Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol [*]

Marta Kwiatkowska[1], Gethin Norman[1], and Jeremy Sproston[2]

[1] School of Computer Science, University of Birmingham,
Birmingham B15 2TT, United Kingdom
[2] Dipartimento di Informatica, Università di Torino, 10149 Torino, Italy

Abstract. The international standard IEEE 802.11 was developed recently in recognition of the increased demand for wireless local area networks. Its medium access control mechanism is described according to a variant of the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme. Although collisions cannot always be prevented, randomized exponential backoff rules are used in the retransmission scheme to minimize the likelihood of repeated collisions. More precisely, the backoff procedure involves a uniform probabilistic choice of an integer-valued delay from an interval, where the size of the interval grows exponentially with regard to the number of retransmissions of the current data packet. We model the two-way handshake mechanism of the IEEE 802.11 standard with a fixed network topology using probabilistic timed automata, a formal description mechanism in which both nondeterministic choice and probabilistic choice can be represented. From our probabilistic timed automaton model, we obtain a finite-state Markov decision process via a property-preserving discrete-time semantics. The Markov decision process is then verified using Prism, a probabilistic model checking tool, against probabilistic, timed properties such as "at most 5,000 microseconds pass before a station sends its packet correctly."

1 Introduction

Wireless communication devices are increasingly becoming part of our daily lives. In particular, *Wireless Local Area Networks* (WLANs) are often used in cases when data communication over a small area is required, but a wired network is not practical or economic. The international standard IEEE 802.11 was developed recently to cater for the burgeoning use of WLANs, and has enabled the use of heterogeneous communication devices from different vendors within the same network. In contrast to wired devices, stations of a wireless network cannot listen to their own transmission, and are therefore unable to employ medium access control schemes such as Carrier Sense Multiple Access with Collision Detection (CSMA/CD) in order to prevent simultaneous transmission on the channel. Instead, the IEEE 802.11 standard describes a Carrier Sense Multiple Access with

[*] Supported in part by the EPSRC grant GR/N22960, the CNR grant No.99.01716.CTO1, and the EU within the DepAuDE project IST-2001-25434.

H. Hermanns and R. Segala (Eds.), 2nd Joint Int. Workshop on Process Algebra and Performance Modelling and Probabilistic Methods in Verification, volume 2399 of LNCS, pages 169–187, 2002.

Collision Avoidance (CSMA/CA) mechanism, using a randomized exponential backoff rule to minimize the likelihood of transmission collision. The backoff procedure is implemented by first choosing an integer valued delay from a bounded interval, where the choice is made according to the uniform probability distribution over the interval. Then the station is required to wait for a length of time dependent on this integer-valued delay. An important characteristic of the backoff procedure, which results in the probability of repeated transmission collisions decreasing as the number of transmission collisions of a particular data packet increases, is that the size of the interval grows exponentially in the number of retransmissions.

Previous studies of the IEEE 802.11 standard have either concerned simulation [15] or analytic approaches from the field of performance evaluation [4,11]. In this paper, we consider automatic verification of a medium access control subprotocol of the IEEE 802.11 WLAN standard using *probabilistic model checking* [10,5]. Given a probabilistic model, expressed as a stochastic process such as a Markov decision process [9], and a (probabilistic) specification, such as "a data packet is delivered with probability 1", the probabilistic model checking algorithm determines which states of the model satisfy the specification.

We model a two-way handshake mechanism of the IEEE 802.11 medium access control scheme, operating in a fixed network topology consisting of two sending stations and two destination stations. Our modelling formalism is that of *probabilistic timed automata* [17], which, like Markov decision processes, allows both nondeterministic choice (used for example, to model asynchrony between sub-components of the system) and probabilistic choice (which, for example, is present in the randomized backoff procedure) to coexist in the same model. Probabilistic timed automata are an extension of timed automata [1]; that is, they are timed automata for which discrete probability distributions range over the edges of the control graph. Equivalently, probabilistic timed automata can be thought of as an extension of Markov decision processes for which the values of a set of real-valued clocks can influence the transitions from each state.

The initial stages of the modelling process employ the timed automata model checking tool Uppaal [20] to automatically verify the soundness of several abstractions applied to our probabilistic timed automaton model, in order to reduce its complexity in anticipation of probabilistic model checking. The correctness of this process relies on equipping the non-probabilistic Uppaal model with additional event labels to represent probabilistic choice. From the resulting, smaller probabilistic timed automaton, which nevertheless has an infinite number of states due to the presence of real-valued clock variables, we use a property preserving discrete time semantics to obtain a finite state Markov decision process. We then use the probabilistic model checking tool Prism [16,21] to verify properties referring both to the likelihood of repeated transmission collision, and to the probability that a station sends a packet correctly within a certain deadline. In contrast to previous numerical analyses of the IEEE 802.11 medium access control scheme, such as [11], we use nondeterminism to model the interleaving which results from asynchronous parallel composition of system components, to model unspecified time delays, and as a conservative over-approximation mechanism when constructing abstractions. Following this methodology, the results we compute through probabilistic model checking give upper and lower bounds on the probability of satisfying the properties of interest.

The paper proceeds by first giving an informal description of a two-way handshake sub-protocol of the IEEE 802.11 standard in the next section. In Section 3, we introduce probabilistic timed automata, defining both their continuous and discrete-time semantics. Section 4 then explains how probabilistic timed automata can be used to model the sub-protocol, describes the construction of the abstract models, and presents the probabilistic model checking results. Section 5 concludes the paper.

2 The Basic Access mechanism of the IEEE 802.11 DCF

Our focus is on a contention resolution protocol of a basic class of IEEE 802.11 WLAN. The class, referred to as the Independent Basic Service Set or "ad hoc networks", comprises a number of stations communicating over a shared channel in a peer-to-peer manner, without a centralized medium access control (MAC) protocol arbitrating requests to transmit on the channel. Instead, the aim of MAC schemes for such networks is to keep the number of *collisions* (simultaneous transmissions) on the channel to a minimum. For this purpose, the IEEE 802.11 standard defines a Distributed Coordination Function (DCF) based on a CSMA/CA protocol. An important feature of the DCF is that of a randomized, slotted exponential backoff, which is designed to break the symmetry between stations that are repeating previously failed transmissions i.e. transmissions which collided.

As the IEEE 802.11 standard specifies that a sending station monitors the channel prior to transmitting, collisions can occur if multiple stations are simultaneously in their *vulnerable period*. If a station is in this period, which occurs when it starts to send its data, then this transmission can only be detected by other stations after some delay (equal to the length of the vulnerable period); hence, another station may also decide to begin transmitting, resulting in a collision. The duration of the vulnerable period is given by the sum of (1) the time taken for a station to assess the channel and deliver its state to the MAC layer, (2) the time taken for the destination station to change from a receiving to a transmitting state, and (3) the air propagation time.

The standard defines two transmission mechanisms, of which we focus on Basic Access (BA). In this scheme, when a station in a WLAN is ready to transmit a new data packet, it must sense that the channel is free for a duration given by the DCF Interfame Space (DIFS), the length of which depends on the physical layer, and which should be at least as long as the vulnerable period. If the channel is free for this period, then the station can commence transmission of its frame to another station. Upon termination of the transmission, the sending station listens immediately to the channel, in order to detect whether another station is currently transmitting. If so, the sending station decides that a collision

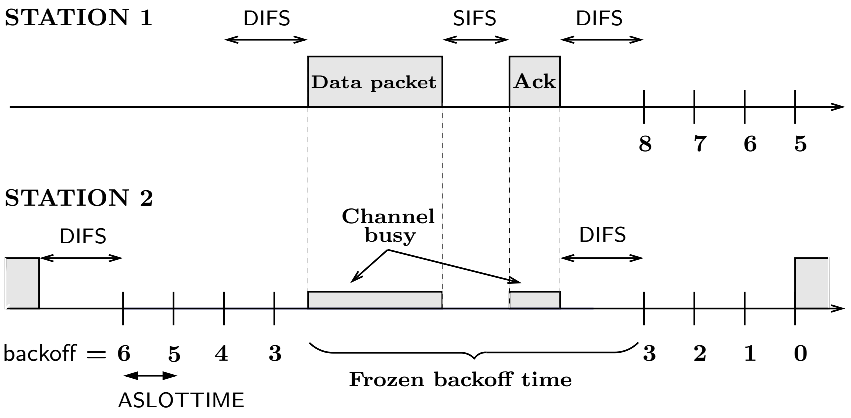

Fig. 1. An example of the backoff procedure.

has occurred; if not, it then waits for an acknowledgement to be sent from the destination station. The importance of the acknowledgement in the context of wireless devices can be seen on consideration that a sending station cannot listen to its own transmission; if this was instead possible, then the station could detect that a transmission was successful.

The sending station enters the *backoff procedure* if either:

- the channel is not sensed idle for a DIFS;
- the channel is sensed busy after the station finishes a data transmission;
- a positive acknowledgement of successful transmission is not received from the destination station before a timeout; or
- the station receives an acknowledgement and wishes to send another packet.

The backoff procedure first consists of the station monitoring the channel; if it is busy, the station waits until it is free, after which it must continue to be sensed free for a DIFS. Next, there is a random choice of backoff value, which indicates the number of time periods called "slots" which must be passed through before the station can start transmitting. The duration of the slot is given by ASLOTTIME, and must be at least as large as the vulnerable period. If the channel is detected idle for an ASLOTTIME, the backoff value is decremented by 1. This decrementing procedure is temporarily suspended if a transmission is detected, and is resumed *only* after the channel is sensed free for DIFS time units. When the backoff value reaches 0, the station can commence its transmission.

An example of a backoff procedure is illustrated in Figure 1, which is adapted from [4]. We consider the case in which station 2 has just finished sending a data packet, whereas station 1 has yet to send a packet. After station 2 has finished transmission, its waits for DIFS time units before selecting randomly the backoff value of 6. Station 2 then proceeds to decrement the value of its backoff value by 1 for each duration of length ASLOTTIME passed through. However, after detecting that the channel is free for DIFS time units, station 1 decides to send its data packet before station 2 has finished its backoff countdown. The figure shows how the backoff value of station 2 is frozen while the channel is occupied, both when the data packet and the accompanying acknowledgement are sent, and also during the Short Interframe Space[1] (SIFS) which separates these transmissions. That the backoff countdown is frozen during SIFS follows from the fact that the timing parameters of the IEEE 802.11 standard specify that $\mathsf{SIFS} < \mathsf{DIFS}$, and that the channel must be detected free for DIFS time units before the backoff countdown can be resumed.

The random selection of backoff value is implemented as a uniform distribution over integers in the range $[0, CW]$ (the contention window), where $CW = (aCWmin + 1) \cdot 2^{bc} - 1$; the value $aCWmin$ is a constant given by the physical layer, whereas bc is a variable called the *backoff counter*, which represents the number of unsuccessful retransmissions of the pending data packet that have been made (therefore bc is initially 0). The backoff counter can increase to a ceiling imposed by MAX_BACKOFF, which again is a constant given by the physical layer (more precisely, it is calculated through the constant $aCWmax$, given by the physical layer, which defines the maximal contention window).

We impose a number of restrictions and assumptions when modelling the IEEE 802.11 Basic Access DCF mechanism. A fixed network topology, consisting of two sending stations and two destination stations is assumed, meaning that the Extended Interfame Space is not modelled. We also do not consider the Timing Synchronization Function, which stipulates that short frames are periodically broadcast by a designated station in order to synchronize the local clocks of all other stations. Finally, for simplicity we assume that retry limits, which bound the number of retransmissions of a data packet, are set to infinity. However, we anticipate that these features could be included in our model in the future.

3 Probabilistic Timed Automata

3.1 Syntax of probabilistic timed automata

Time, clocks, zones and distributions. Let $\mathbb{T} \in \{\mathbb{R}, \mathbb{N}\}$ be the *time domain* of either the non-negative reals or naturals. Let $\mathcal{X}$ be a finite set of variables called *clocks* which take values from the time domain $\mathbb{T}$. A point $v \in \mathbb{T}^{|\mathcal{X}|}$ is referred to as a *clock valuation*. Let $\mathbf{0} \in \mathbb{T}^{|\mathcal{X}|}$ be the clock valuation which assigns 0 to all clocks in $\mathcal{X}$. For any $v \in \mathbb{T}^{|\mathcal{X}|}$ and $t \in \mathbb{T}$, the clock valuation $v \oplus t$ denotes the *time increment* for v and t (we present two alternatives for $\oplus$ in Section 3.2; for the time domain $\mathbb{R}$ it is standard addition $+$). We use $v[X := 0]$ to denote the clock valuation obtained from v by resetting all of the clocks in $X \subseteq \mathcal{X}$ to 0, and leaving the values of all other clocks unchanged.

[1] The Short Interframe Space is the time the IEEE 802.11 standard specifies that a destination station should wait for after successfully receiving data.

Let $\mathit{Zones}(\mathcal{X})$ be the set of *zones* over $\mathcal{X}$, which are conjunctions of atomic constraints[2] of the form $x \sim c$ for $x \in \mathcal{X}$, $\sim \in \{\leq, =, \geq\}$, and $c \in \mathbb{N}$. The clock valuation v *satisfies* the zone ζ, written $v \triangleleft \zeta$, if and only if ζ resolves to true after substituting each clock $x \in \mathcal{X}$ with the corresponding clock value from v.

A discrete probability *distribution* over a countable set Q is a function $\mu : Q \to [0,1]$ such that $\sum_{q \in Q} \mu(q) = 1$. For a possibly uncountable set Q', let $\mathsf{Dist}(Q')$ be the set of distributions over countable subsets of Q'. For some element $q \in Q$, let $\mu_q \in \mathsf{Dist}(Q)$ be the distribution which assigns probability 1 to q.

Syntax of probabilistic timed automata. We review the definition of probabilistic timed automata [17]. An added feature is that of urgent events, which are a well-established concept for classical timed automata [12,7].

Definition 1. *A* probabilistic timed automaton *is a tuple* $(L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ *where: L is a finite set of* locations *including the* initial location $\bar{l} \in L$*; Σ is a finite set of* events, *of which $\Sigma_u \subseteq \Sigma$ are declared as being* urgent*; the function $\mathit{inv} : L \to \mathit{Zones}(\mathcal{X})$ is the* invariant condition*; the finite set $\mathit{prob} \subseteq L \times \mathit{Zones}(\mathcal{X}) \times \Sigma \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$ is the* probabilistic edge relation.

A *state* of a probabilistic timed automaton is a pair (l, v) where $l \in L$ and $v \in \mathbb{T}^{|\mathcal{X}|}$ are such that $v \triangleleft \mathit{inv}(l)$. Informally, the behaviour of a probabilistic timed automaton can be understood as follows. The model starts in the initial location $\bar{l}$ with all clocks set to 0, and hence the initial state is $(\bar{l}, \mathbf{0})$. In this, and any other state (l, v), there is a nondeterministic choice of either (1) making a *discrete transition* or (2) letting *time pass*. In case (1), a discrete transition can be made according to any $(l, g, \sigma, p) \in \mathit{prob}$ with source location l which is *enabled*; that is, g is satisfied by the current clock valuation v. Then the probability of moving to the location l' and resetting all of the clocks in X to 0 is given by $p(X, l')$. In case (2), the option of letting time pass is available only if the invariant condition $\mathit{inv}(l)$ is satisfied while time elapses and there does not exist an enabled probabilistic edge with an urgent event.

Note that a *timed automaton* [1] is a probabilistic timed automaton for which every probabilistic edge (l, g, σ, p) is such that $p = \mu_{(X,l')}$ for some $(X, l') \in 2^{\mathcal{X}} \times L$.

Higher-level modelling. To aid higher-level modelling, a notion of urgency can be associated with locations, in addition to events. Once an urgent location is entered, it must be left immediately, without time passing. Urgent locations can be represented syntactically using an additional clock [7,26].

Integer variables with bounded ranges, which can be tested within enabling conditions and reset by edge distributions, can also be represented syntactically within the probabilistic timed automaton framework above by encoding the values of such variables within locations [25]. Indeed, the probabilistic choice of the

[2] Readers familiar with timed automata will note that we consider the syntax of *closed* zones, which do not feature atomic constraints of the form $x > c$ or $x < c$.

backoff procedure in the IEEE 802.11 WLAN sub-procedure that we study takes the form of a random assignment to an integer variable; hence notation such as $\mathsf{backoff} := \mathit{RANDOM}()$, for an integer variable $\mathsf{backoff}$, should be interpreted as a probabilistic choice between locations, as is standard for probabilistic timed automata.

It is often useful to define complex systems as the *parallel composition* of a number of interacting sub-components. The definition of the parallel composition operator $\|$ uses ideas from the theory of (untimed) probabilistic systems [23] and classical timed automata [1]. Let $\mathsf{PTA}_i = (L_i, \bar{l}_i, \mathcal{X}_i, \Sigma_i, \mathit{inv}_i, \mathit{prob}_i)$ for $i \in \{1, 2\}$.

Definition 2. *The* parallel composition *of two probabilistic timed automata* PTA_1 *and* PTA_2 *is the probabilistic timed automaton* $\mathsf{PTA}_1 \| \mathsf{PTA}_2 = (L_1 \times L_2, (\bar{l}_1, \bar{l}_2), \mathcal{X}_1 \cup \mathcal{X}_2, \Sigma_1 \cup \Sigma_2, \mathit{inv}, \mathit{prob})$ *where* $\mathit{inv}(l, l') = \mathit{inv}_1(l) \wedge \mathit{inv}_2(l')$ *for all* $(l, l') \in L_1 \times L_2$ *and* $((l_1, l_2), g, \sigma, p) \in \mathit{prob}$ *if and only if one of the following conditions holds:*

- $\sigma \in \Sigma_1 \setminus \Sigma_2$ *and there exists* $(l_1, g, \sigma, p_1) \in \mathit{prob}_1$ *such that* $p = p_1 \otimes \mu_{(\emptyset, l_2)}$*;*
- $\sigma \in \Sigma_2 \setminus \Sigma_1$ *and there exists* $(l_2, g, \sigma, p_2) \in \mathit{prob}_2$ *such that* $p = \mu_{(\emptyset, l_1)} \otimes p_2$*;*
- $\sigma \in \Sigma_1 \cap \Sigma_2$ *and there exists* $(l_1, g_1, \sigma, p_1) \in \mathit{prob}_1$ *and* $(l_2, g_2, \sigma, p_2) \in \mathit{prob}_2$ *such that* $g = g_1 \wedge g_2$ *and* $p = p_1 \otimes p_2$

where for any $l_1 \in L_1$*,* $l_2 \in L_2$*,* $X_1 \subseteq \mathcal{X}_1$ *and* $X_2 \subseteq \mathcal{X}_2$*:* $p_1 \otimes p_2(X_1 \cup X_2, (l_1, l_2)) = p_1(X_1, l_1) \cdot p_2(X_2, l_2)$*.*

3.2 Semantics of probabilistic timed automata

Probabilistic systems. The semantics of probabilistic timed automata is defined in terms of transition systems exhibiting both nondeterministic and probabilistic choice. We call such models *probabilistic systems*; they are essentially equivalent to Markov decision processes [9], simple probabilistic automata [23], and probabilistic-nondeterministic systems [5].

Definition 3. *A* probabilistic system $\mathsf{PS} = (S, \bar{s}, \mathit{Act}, \mathit{Steps})$ *consists of a set* S *of* states*, an* initial state $\bar{s} \in S$*, a set* Act *of* actions*, and a* probabilistic transition relation $\mathit{Steps} \subseteq S \times \mathit{Act} \times \mathsf{Dist}(S)$*.*

A *probabilistic transition* $s \xrightarrow{a,\mu} s'$ is made from a state $s \in S$ by first nondeterministically selecting an action-distribution pair (a, μ) such that $(s, a, \mu) \in \mathit{Steps}$, and second by making a probabilistic choice of target state s' according to the distribution μ, such that $\mu(s') > 0$. We refer to probabilistic transitions of the form $s \xrightarrow{a,\mu_{s'}} s'$ (recall $\mu_{s'}(s') = 1$) as *transitions.* A *transition system* $\mathsf{TS} = (S, \bar{s}, \mathit{Act}, \mathit{Steps})$ is a probabilistic system for which every probabilistic transition is a transition.

We consider two ways in which a probabilistic system's computation may be represented. A *path* represents a particular resolution of both nondeterminism *and* probability. Formally, a path of a probabilistic system is a non-empty finite or infinite sequence of probabilistic transitions $\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$ such that

$s_0 = \bar{s}$. We denote by $\omega(i)$ the $(i+1)$th state of ω and $\mathit{last}(\omega)$ the last state of ω if ω is finite. On the other hand, an *adversary* represents a particular resolution of nondeterminism *only*. Formally, an adversary of a probabilistic system is a function A mapping every finite path ω to a pair (a, μ) such that $(\mathit{last}(\omega), a, \mu) \in \mathit{Steps}$ [27]. Let $\mathit{Adv}_{\mathsf{PS}}$ be the set of adversaries of PS. For any $A \in \mathit{Adv}_{\mathsf{PS}}$, let $\mathit{Path}^A_{\mathit{ful}}$ denote the set of infinite paths associated with A. Then, we define the probability measure Prob^A over $\mathit{Path}^A_{\mathit{ful}}$ according to classical techniques [14].

The *maximal (minimal) reachability probability* is the maximum (minimum) probability with which a given set of states can be reached from the initial state. Formally, for a probabilistic system $\mathsf{PS} = (S, \bar{s}, \mathit{Act}, \mathit{Steps})$, set $F \subseteq S$ of target states, and adversary $A \in \mathit{Adv}_{\mathsf{PS}}$, let:

$$\mathit{ProbReach}^A(F) \stackrel{\text{def}}{=} \mathit{Prob}^A\{\omega \in \mathit{Path}^A_{\mathit{ful}} \mid \exists i \in \mathbb{N} \,.\, \omega(i) \in F\} \ .$$

Then the maximal and minimal reachability probabilities $\mathit{MaxProbReach}_{\mathsf{PS}}(F)$ and $\mathit{MinProbReach}_{\mathsf{PS}}(F)$, respectively, are defined as follows:

$$\begin{aligned} \mathit{MaxProbReach}_{\mathsf{PS}}(F) &\stackrel{\text{def}}{=} \sup\nolimits_{A \in \mathit{Adv}_{\mathsf{PS}}} \mathit{ProbReach}^A(F) \\ \mathit{MinProbReach}_{\mathsf{PS}}(F) &\stackrel{\text{def}}{=} \inf\nolimits_{A \in \mathit{Adv}_{\mathsf{PS}}} \mathit{ProbReach}^A(F) \ . \end{aligned}$$

Semantics of probabilistic timed automata. We now give the semantics of probabilistic timed automata defined in terms of probabilistic systems. Observe that the definition is parameterized both by the time domain $\mathbb{T}$ and the time increment $\oplus$.

Definition 4. *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ *be a probabilistic timed automaton. The* semantics *of* PTA *with respect to the time domain* $\mathbb{T}$ *and the time increment* $\oplus$ *is the probabilistic system* $[\![\mathsf{PTA}]\!]^{\oplus}_{\mathbb{T}} = (S, \bar{s}, \mathit{Act}, \mathit{Steps})$ *where:* $S \subseteq L \times \mathbb{T}^{|\mathcal{X}|}$ *and* $(l, v) \in S$ *if and only if* $v \triangleleft \mathit{inv}(l)$*;* $\bar{s} = (\bar{l}, \mathbf{0})$*;* $\mathit{Act} = \mathbb{T} \cup \Sigma$*; and* $((l, v), a, \mu) \in \mathit{Steps}$ *if and only if one of the following conditions holds:*

Time transitions. $a \in \mathbb{T}$ *and* $\mu = \mu_{(l, v \oplus t)}$ *such that:*

1. $v \oplus t' \triangleleft \mathit{inv}(l)$ *for all* $0 \leq t' \leq t$*, and,*
2. *for all probabilistic edges of the form* $(l, g, \sigma, -) \in \mathit{prob}$*, if* $v \triangleleft g$*, then* $\sigma \notin \Sigma_u$*;*

Discrete transitions. $a \in \Sigma$ *and there exists* $(l, g, \sigma, p) \in \mathit{prob}$ *such that* $v \triangleleft g$ *and for any* $(l', v') \in S$*:*

$$\mu(l', v') = \sum_{\substack{X \subseteq \mathcal{X} \ \& \\ v' = v[X := 0]}} p(X, l') \ .$$

The summation in the definition of discrete transitions is required for the cases in which multiple clock resets result in the same target state (l', v'). Note that the semantics of timed automata is given in terms of transition systems.

In our setting, the semantics falls into two classes, depending on whether the underlying model of time is the positive reals or the naturals. If $\mathbb{T} = \mathbb{R}$ we let

$\oplus$ equal $+$ and refer to $[\![\mathsf{PTA}]\!]^{+}_{\mathbb{R}}$ as the *continuous semantics* of the probabilistic timed automaton PTA. In contrast, if $\mathbb{T} = \mathbb{N}$, we let $\oplus$ equal $\oplus_{\mathbb{N}}$ which is defined below and refer to $[\![\mathsf{PTA}]\!]^{\oplus_{\mathbb{N}}}_{\mathbb{N}}$ as the *integer semantics* of PTA. Let PTA be a probabilistic automaton; for any $x \in \mathcal{X}$, let $\mathbf{k}_x$ denote the greatest constant the clock x is compared to in the zones of PTA. Then, for any clock valuation $v \in \mathbb{N}^{|\mathcal{X}|}$ and time duration $t \in \mathbb{N}$, let $v \oplus_{\mathbb{N}} t$ be the clock valuation of $\mathcal{X}$ which assigns the value $\min\{v_x + t, \mathbf{k}_x + 1\}$ to all clocks $x \in \mathcal{X}$ (although the operator $\oplus_{\mathbb{N}}$ is dependent on PTA, we elide a sub- or superscript indicating this for clarity).

Note that the definition of integer semantics for probabilistic timed automata is a generalization of the analogous definition for the classical model in [3]. As we henceforth use the same type of time increment for a particular choice of time domain, we omit the $+$ and $\oplus_{\mathbb{N}}$ superscripts from the notation. The fact that the integer semantics of a probabilistic timed automaton is finite, and the continuous semantics of probabilistic timed automaton is generally uncountable, can be derived from the definitions. As noted by [18], the semantics of the parallel composition of two probabilistic timed automata corresponds to the semantics of the parallel composition of their individual semantic probabilistic systems, for both the continuous and integer semantics.

The following theorem is key to establishing the correctness of the integer semantics with regard to the probability of reaching a certain set of target locations of a probabilistic timed automaton. The theorem[3] states that both the maximal and minimal probabilities of reaching a target location are equal in the continuous and integer semantics, and is a probabilistic extension of a similar result established in [3]. The proof of the theorem appears in [19]. Let $L' \subseteq L$ be a set of target locations of a probabilistic timed automaton PTA, and let the set of all states in $[\![\mathsf{PTA}]\!]_{\mathbb{T}}$ corresponding to locations in L' be denoted by $F^{L'}_{\mathbb{T}} = \bigcup\{(l, v) \mid l \in L',\ v \in \mathbb{T}^{|\mathcal{X}|}\ \wedge\ v \triangleleft \mathit{inv}(l)\}$.

Theorem 1. *For every probabilistic timed automata* PTA *and target set* $L' \subseteq L$ *of locations:*

$$\mathit{MaxProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}(F^{L'}_{\mathbb{R}}) = \mathit{MaxProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}(F^{L'}_{\mathbb{N}})$$
$$\mathit{MinProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}(F^{L'}_{\mathbb{R}}) = \mathit{MinProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}(F^{L'}_{\mathbb{N}}).$$

Traces and trace distributions. A *trace* of a transition system is a sequence of actions which is obtained from a path by projecting all information except the actions. The notion of (finite or infinite) traces can be lifted to paths of probabilistic systems in a natural manner; for example, the trace of the infinite path $s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$ is the infinite sequence $a_0 a_1 \cdots$. Let A be an adversary of the probabilistic system $\mathsf{PS} = (S, \bar{s}, \mathit{Act}, \mathit{Steps})$, and let $f : \mathit{Path}^{A}_{\mathit{ful}} \to \mathit{Act}^{\omega}$ be a function assigning the trace to each infinite path of A. The *trace distribution* [22] of A is a probability measure over traces characterized by $f(\mathit{Prob}^{A})$. The set of trace distributions of PS, denoted by $\mathsf{tdist}(\mathsf{PS}) \subseteq \mathit{Act}^{\omega} \to [0, 1]$, comprises

[3] As in the non-probabilistic case [3], the theorem relies on the fact that only closed zones are used within the description of the probabilistic timed automaton.

the trace distributions corresponding to all of the adversaries of PS. Given two probabilistic systems PS_1, PS_2, we say that PS_1 *trace distribution refines* PS_2, denoted by $\mathsf{PS}_1 \preceq_D \mathsf{PS}_2$, if and only if $\mathsf{tdist}(\mathsf{PS}_1) \subseteq \mathsf{tdist}(\mathsf{PS}_2)$.

We next establish a result that is used to bridge the divide between reasoning about sets of states, as done in reachability analysis, and reasoning about actions, as done in trace-theoretic approaches.

Lemma 1. *Let* $\mathsf{PS}_1 = (S_1, \bar{s}_1, \mathit{Act}_1, \mathit{Steps}_1)$ *and* $\mathsf{PS}_2 = (S_2, \bar{s}_2, \mathit{Act}_2, \mathit{Steps}_2)$ *be two probabilistic systems such that* $\mathsf{PS}_1 \preceq_D \mathsf{PS}_2$. *If both of the following conditions hold:*

1. $F_1 \subseteq S_1 \setminus \{\bar{s}_1\}$, $F_2 \subseteq S_2 \setminus \{\bar{s}_2\}$, *and*
2. *there exists* $\mathit{ReachActs} \subseteq \mathit{Act}_1 \cap \mathit{Act}_2$ *such that for any* $i \in \{1,2\}$, $s_i \in S_i \setminus F_i$ *and* $s'_i \in F_i$, *we have* $s_i \xrightarrow{a,\mu} s'_i$ *if and only if* $a \in \mathit{ReachActs}$,

then we have:

$$\mathit{MaxProbReach}_{\mathsf{PS}_1}(F_1) \leq \mathit{MaxProbReach}_{\mathsf{PS}_2}(F_2)$$
$$\mathit{MinProbReach}_{\mathsf{PS}_1}(F_1) \geq \mathit{MinProbReach}_{\mathsf{PS}_2}(F_2)\,.$$

We lift the notion of trace distributions, sets of trace distributions and trace distribution refinement from probabilistic systems to probabilistic timed automata. For example, for two probabilistic timed automata PTA_1 and PTA_2, we say that PTA_1 trace distribution refines PTA_2, written $\mathsf{PTA}_1 \preceq_D \mathsf{PTA}_2$, if and only if $[\![\mathsf{PTA}_1]\!]_{\mathbb{R}} \preceq_D [\![\mathsf{PTA}_2]\!]_{\mathbb{R}}$.

The analogue of trace distribution refinement for transition systems and timed automata is *trace refinement*. Given two transition systems TS_1, TS_2, we say that TS_1 *trace refines* TS_2, denoted by $\mathsf{TS}_1 \preceq_T \mathsf{TS}_2$, if the set of traces of TS_1 is included in the set of traces of TS_2. Furthermore, for two timed automata TA_1 and TA_2, we say that TA_1 trace refines TA_2, written $\mathsf{TA}_1 \preceq_T \mathsf{TA}_2$, if and only if $[\![\mathsf{TA}_1]\!]_{\mathbb{R}} \preceq_T [\![\mathsf{TA}_2]\!]_{\mathbb{R}}$.

In the presence of urgent events, trace refinement is not a precongruence [13]. As the timed ready simulation preorder proposed by Jensen et al. is too fine a notion to imply a refinement relation for our models of the IEEE 802.11 BA protocol, we instead drop the requirement of urgency on events; then the set of traces of a timed automaton with urgency is contained within the set of traces of the timed automaton obtained by *changing urgent actions to non-urgent actions*. Formally, for a timed automaton $\mathsf{TA} = (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$, let its *lazy timed automaton* $\mathsf{LTA} = (L, \bar{l}, \mathcal{X}, \Sigma', \mathit{inv}, \mathit{prob})$ be such that $\Sigma' = \Sigma$ but $\Sigma'_u = \emptyset$. This gives us the following result; observe that the converse of the lemma does not hold.

Lemma 2. *Let* TA_1 *and* TA_2 *be two timed automata such that* $\Sigma_{u,2} \subseteq \Sigma_{u,1}$, *and let* LTA_1 *and* LTA_2 *be their lazy timed automata counterparts, respectively. Then:*

$$\mathsf{LTA}_1 \preceq_T \mathsf{LTA}_2 \;\;\Rightarrow\;\; \mathsf{TA}_1 \preceq_T \mathsf{TA}_2\,.$$

Formerly probabilistic timed automata. For a probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$, its *formerly probabilistic timed automaton*, denoted by $\mathsf{FPTA} = (L, \bar{l}, \mathcal{X}, \Sigma', inv, prob')$, is the timed automaton which agrees with PTA on all of its elements apart from the event set Σ' and the edge relation $prob'$, which are defined in the following manner. For each probabilistic edge $e = (l, g, \sigma, p) \in prob$, we define the set Σ'_e of events comprising of tuples $\langle\langle l, g, \sigma, p, X, l' \rangle\rangle$ for each (X, l') such that $\mu(X, l') > 0$, and let $\Sigma' = \bigcup_{e \in prob} \Sigma'_e$. The edge relation $prob'$ is defined to be the smallest set such that, for each event $\langle\langle l, g, \sigma, p, X, l' \rangle\rangle \in \Sigma'$, there exists an edge $(l, g, \langle\langle l, g, \sigma, p, X, l' \rangle\rangle, p_{(X,l')}) \in prob'$. The following lemma states formally a property first introduced in [18].

Lemma 3. *Let* PTA_1 *and* PTA_2 *to be two probabilistic timed automata, and let* FPTA_1 *and* FPTA_2 *be their formerly probabilistic timed automata. If* $\mathsf{FPTA}_1 \preceq_T \mathsf{FPTA}_2$ *then* $\mathsf{PTA}_1 \preceq_D \mathsf{PTA}_2$.

The intuition underlying the proof is that traces of FPTA_1 can be assembled into sets which define an adversary A of PTA_1; from $\mathsf{FPTA}_1 \preceq_T \mathsf{FPTA}_2$, an adversary of PTA_2 with the *same* trace distribution of A can also be constructed.

In practice, we would not construct a formerly probabilistic timed automaton as outlined above, because even for relatively small models the construction process would be laborious when done by hand. First, note that it suffices to include information of the form $\langle\langle l, g, \sigma, p, X, l' \rangle\rangle$ only for probabilistic edges not made with probability 1; for all other probabilistic edges of the form $(l, g, \sigma, p_{(X,l')}) \in prob$, we include the edge $(l, g, \langle\langle \sigma \rangle\rangle, p_{(X,l')}) \in prob'$. Lemma 3 continues to hold in this case, because we can nevertheless assemble sets of paths of FPTA_1 and FPTA_2 that result in the same trace distributions.

Let $\mathsf{PTA}_{1\|2} = \mathsf{PTA}_1 \| \mathsf{PTA}_2$. Observe that, in general, $\mathsf{FPTA}_1 \| \mathsf{FPTA}_2 \neq \mathsf{FPTA}_{1\|2}$, because probabilistic edges which do *not* assign probability 1 to a single outcome may synchronize in $\mathsf{PTA}_{1\|2}$, creating new product distributions of the form $p \otimes p'$ which do not have an analogue in $\mathsf{FPTA}_1 \| \mathsf{FPTA}_2$. For our case study, this does not cause a problem, since all transitions which do not occur with probability 1 (the transitions for setting the values of backoff) do not synchronize.

4 Modelling and verification

4.1 Modelling using Uppaal

Detailed model. Our initial step in modelling the IEEE 802.11 BA mechanism was to design a detailed model intended to represent the behaviour of the protocol when there is a collision; that is, when two stations send data packets at the same time. The model **WLAN** consists of five components operating in parallel, namely **Send**$_1$, **Send**$_2$ (sending stations), **Dest**$_1$, **Dest**$_2$ (destination stations), and **Chan** (the channel). In the following, we assume familiarity with the conventions for the graphical representation of timed automata. Note that we use the parameters of the Frequency Hopping Spread Spectrum (FHSS) physical layer, with a transmission bit rate of 2Mbps for the data payload.

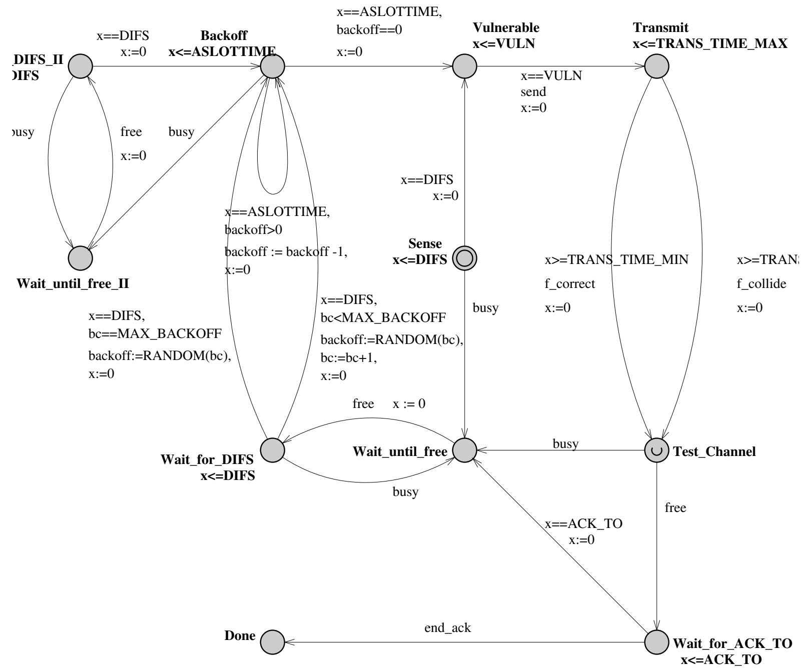

Fig. 2. Template for the sender stations.

The template for the senders is shown in Figure 2. Unless indicated otherwise, all transitions are made with probability 1. Note that the events busy and free are the urgent events of the sender. The initial location is indicated by the double circle. The sender begins with a data packet to send, and senses the channel. If the channel remains free for $\mathsf{DIFS} = 128\mu s$, then the sender enters its vulnerable period (explained in Section 2) and starts sending a packet, otherwise the station enters backoff. The time taken to send a packet is nondeterministic (within $\mathsf{TRANS\_TIME\_MIN} = 224\mu s$ and $\mathsf{TRANS\_TIME\_MAX} = 15,717\mu s$) and the success of the transmission – whether the event f_correct (successful) or f_collide (unsuccessful) is performed – depends on whether a collision has occurred and is recorded by the channel. The sender then immediately tests the channel (represented by the urgent location Test_Channel). If the channel is busy, the sender enters the backoff procedure, otherwise it waits for an acknowledgement. If the acknowledgement arrives within $\mathsf{ACK\_TO} = 300\mu s$, then the packet has been sent correctly and the sender finishes; otherwise it times-out and enters the backoff procedure. In the backoff procedure the sender first waits for the channel to be free for DIFS and then sets its backoff value according to the

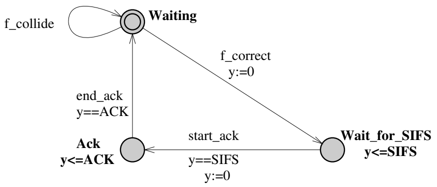

Fig. 3. Template for the destination stations.

random assignment[4] backoff := $RANDOM(bc)$, where bc, the backoff counter, is updated if its current value is less than its maximal value (MAX_BACKOFF = 6 since $aCWmax = 1023$). The sender then decrements backoff by 1 if the channel remains free for ASLOTTIME = $50\mu s$. However, if the channel is sensed busy within this slot, it waits until the channel becomes free and then waits for DIFS before resuming its backoff procedure. When the value of backoff reaches 0 the sender starts re-sending its data packet.

The template for the destinations is shown in Figure 3. Each destination waits for an incoming packet. If a packet arrives correctly (event f_correct), then the destination waits for SIFS = $28\mu s$ and subsequently sends the acknowledgement, which takes ACK = $205\mu s$ time units to send. On the other hand, if the message arrives garbled (event f_collide), the destination does nothing.

The probabilistic timed automaton **Chan**, which represents the channel, is shown in Figure 4. The location FREE corresponds to the case in which the channel is available. From this location, receipt of a data packet from station 1 (event send_1, sent by $\mathbf{Send}_1$) triggers the transition to location RCV1; then this packet can either finish successfully (event $\mathsf{f\_correct}_1$, sent by $\mathbf{Send}_1$ again) and return the channel to the location FREE, or collide with a transmission by station 2 (event send_2 sent by $\mathbf{Send}_2$) and make the channel proceed to RCV1RCV2. From the latter location only $\mathsf{f\_collide}_i$ events can remove the data packets from the channel. The left-hand side of the figure shows the part of the model used to represent the receipt of the acknowledgement on the channel. Note that the situations in which an acknowledgement is sent at the same time as a data packet, and in which two acknowledgements collide, are not modelled in this automaton. Although our original channel model did include locations to cater for this possibility, they were removed after reachability analysis of our model using the timed automata model checker UPPAAL [20] (with the FHSS timing parameters) established that such collisions are not possible.

Abstract model. Before constructing discrete-time models of our probabilistic timed automata, we first apply a number of abstractions. In particular, the

[4] A uniform choice over integers in the range $[0, (aCWmin + 1) \cdot 2^{bc} - 1]$ where $aCWmin = 15$.

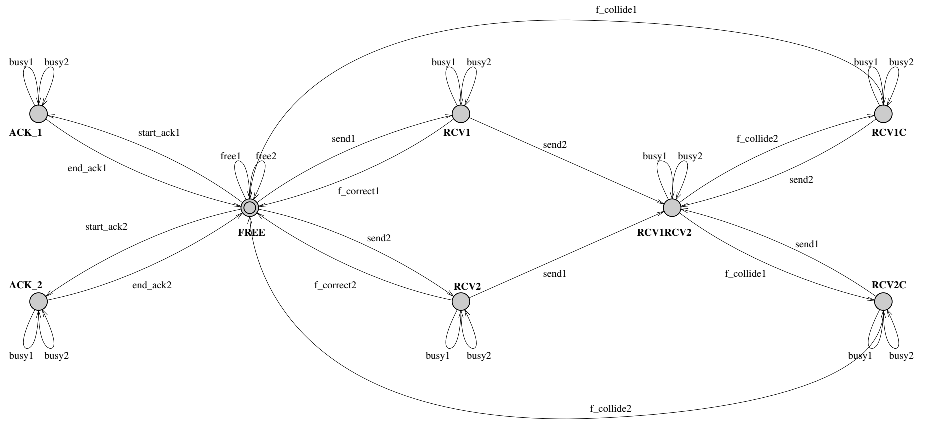

Fig. 4. Template for the channel.

destination, the behaviour of which is deterministic, is incorporated into the sender stations to obtain the probabilistic timed automata $\mathbf{AbsSend}_i$, for $i \in \{1,2\}$. Furthermore, using the result that acknowledgements can never collide, the same locations of the channel are used for the receipt of data packets and acknowledgements, resulting in the probabilistic timed automaton $\mathbf{AbsChan}$. Our aim is to verify the soundness of the above abstraction with respect to trace distribution refinement. That is, for the concrete and abstracted systems, denoted by

$$\begin{aligned} \mathbf{WLAN} &= \mathbf{Send}_1 \| \mathbf{Dest}_1 \| \mathbf{Chan} \| \mathbf{Dest}_2 \| \mathbf{Send}_2 \\ \mathbf{AbsWLAN} &= \mathbf{AbsSend}_1 \| \mathbf{AbsChan} \| \mathbf{AbsSend}_2 \end{aligned}$$

respectively, we show that $\mathbf{WLAN} \preceq_D \mathbf{AbsWLAN}$.

Our abstraction methodology is illustrated in Figure 5. We first construct lazy formerly probabilistic timed automata models for each sub-component of the detailed and abstract models using the methodology in Section 3, which are denoted by a prime. Then, using the timed automata model checker Uppaal, together with the methodology for testing trace refinement of timed automata presented in [24], we establish that $\mathbf{Send}'_i \| \mathbf{Dest}'_i \| \mathbf{Chan}' \preceq_T \mathbf{AbsSend}'_i$ for $i \in \{1,2\}$, and $\mathbf{Chan}' \preceq_T \mathbf{AbsChan}'$. This is illustrated in Figure 5 by the dashed lines from the sub-components of $\mathbf{WLAN}'$ to the sub-components of $\mathbf{AbsWLAN}'$ shown in the right-most box. As trace refinement is compositional, and as the set of traces of $(\mathsf{TA}_1 \| \mathsf{TA}_2) \| \mathsf{TA}_2$ equals that of $\mathsf{TA}_1 \| \mathsf{TA}_2$ for any timed automata TA_1, TA_2, we conclude that:

$$\mathbf{Send}'_1 \| \mathbf{Dest}'_1 \| \mathbf{Chan}' \| \mathbf{Dest}'_2 \| \mathbf{Send}'_2 \preceq_T \mathbf{AbsSend}'_1 \| \mathbf{AbsChan}' \| \mathbf{AbsSend}'_2 \ ,$$

as denoted by the dashed line from $\mathbf{WLAN}'$ to $\mathbf{AbsWLAN}'$ in the central box of Figure 5. Hence, using Lemma 2 and Lemma 3, it follows that:

$$\mathbf{Send}_1 \| \mathbf{Dest}_1 \| \mathbf{Chan} \| \mathbf{Dest}_2 \| \mathbf{Send}_2 \preceq_D \mathbf{AbsSend}_1 \| \mathbf{AbsChan} \| \mathbf{AbsSend}_2 \ .$$

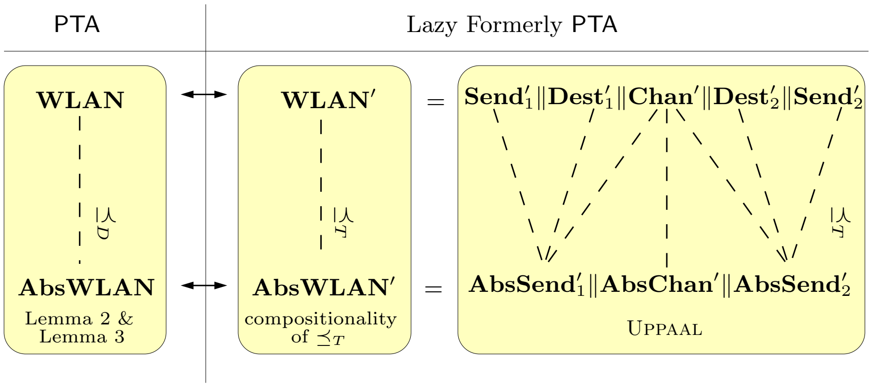
PTA
Lazy Formerly PTA
WLAN
$\preceq_D$
AbsWLAN
Lemma 2 & Lemma 3
WLAN′
$\preceq_T$
AbsWLAN′
compositionality of $\preceq_T$
=
$\mathbf{Send}'_1\|\mathbf{Dest}'_1\|\mathbf{Chan}'\|\mathbf{Dest}'_2\|\mathbf{Send}'_2$
$\preceq_T$
=
$\mathbf{AbsSend}'_1\|\mathbf{AbsChan}'\|\mathbf{AbsSend}'_2$
Uppaal

Fig. 5. A diagram illustrating the abstraction procedure.

Thus, $\mathbf{WLAN} \preceq_D \mathbf{AbsWLAN}$ as required, as shown in Figure 5 by the dashed line from **WLAN** to **AbsWLAN** in the left-most box.

Note that, as the probabilistic choice involved in setting the backoff value involves 1024 possibilities, instead of constructing up to 1024 edges in the formerly probabilistic timed automata $\mathbf{Send}'_i$ and $\mathbf{AbsSend}'_i$, we model the choice of $\mathsf{backoff}_i$ using a "widget": a sub-automaton that determines the range of the choice of $\mathsf{backoff}_i$, sets $\mathsf{backoff}_i$ to the maximal value of the range, and then permits a nondeterministic choice between decrementing the value of $\mathsf{backoff}_i$ or proceeding to the location Backoff. The transitions of the widget are labelled appropriately to ensure correspondence between the values of $\mathsf{backoff}_i$ chosen by the formerly probabilistic timed automata $\mathbf{Send}'_i$ and $\mathbf{AbsSend}'_i$. Naturally, no time is permitted to elapse while in the widget.

A final abstraction step comprises an adjustment to the *time scale* of the model, in anticipation of analysis using probabilistic model checking in Prism. Thus far, we have assumed that a time unit is equal to 1μs; however, this leads to the model having a largest constant of 15,717, which results in a prohibitively large model size when using the discrete-time semantics introduced in Section 3.2. Therefore, we choose a new time unit of $\mathsf{ASLOTTIME} = 50\mu$s, rounding upper bounds on the values of the clocks up, lower bounds down, and let $\mathbf{AbsWLAN_t}$ be the resulting probabilistic timed automaton, the largest constant of which is now 315. For a timed automaton TA, it is established in [2] that the trace set of a timed automaton $\mathsf{TA_t}$ after such a transformation includes that of the original model, denoted by $\mathsf{TA} \preceq_T \mathsf{TA_t}$. Therefore, we have $\mathbf{AbsWLAN}' \preceq_T \mathbf{AbsWLAN_t}'$, and, by Lemma 3, also that $\mathbf{AbsWLAN} \preceq_D \mathbf{AbsWLAN_t}$. By the transitivity of $\preceq_D$, we know that **WLAN** is a trace distribution refinement of our final abstract model $\mathbf{AbsWLAN_t}$.

k	Iterations	Time per iteration (s)	Probability
1	18	0.052	1
2	89	0.206	0.183593
3	208	0.366	0.017032
4	423	0.549	7.942e-4
5	816	0.819	1.85e-5
6	1,571	1.32	2.17e-7

Table 1. Maximum probability of either station's backoff counter reaching k.

4.2 Verification using PRISM

In this section, we use the probabilistic model checking tool PRISM to automatically verify the satisfaction of probabilistic reachability properties. The model that we build with PRISM is the discrete-time semantic model of the final abstraction given in the previous section ($[\![\mathbf{AbsWLAN_t}]\!]_{\mathbb{N}}$). Note that, for all the properties considered, the correspondence between **WLAN** and **AbsWLAN** required by Lemma 1 holds, and hence the results obtained for the abstract model are upper (lower) bounds for maximal (minimal) reachability probabilities of the full model.

Due to the size of the models, all experiments were performed with PRISM's most space efficient model checking engine, which uses Multi-Terminal Binary Decision Diagrams (MTBDDs) [6]. We ran all experiments on a 440 MHz SUN Ultra 10 workstation with 512 MB memory under the Solaris 2.7 operating system. All properties were checked with an accuracy of $\varepsilon = 10^{-6}$. Further details on the PRISM model and the model checking results are available from the PRISM web page [21].

The model took 72.1 seconds to construct and has 5,958,233 states. We then calculated the minimal probability of both stations eventually sending their packet correctly. As expected, this has probability 1, and hence the probability is also 1 for the detailed model. This calculation took 7,137 iterations and the time per iteration was 14.1 seconds. Furthermore, we calculated the maximum probability of either station's backoff counter reaching k, the results of which are presented in Table 1. Observe that the greater the value of the backoff counters, the greater the number of collisions, and hence the longer it takes for a data packet to be sent correctly. We observe that the probability of reaching k falls rapidly as k increases. Note that for $k = 1$ the probability is 1, which corresponds to the fact that we model the case where stations initially collide.

We then investigated soft deadline properties, namely calculating the *minimum probability of a station delivering a packet within some deadline*. In particular, we considered this property for a variety of deadlines and different values of $\mathsf{TRANS\_TIME\_MAX}$. For example, for the case when $\mathsf{TRANS\_TIME\_MAX} = 2,500$ and the deadline equals $10,000\mu$s, the model has 583,661,380 states, took 658 seconds to construct and the minimum probability of a station sending a

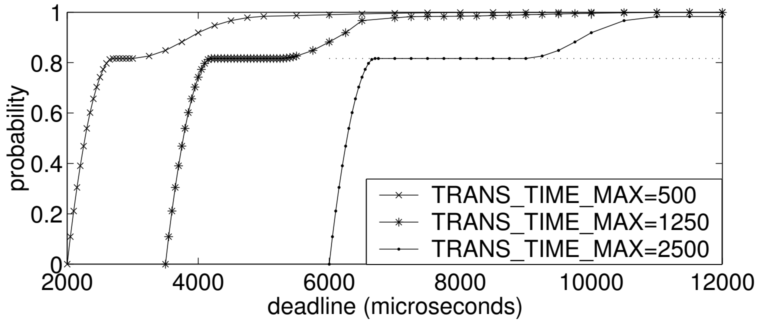

Fig. 6. A graph showing the results for the deadline properties.

packet within the deadline is 0.918914, where the verification procedure required 354 iterations and the time per iteration was 3.22 seconds.

In Figure 6 we have plotted the minimum probability of a packet being sent for different deadlines and values of TRANS_TIME_MAX. Note that since stations initially collide with probability 1, the probability will be zero for any deadline which does not allow the stations to collide, enter the backoff procedure and then resend their data packets. This can be seen in the graph by noting that the probability is zero for all deadlines less than or equal to $2 \cdot \mathsf{TRANS\_TIME\_MAX} + 1,000$, i.e. the time for a station to send its packet twice plus a constant which includes the time to wait for an acknowledgement and enter backoff.

The dotted line in the graph corresponds to the minimum probability of a station sending a packet correctly while not entering backoff more than once; the results below this line correspond to deadlines where only the first backoff procedure can influence the outcome (that is, for these deadlines there is insufficient time for a station to enter the backoff procedure more than once and send its data correctly). Furthermore, the portions of the graph where the probability does not increase correspond to deadlines which are not large enough for a station to enter backoff more than once and successfully send its data packet, but are sufficient for all cases when backoff is entered at most once. We note that, for each deadline and value of TRANS_TIME_MAX, the minimum probability corresponds to the case when the adversary chooses TRANS_TIME_MAX as the time it takes to send each data packet.

5 Conclusions

We have presented an application of probabilistic model checking to a subprotocol the IEEE 802.11 standard for WLANs, also using non-probabilistic analysis in a “proof assistant” role to verify the soundness of several abstractions of our original protocol model. The use of nondeterminism allows us to model asynchronous behaviour of stations, in addition to providing a conservative approximation mechanism when constructing smaller abstract models.

Future work could lift several simplifying assumptions that were made in this work, such as the fixed network topology in which sending stations cannot also be destination stations, and the absence of the Timing Synchronization Function. It is straightforward to increase the number of sending and destination stations in our framework, although naturally such verification attempts may suffer from the state-explosion problem commonly encountered in model checking.

We considered probabilistic reachability properties, which are counter-parts of *transient* properties of stochastic processes. It is intended to extend the range of properties that can be verified by the tool PRISM, to include for example expected-time properties [8].

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, A. Itai, R. P. Kurshan, and M. Yannakakis. Timing verification by successive approximation. *Information and Computation*, 18(1):142–157, 1995.
3. D. Beyer. Improvements in BDD-based reachability analysis of timed automata. In *Proc. FME'01*, volume 2021 of *LNCS*, pages 318–343. Springer, 2001.
4. G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communication*, 18:535–547, 2000.
5. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. FSTTCS'95*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.
6. E. M. Clarke, M. Fujita, P. McGeer, K. McMillan, J. Yang, and X. Zhao. Multi-Terminal Binary Decision Diagrams: An efficient data structure for matrix representation. In *Proc. IWLS'93*, pages 6a:1–15, 1993. Also available in *Formal Methods in System Design*, 10(2/3), 1997.
7. C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *Proc. RTSS'95*, pages 66–75. IEEE Computer Society Press, 1995.
8. L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proc. CONCUR'99*, volume 1664 of *LNCS*, pages 66–81. Springer, 1999.
9. C. Derman. *Finite-State Markovian Decision Processes*. Academic Press, 1970.
10. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
11. A. Heindl and R. German. Performance modeling of IEEE 802.11 wireless LANs with stochastic Petri nets. *Performance Evaluation*, 44:139–164, 2001.
12. T. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In *Proc. TACAS'95*, volume 1019 of *LNCS*, pages 41–71. Springer, 1995.
13. H. Jensen, K. Larsen, and A. Skou. Scaling up UPPAAL: Automatic verification of real-time systems using compositionality and abstraction. In *Proc. FTRTFT 2000*, volume 1926 of *LNCS*, pages 19–30. Springer, 2000.
14. J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains*. Graduate Texts in Mathematics. Springer, 2nd edition, 1976.
15. A. Koepsel, J.-P. Ebert, and A. Wolisz. A performance comparison of point and distributed coordination function of an IEEE 802.11 WLAN in the presence of real-time requirements. In *Proc. MoMuC 2000*, 2000.

16. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *Proc. TACAS 2002*, volume 2280 of *LNCS*, pages 52–66. Springer, 2002.
17. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 286, 2002. To appear.
18. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. Submitted. Extended abstract appears in *Proc. Int. Workshop on Application of Formal Methods to IEEE 1394 Standard*, 2001.
19. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. Technical Report CSR-02-05, School of Computer Science, University of Birmingham, 2002.
20. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Software Tools for Technology Transfer*, 1(1+2):134–152, 1997.
21. PRISM web page. `http://www.cs.bham.ac.uk/$\sim$dxp/prism/`.
22. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.
23. R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
24. D. Simons and M. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using UPPAAL2k. *Software Tools for Technology Transfer*, 3(4):469–485, 2001.
25. S. Tripakis. *The formal analysis of timed systems in practice*. PhD thesis, Université Joseph Fourier, 1998.
26. S. Tripakis. Timed diagnostics for reachability properties. In *Proc. TACAS'99*, volume 1579 of *LNCS*, pages 59–73. Springer, 1999.
27. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. FOCS'85*, pages 327–338. IEEE Computer Society Press, 1985.