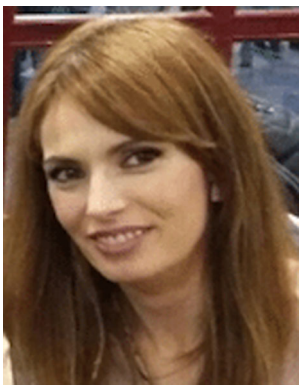
41. Moraes, R., Portugal, P., Vasques, F.: A Stochastic Petri Net model for the simulation analysis of the IEEE 802.11e EDCA communication protocol. In: Proceedings of IEEE ETFA, pp. 38–45 (2006)
42. Zairi, S., Niel, E., Zouari, B.: Global generic model for formal validation of the wireless sensor networks properties. Proc. IFAC World Congr. **44**(1), 5395–5400 (2011)
43. Zairi, S., Mezni, A., Zouari, B.: Formal approach for modeling, verification and performance analysis of wireless sensors network. In: Wired/Wireless Internet Communications. Springer, Cham, pp. 381–395 (2015)
44. Mokdad, L., Ben-Othman, J., Yahya, B., Niagne, S.: Performance evaluation tools for QoS MAC protocol for wireless sensor networks. Ad Hoc Netw. **12**, 86–99 (2014)



Estefanía Coronado is an Expert Researcher in the Smart Networks and Services Unit at Fondazione Bruno Kessler (Italy). In 2018 she completed her PhD at the University of Castilla-La Mancha (Spain) on multimedia distribution over SD-WLANs using ML. She received the M.Sc. degrees in Computer Engineering and Advanced Computer Technologies in 2014 and 2015. She published around 20 papers in international journals and conferences and received the IEEE INFOCOM Best Demo Award 2019. Her research interests include wireless and mobile communications, network slicing, SDN, NFV, AI-driven network management.

Valentín Valero received the degree in mathematics from the Complutense University of Madrid in 1987 and the Ph.D. degree in mathematics from the Department of Computer Science, Complutense University of Madrid, in 1993. Since 1987, he has been a member of the Computer Science Department, University of Castilla-La Mancha, Spain, where he is a Full Professor of distributed systems and operating systems with the Computer Science School of Albacete. His current research interests include concurrency, specifically in formal models for analysis and design of concurrent systems and real-time systems.

Luis Orozco-Barbosa received the B.Sc. degree in electrical and computer engineering from Universidad Autónoma Metropolitana, Mexico, in 1979, the Diplôme d'Etudes Approfondies in computer science from the Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées, France, in 1984, and the Doctorat d'Université in Computer Science from Université Pierre et Marie Curie, France, in 1987. From 1991 to 2002, he was a Faculty Member with the School of Information Technology and Engineering, University of Ottawa, Canada. In 2002, he joined the Department of Computer Engineering, University of Castilla La Mancha, Spain. In 2002, he was appointed as the Director of the Albacete Research Institute of Informatics. He has conducted numerous research projects with the private sector and served as a Technical Advisor for the Canadian International Development Agency and the Spanish International Cooperation Council. His current research interests include Internet protocols, performance evaluation, wireless multimedia communications, and IoT technologies. He is an IEEE Member.



María-Emilia Cambronero is an Associate Professor in Computer Science at Castilla-La Mancha University, Spain, obtaining the tenure distinction in 2012. She received her Ph.D. in 2007 from the University of Castilla-La Mancha and was an assistant professor for several years in the same university. Her research goals are aimed to make software more reliable, more secure, and easier to design. Her primary technical interests include software engineering and related areas, including contract specification, program monitoring, testing, and verification. Her research combines strong theoretical foundations with realistic experimentation in the area of web services and cloud computing.

Fernando L. Pelayo obtained his degree M.Sc. in Mathematics at the Complutense University of Madrid, his European Ph.D. in Computer Science at the University of Castilla - La Mancha and his Ph.D. in Applied Mathematics in Polytechnic University of Cartagena. Nowadays he is developing his teaching activities in the UCLM: courses on "5Algorithms, Analysis and Design" and "Automata Theory and formal aspects of Computing" . He also teaches at Spanish Distance Learning University, UNED, "Discrete Mathematics and Logic". All these subjects within the Computing Engineering M.Sc. His main research interests are focused on formal aspects of concurrency and performance, discrete dynamical systems, quantum computing and practical issues on control theory of embedded systems.