Formal Methods for the Analysis of Wireless Network Protocols

Matthias Fruth
Trinity College

A thesis submitted
to the University of Oxford
for the degree of Doctor of Philosophy

Michaelmas Term 2011

Abstract

In this thesis, we present novel software technology for the analysis of wireless networks, an emerging area of computer science. To address the widely acknowledged lack of formal foundations in this field, probabilistic model checking, a formal method for verification and performance analysis, is used. Contrary to test and simulation, it systematically explores the full state space and therefore allows reasoning about all possible behaviours of a system.

This thesis contributes to design, modelling, and analysis of ad-hoc networks and randomised distributed coordination protocols.

First, we present a new hybrid approach that effectively combines probabilistic model checking and state-of-the-art models from the simulation community in order to improve the reliability of design and analysis of wireless sensor networks and their protocols. We describe algorithms for the automated generation of models for both analysis methods and their implementation in a tool.

Second, we study spatial properties of wireless sensor networks, mainly with respect to Quality of Service and energy properties.

Third, we investigate the contention resolution protocol of the networking standard ZigBee. We build a generic stochastic model for this protocol and analyse Quality of Service and energy properties of it. Furthermore, we assess the applicability of different interference models.

Fourth, we explore slot allocation protocols, which serve as a bandwidth allocation mechanism for ad-hoc networks. We build a generic model for this class of protocols, study real-world protocols, and optimise protocol parameters with respect to Quality of Service and energy constraints. We combine this with the novel formalisms for wireless communication and interference models, and finally we optimise local (node) and global (network) routing policies.

This is the first application of probabilistic model checking both to protocols of the ZigBee standard and protocols for slot allocation.

Acknowledgements

I would like to express my gratitude to everyone who supported me during the course of this work.

First and foremost, I am greatly indebted to my supervisor, Marta Kwiatkowska, who enabled this research by taking me on as a student, for her invaluable guidance and unerring support.

I would like to thank all my colleagues and collaborators: I am particularly grateful to David Parker and Gethin Norman, who commented on earlier drafts of this thesis. I would like to thank Annabelle McIver and Ansgar Fehnker, who invited me to stay at National ICT Australia and work with them. I am grateful to Christel Baier, who welcomed me to stay with her group while I was writing this thesis. I am grateful for the financial support received from the UbiVal project (EPSRC grant EP/D076625), from National ICT Australia, and from Trinity College.

I am grateful to Alice Miller and James Worrell for examining the thesis.

Many thanks to my friends, especially Christoph, David, Henry, Mette, and Olga, for making my time in Oxford more enjoyable.

Finally, I am indefinitely grateful to my family: to my parents, for providing my education and supporting me throughout, and most importantly to Heike, for all her support and love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent years, wireless networking has enjoyed a great and steadily growing popularity in both research and industry. However, the development of adequate formalisms for modelling and analysis of wireless networks has not kept pace with this, with Chalmers et al. in the Grand Challenges for Computing Research [21] stating a "considerable lack of formal foundations".

While simulation is the standard tool for analysing wireless network protocols, it suffers from a number of well-documented problems. First, the underlying mathematical models are usually opaque and unavailable to users. Furthermore, they are often unrealistic, and results can vary widely between different simulators and field experiments. Consequently, simulation results can depend as much on the simulator as on the design of the protocol [35]. Second, simulators generally do not adequately support computational behaviours such as nondeterminism, again leading to unrealistic results. Indeed, in some cases nondeterminism is treated probabilistically, implying incorrect calculations for any environment that does not comply with such assumptions. Third, the optimisation of a protocol requires a thorough comparison of different designs, for example, a sensitivity analysis in cases when system performance depends on the choice of system parameters. With simulation, such comparative studies require the statistical interpretation of an expensively large number of simulation runs, making the analysis difficult and costly.

Applications of formal methods for the analysis of computer networks are usually moti-

vated by a desire to study them more thoroughly, that is, to increase breadth, depth, and reliability of the analyses, thus increasing coverage, feasible system complexity, and depth of the results. There have been several successful applications of probabilistic model checking to wireless networks. However, its feasibility is typically limited to relatively small networks and an accurate representation of the physical features of wireless communication has usually been neglected.

Given the recent availability of advanced and experimentally validated analytical models for wireless communication (most prominently the lognormal shadowing model by Zuniga and Krishnamachari [102]), we aim to improve accuracy and scope of existing analysis approaches for wireless network protocols by combining formal methods with models from the simulation community.

1.2 Related work

This section reviews previous and current literature of the scientific field relevant to this thesis. Section 1.2.1 introduces existing approaches for modelling and analysis of wireless networks. Section 1.2.2 presents approaches combining formal methods and simulation. Section 1.2.3 covers the analysis of different quantitative aspects of wireless communication. Finally, Section 1.2.4 summarises work on randomised protocols for wireless networks.

1.2.1 Modelling and analysis of wireless networks

Traditionally, the main method for analysing wireless networks is simulation. While this is still the case, the growing maturity and prevalence of formal methods has led to their application to various aspects of wireless networks.

Simulation-based approaches There are several established tools for the simulation of wireless networks, including *ns-2* [76], *OPNeT Modeler* [77] and *GloMoSim* [68]. Many case studies demonstrate the use of simulation methods, for example, [53, 73, 75, 90].

There has been broad criticism of wireless-network simulation approaches and tools, showing various inadequacies that lead to unrealistic results. Considering a simple generic flooding algorithm modelled using three established wireless-network simulators, Cavin et al. [20]

demonstrate that analysis results vary widely between different simulators and also compared to field experiments. They attribute this to a lack of transparency and accuracy in the underlying mathematical models. Similar observations are made by Kotz et al. [52], who criticise the use of oversimplified assumptions in the radio models of many simulators, a lack of empirical validation, and incorrect abstractions of network layers.

Recently, however, the quality of wireless communication models for simulation has been improving, owing to the development of empirically validated channel and radio models. Such models have been proposed, amongst others by Zuniga and Krishnamachari [102] and by Seada et al. [86]; the simulator *Castalia* [19], developed since 2006, is based on these models. Pham et al. [79] experimentally validate the wireless models implemented in Castalia, using real wireless sensor nodes based on ZigBee transceiver devices.

Formal methods-based approaches Model checking is an algorithmic approach to exhaustively and automatically establish system properties. Probabilistic model checking, in particular the probabilistic model checker *PRISM* [60, 80], has been successfully employed for the verification of various network protocols. These include: IEEE 802.11 WLAN contention resolution [64], IEEE 1394 FireWire root contention [65], and Bluetooth device discovery [30]. Kwiatkowska et al. [60] give an overview about different types of probabilistic temporal-logic properties of wireless network protocols that can be analysed using PRISM.

Demaille et al. [25] use the approximate verification tool *APMC* [22] to analyse a wireless sensor network for intrusion detection. They consider LTL properties for discrete-time probabilistic models of 100 and 400 nodes. Kwon and Agha [67] use very simple *Discrete-time Markov chain (DTMC)*[1] models, consisting of three states and one probabilistic transition per state, to analyse large wireless sensor networks. Using a statistical abstraction technique, they generate a model of 3,000 identical DTMCs. Discrete-event simulation and experiments with 90 real sensor nodes are used to statistically validate the model. Using the iLTL model checker, which has been developed by the authors, temporal-logic properties such as system availability and energy consumption are computed.

Sharma et al. [87] have modelled and analysed the synchronisation mechanism of the Wireless Sensor Network (WSN) implementation language Insense. In this language, which

[1] DTMCs are introduced in Section 2.1.2.

is modelled in Promela, the input language of the model checker SPIN, WSN applications are described using two high-level constructs: *components*, which describe single threads, and *channels*, which describe the communication between components. Hardware access is realised via a C library. Using SPIN, a range of LTL properties was verified.

Heidemann et al. [45] mentions that finding the right level of detail for abstractions of wireless networks is difficult even for networking specialists; they suggest the use of visualisation techniques. McIver [70] propose an abstraction and refinement approach for the modelling of wireless networks, based on probabilistic action systems. McIver and Fehnker [71] mention that simulators are predominant but results usually "have not been validated against empirical data"; they use probabilistic action systems to describe abstractions made for an analysis of a flooding protocol. Schuts et al. [85] study a wireless sensor protocol (clock synchronisation algorithm in gMAC) using timed automata and UPPAAL. All analyses were feasible only for an example network of at most three nodes. The authors conclude that more realistic wireless models and efficient probabilistic abstractions should be used.

1.2.2 Combining network simulation and probabilistic model checking

In the literature, there is little evidence of intertwinings of network simulation and probabilistic model checking. Fehnker and Gao [35] use the probabilistic model checker PRISM to analyse a simple gossip protocol. In order to obtain more realistic results than with current approaches, formal analysis with DTMC and *Markov decision process (MDP)*[2] models is supplemented by Monte-Carlo simulation using a MATLAB implementation, which validates the obtained results for larger models. The authors propose to build a formal model and a simulation model from one uniform description.

To the best of our knowledge, CaVi [16] is the only tool that provides a uniform interface for formal analysis and simulation of wireless sensor networks and it is the only tool that incorporates realistic wireless channel or radio models into model checking.

Seada et al. [86] mention the problem of finding sensible topologies when designing wireless sensor networks, that is, adjusting channel and radio parameters values in order to achieve good reception probabilities.

[2]MDPs are introduced in Section 2.1.3.

1.2.3 Analysing quantitative aspects of wireless communication

Spatial characteristics Spatial properties of wireless networks have been considered in a number of studies. Fehnker et al. [36] perform an exhaustive analysis of a class of possible topologies of the LMAC protocol (a specific Medium Access Control (MAC) protocol) for wireless sensor networks; they use timed automata models with the tool UPPAAL. Yue et al. [98] analyse the gMAC protocol, considering static and mobile scenarios with different initial configurations; they use the discrete-event simulator of the Möbius tool suite.

Interference Most models of wireless network communication use simplified interference assumptions such as "when two stations send at the same time, a collision always occurs and both messages get corrupted" or "collisions never happen". Kotz et al. [52] criticise the lack of realistic interference models. More realistic analytical models for interference have been proposed, for example by Zuniga and Krishnamachari [102].

Energy Seada et al. [86] analyse energy consumption of geographic routing protocol. Kwon and Agha [67] investigate energy consumption of a wireless sensor network for intrusion detection. Both studies use analytical formulae and discrete-event simulation.

Gross et al. [41] study energy properties for IEEE 802.15.4 wireless sensor networks, using discrete-event simulation of probabilistic timed automata with the tool Möbius.

Yue et al. [99] study energy consumption for a randomised leader election protocol. They consider a simple MDP model for interference (consisting of two states only), which they analyse using PRISM. Analysis results show that prioritising stations with higher power level reduces the overall energy consumption but also changes the respective stations' probability of being elected leader. Yue et al. [98] study the energy consumption per message propagation for the proprietary gMAC protocol, using discrete-event simulation.

1.2.4 Analysing randomised protocols for wireless networks

Gossip protocols The literature reports on several applications of formal methods to gossip protocols, a choice of which is discussed here. Cardell-Oliver [18] analyse performance properties of a simple, generic flooding protocol. Using a timed automata model with a simulator, they evaluate different topologies (parameterising density and noisiness) with respect

to transmission probability and expected number of rounds. Fehnker and Gao [35] also study simple gossip protocols.

Using UPPAAL and PRISM, McIver and Fehnker [71] analyse simple performance properties such as "flooding is unreliable" and minimum/maximum expected delay.

Kwiatkowska et al. [55] study a gossip protocol called *peer sampling* using probabilistic model checking and discrete-event simulation. Using MDP models of small networks (at most four nodes), they analyse properties such as the expected number of rounds to form a connected network, the expected path length between nodes, and the respective schedulings that correspond to these values.

Bakhshi et al. [13] analyse a gossip protocol for data sharing in a distributed network (the "shuffling algorithm"). They compare results from real experiments, simulation, and an analytical model based on differential equations. In order to allow a more efficient analysis of certain protocol behaviours, abstractions for transition probabilities are computed. Relationships between system parameters are analysed and parameter values are optimised.

Bakhshi and Fehnker [12] model this protocol using PRISM, MATLAB, and PeerSim for separate models, using DTMC and MDP models. They consider different scheduling policies, a coverage property ("the number of nodes that have 'seen' a given item over time"), and a notion of rounds. By comparing PRISM with non-formal models, they identified 'hidden' assumptions in those frameworks, for example that fractions were equivalent to probabilities.

Bakhshi et al. [11] analyses a different gossip protocol, the "gossiping time protocol", using mean-field approximation, a technique that exploits symmetries in large stochastic processes by replacing it with smaller deterministic processes. For DTMC models, they adjust protocol parameters and analyse performance measures for whole model.

Contention resolution protocols Kwak et al. [53] analyse performance aspects of the IEEE 802.15.4 CSMA-CA contention resolution protocol using an analytical model and an implementation of a simulator.

Tschirner et al. [97] study performance properties of body sensor networks, using UPPAAL for model checking as well as simulation of timed automata models (for 15 nodes) and the Castalia wireless-network simulator for validation. For a specific IEEE 802.15.4/ZigBee transceiver device, they aim to adjust timing parameters such that a given set of QoS

properties, mainly concerning multi-hop reception, is satisfied. The authors conclude that these properties are satisfied for a model with adjusted parameter values, accepting that an unrealistic channel model is used.

Zayani et al. [101] study a backoff procedure for the academic wireless sensor network protocol ECo-MAC. They analyse the protocol using PRISM with DTMC models, and validate this analysis using the OPNeT simulator. Safety properties and the effect of the number of neighbours and the contention window length were analysed, but no models or reward definitions are given.

Slot allocation protocols Rana et al. [82] provide an overview of hybrid slot allocation protocols, that is, protocols combining scheduled and random access.

Altman et al. [4] discuss different policies for a distributed choice of retransmission probabilities in the slotted variant of the MAC protocol *ALOHA*.

Kim and Park [51] propose a new MAC protocol called TC-MAC that is aimed to be more energy-efficient than previous MAC protocols for WSNs, namely S-MAC, which combines contention-based and time-based medium access; different variants of TC-MAC, and, for comparison, S-MAC, are compared using analytical formulae and the simulator ns-2. Horng and Kuo [48] study another new slot allocation protocol, a variation of TDMAM; they use analytical formulae as well as simulation. El Khoury and El-Azouzu [31] study forwarding in a multi-hop wireless network, with respect to topology but under a simplified interference model; they use analytical expressions and discrete-time simulation. Busse et al. [17] analyse forwarding strategies for wireless sensor networks; analytical expressions and results for throughput and energy consumption are obtained with a simulator.

Yue et al. [98] study energy consumption for the proprietary *gossiping MAC protocol (gMAC)*. The protocol includes a fully decentralised slot allocation algorithm. For static and simple mobility scenarios (rotating a single fixed row in the grid by one position), different grid positions of nodes sending initially, and a simple interference model with fixed communication range are assumed (not a realistic wireless model). This is analysed using the discrete-event simulator from the Möbius tool suite. The analysis results reveal that the randomised sending strategies are optimal with respect to a certain energy-per-propagation measure. The mobility results are limited.

1.3 Summary

The presentation of related work has shown several past and current research activities relevant for modelling and analysis of wireless network protocols. They use both simulation-based and formal methods-based approaches; however, we are not aware of any effective combinations interfacing the two. In particular, realistic models of physical characteristics of wireless communication such as noise, interference, and spatial characteristics are missing.

Although the computation of manual abstractions is common whenever formal methods are applied, we are not aware of any graphical modelling approach specifically for wireless networks, particularly one that supports the automatic computation of probabilistic abstractions and the automatic generation of models suitable for probabilistic model checking.

While there have been formal analyses of wireless communication that cover Quality of Service properties such as throughput and energy consumption, other aspects such as network topology and colliding transmissions have not been covered due to the lack of adequate models.

Of the protocols considered in this thesis, there have already been several formal analyses of gossip protocols; there have been formal analyses of contention resolution protocols, but never for IEEE 802.15.4 or ZigBee; there have been some formal analyses of slot allocation protocols, but never for dynamic slot allocation.

1.4 Contributions

The main contributions of this thesis are:

- demonstration of the feasibility of applying probabilistic model checking to wireless local area networks, wireless sensor networks, and randomised distributed coordination protocols;
- first proposal for a hybrid framework for simulation and probabilistic model checking of wireless sensor networks, using wireless communication models from the simulation community;
- first application of formal methods/probabilistic model checking to the IEEE 802.15.4/ZigBee networking standard;

- first study of interference models using formal methods/probabilistic model checking; and

- first application of formal methods/probabilistic model checking to slot allocation protocols.

Throughout this thesis, we have investigated effective and efficient combinations of probabilistic model checking and discrete-event simulation for wireless networks, particularly for randomised coordination protocols of wireless networks. We have incorporated realistic mathematical models for interference in wireless communication that allow high scalability and accuracy. For example, complex protocols are supported in full detail for arbitrary networks of competitive sizes.

Our formal analysis of the IEEE 802.15.4 contention resolution protocol is the first application of formal methods to this networking standard. As part of this work, we have compared interference models commonly used in formal analysis with novel interference models based on lognormal shadowing. We have modelled the full extent of this protocol, thereby developing several abstractions and optimisations, and evaluated various qualitative and quantitative performance properties. It has been demonstrated that our hybrid analysis framework is superior to hitherto existing approaches.

Our formal analysis of slot allocation protocols is the first application of formal methods to this class of protocols. It has been demonstrated how optimal parameter values can systematically be determined from a large parameter space. Furthermore, it has been shown how results from our hybrid analysis framework can be used to support network design decisions.

1.5 Joint work

This thesis is based on several peer-reviewed publications of the thesis' author and his collaborators.

Chapter 3 presents the first hybrid modelling and analysis approach that combines probabilistic model checking and simulation. This work has been carried out jointly with Annabelle McIver, Ansgar Fehnker, and Athanassios Boulis. Parts of it have been presented and pub-

lished at NICTA Techfest 2007 [32], at MeMoT 2007[3] [33], at QEST 2008[4] [16], and as a book chapter in the Lecture Notes in Computer Science series [34]. The work started with the thesis author's three-month visit to NICTA. The papers [16] and [32] were written by Ansgar Fehnker; the first one mainly reports on results from work conducted during the thesis author's visit; the second one also includes newer features of the tool, which are not described in this thesis. The papers [33] and [34] were written by Annabelle McIver, based on extensive email discussions between the thesis' author, Ansgar Fehnker, and her; for both papers, the thesis' author participated in discussions, presented ideas, and contributed changes and additions to the text. The text of this chapter has been written by the thesis' author, but is based on the joint publications listed above. Contributions of the thesis' author to this work include:

- contribution to design and implementation of the CaVi modelling and analysis framework;
- design and initial implementation of CaVi's models for networks, nodes, and inter-node relationships;
- initial design and initial implementation of the computation and visualisation engine; and
- design and initial implementation of the generator for parameterised Castalia experiments from CaVi networks;
- evaluation of approaches and contribution to the design of the generator for PRISM models from protocol-specific templates;
- contribution to case studies (flooding/gossip protocols under different interference scenarios).

That is, he designed and implemented most parts of the tool, developed an efficient mechanism for generating Castalia experiments for CaVi networks, and – jointly with his collaborators – developed an efficient mechanism for generating PRISM models for flooding/gossip protocols and different interference scenarios.

[3]MeMoT 2007 stands for the Workshop on Methods, Models and Tools for Fault Tolerance at the 7th International Conference on Integrated Formal Methods (IFM 2007).

[4]QEST 2008 stands for the 5th International Conference on the Quantitative Evaluation of Systems.

Chapter 5 presents the first application of probabilistic model checking to the standards IEEE 802.15.4 and ZigBee. Parts of this work have been presented and published at ISoLA 2006[5] [39], and a summary has appeared on the PRISM website [80]. In subsequent work, the performance of ZigBee security key updates was also analysed using PRISM [100].

Chapter 6 presents the first application of probabilistic model checking to dynamic slot allocation protocols for wireless sensor networks. This work was initially carried out jointly with Annabelle McIver. At the beginning of this study, she formulated a simpler version of the slot allocation problem, modelled it in PRISM, and wrote a preliminary three-page draft about this. After that, the thesis' author continued the study on his own. The motivation and the verbal formulation of the slot allocation problem, which have inspired the introduction to Chapter 6, are based on joint work. All other contributions presented in Chapter 6 solely belong to the thesis' author, including:

- the review of related work;
- the models, including all abstractions used;
- the experiments, especially the consideration of locality and energy properties; and
- the interpretation of the results.

1.6 Structure

This thesis is divided into seven chapters and four appendices.

Chapter 2 introduces preliminary concepts, definitions, and terminology for the following chapters. Chapter 3 describes the hybrid approach that combines simulation models for wireless sensor networks with model checking and its tool implementation. Chapter 4 presents the semi-formal analysis of spatial properties of wireless sensor networks. Chapter 5 contains the formal analysis of the IEEE 802.15.4 contention resolution protocol. Chapter 6 describes the formal analysis of dynamic slot allocation protocols. Chapter 7 concludes this thesis. Appendices A, B, C, and D provide ancillary information for Chapters 3, 4, 5, and 6, respectively.

[5] ISoLA 2006 stands for the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation

Chapter 2

Preliminaries

In this chapter, background material for the following chapters is presented. The material is organised into different sections, on probabilistic model checking and discrete-event simulation.

2.1 Probabilistic model checking

In this section, we give a brief introduction to probabilistic model checking.

Probabilistic model checking [58, 59, 84] is a formal method for the automatic verification and quantitative analysis of probabilistic systems. The objective of a probabilistic model checking algorithm is, for a probabilistic model of a system (usually given as a variant of a Markov process) and a probabilistic temporal-logic property, to decide whether the model satisfies the property or – depending on the type of property – to which numerical value (probability or reward) the property evaluates. For example, given a network protocol, one may want to verify that "the probability that a message is eventually delivered is 1", compute "the probability that 95 percent of all messages are delivered within ten milliseconds", or compute "the average length of the sending queue".

The following subsections introduce basic probabilistic models, a probabilistic temporal logic, and the probabilistic model checker PRISM. To illustrate the following definitions, examples in the PRISM modelling language are given before in Section 2.1.9 a summary about the tool concludes this section.

2.1.1 Basic definitions

Let $\mathbb{R}$ denote the set of real numbers and let $\mathbb{N}$ denote the set of non-negative integers.

Definition 2.1. A *discrete probability distribution* over a countable set Q is a function $\mu : Q \to [0,1]$ such that $\sum_{q \in Q} \mu(q) = 1$. Let $\mathsf{Dist}(Q)$ be the set of discrete probability distributions over Q. For a possibly uncountable set Q', let $\mathsf{Dist}(Q')$ be the set of discrete probability distributions over countable subsets of Q'.

Definition 2.2. A *random variable* over a countable set Q is a function $X : Q \to \mathbb{R}$.

In order to define probability measures, the concept of a smallest σ-algebra is used. More details can be found in [84].

Definition 2.3. A *σ-algebra* is a family $\mathcal{F}$ of subsets of a non-empty set S that is closed with respect to complement and countable union.

Proposition 2.4. For any family $\mathcal{F}$ of subsets of a non-empty set, there is a unique *smallest σ-algebra* that contains $\mathcal{F}$.

In order to describe the repeated execution of an experiment with two possible outcomes, the only memoryless discrete probability distribution is the geometric probability distribution.

Proposition 2.5. Let p be the probability for "success" in a single experiment and X be a random variable describing the number of trials needed to reach the first "success". Then the *geometric probability distribution* P for a sequence of trials is given by $P(X = k) = (1-p)^{k-1} \cdot p$.

Note that the geometric probability distribution is memoryless, that is, the probability for "success" in a single experiment is independent from the number of failures observed previously.

2.1.2 Discrete-time Markov chains

The first basic model used in this work is the *discrete-time Markov chain (DTMC)* [50], a discrete-time formalism that features probability but no nondeterminism. My presentation follows [84].

Definition 2.6. Let AP be a finite set of atomic propositions. A *labelled discrete-time Markov chain* is a tuple $M = (S, \bar{s}, p, l, r)$ where:

- S is the finite set of *states*;
- $\bar{s} \in S$ is the *initial state*;
- $p : S \times S \to [0, 1]$ is the *transition probability function*;
- $l : S \to 2^{AP}$ is the *labelling function*;
- $r : S \times S \to \mathbb{R}$ is the *reward function*.

For the sake of simplicity, this definition implicitly assumes that M is *time-homogeneous* (that is, p, l and r do not vary over time), and that S is discrete. It should also be noted that the transition probability and reward functions of M depend on the past only through the current state.

At the outset, the DTMC is in the initial state $\bar{s}$. At each point of time, the successor state s' of the current state $s \in S$ is chosen with probability $p(s, s')$. At each point in time, a *reward* $r(s, s')$[1] is given. If a reward is positive, it is usually interpreted as *income*, otherwise as *cost*.

The syntax and semantics of DTMCs are illustrated in Example 2.7.

Example 2.7. Consider the PRISM code of a DTMC model representing a sender and a receiver shown in Figure 2.1. After declarations of some constants and a reward structure, the model lists two modules: `sender` and `receiver`. These can be interpreted as a fixed sending device and a mobile receiving device, where the latter has limited battery power and switches off its power once it has received a message from the first.

Each module can contain variable declarations (such as for `sending` and `listening`), with optional specification of initial values (via the keyword `init`). It can also contain commands which describe possible transitions and have the form `[`*action*`]` *guard* $\to$ `(`p_1`'=`u_1`) + ...+ (`p_n`'=`u_n`);`, where *action* is an optional action label, *guard* is a propositional expression over the variables of the model, p_i and u_i, $i = 1, \ldots, n$ are non-negative real-valued expressions; p_i describe probabilities and u_i describe updates to variables.

[1] For most optimality criteria, the reward does not depend on the next state, in which case we use the notation $r(s)$ instead.

```
dtmc

const double P_SEND = 0.6;
const double P_RECV = 0.1;

const double E_SLEEP = 0.1;
const double E_LISTENING = 1.0;
const double E_RECEIVING = 2.0;

rewards "energy"
  (listening = 0): E_SLEEP;
  (listening = 1) & (sending = 0): E_LISTENING;
  (listening = 1) & (sending = 1): E_RECEIVING;
endrewards

rewards "transmissions"
  [send] sending=1: 1;
endrewards

module sender
  sending: [0..1];

  [] sending=0 -> P_SEND: (sending'=1) + (1-P_SEND): : (sending'=0);
  [send] sending=1 -> true;
endmodule

module receiver
  listening: [0..1] init 1; // 0...off, 1...listening

  [] listening=0 -> true;
  [send] listening=1 -> P_RECV: true + (1-P_RECV): (listening'=0);
endmodule
```

Figure 2.1: PRISM representation of a discrete-time Markov chain

At each point in time, the models proceed in discrete steps of uniform but not further specified length. For example, consider the first command of module `sender`. Note that, as there is no action label given, this command can be performed without synchronisation with other modules. When the local variable `sending` of module `sender` equals 0, the system can, with a probability of `P_SEND`, update the value of that variable to 1, or, with a probability of $1 -$ `P_SEND`, keep the value of that variable as it is. In contrast, the second command of this module is labelled with an action, `send`, thus it can only be executed synchronously with one command from each module that shares the action `send`. Note that, for each state, the probability for transitions to each other state is uniquely defined; in particular, the model is indeed a DTMC.

Rewards can be defined using *reward structures* for states and transitions. For example, the reward `energy`, describing energy consumption, is defined with respect to transitions,

whereas the reward `transmissions`, describing the number of attempted transmissions, is defined with respect to states. Note that these labelled reward structures extend the PRISM property specification language beyond the definition of PCTL.

If not specified otherwise, all PRISM models are implicitly interpreted as the parallel composition of their modules. Figure 2.2 shows the state-transition diagram of the DTMC model. There are three states, s_0, s_1, and s_2, which are drawn as circles, where the initial

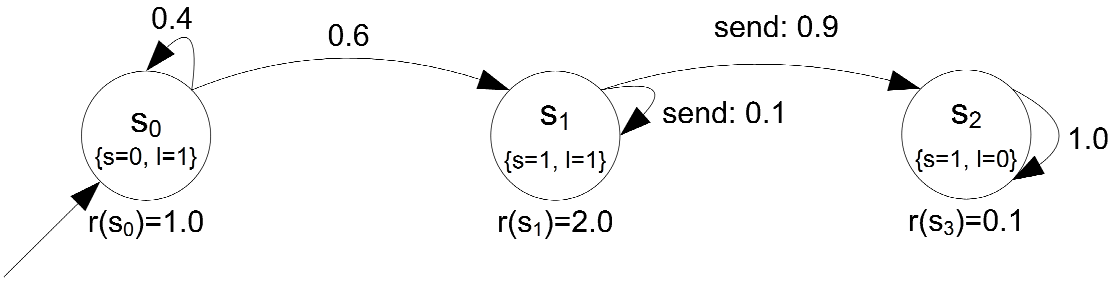


Figure 2.2: State-transition diagram for a discrete-time Markov chain

state s_0 is further emphasised by an incoming arrow without source. The states are labelled with atomic propositions $s = 0$, $s = 1$, $l = 0$, and $l = 1$ (representing the possible values of the variables `sending` and `listening`) in curly brackets. Transitions are indicated by arrows, for example, 0.4 describes that the probability for a transition from s_0 to itself is 0.4. Note that, for each state pair, the probabilities of all outgoing transitions sum up to 1. Each state has been annotated with the respective value of the reward function r, for example, $r(s_0) = 1.0$ describes that the reward for state s_0 is 1.0.

Paths and probability measures

An *infinite path* is an infinite sequence of states $s_0, s_1, \ldots$ where $s_i \in S$ and $p(s_i, s_{i+1}) > 0$ for all $i \geq 0$. A *finite path* is a nonempty sequence of states $s_0, \ldots, s_{n-1}, s_n$ where $s_i \in S$ and $p(s_i, s_{i+1}) > 0$ for all $0 \leq i < n$. Let $\omega(i)$ denote the ith state and $|\omega|$ denote the length of the path ω. The set of all finite and infinite paths starting in state $s \in S$ is denoted Path_s and the set of all paths is denoted Path.

In order to describe the quantitative behaviour of a DTMC with respect to a given property, we define a probability measure over sets of paths. First, we define the probability

that, for a given state $s \in S$, the finite path $\omega = s_0, \ldots, s_n \in \mathrm{Path}_s$ is taken as

$$p_s(\omega) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } n = 0 \\ \prod_{i=0}^{n-1} p(s_i, s_{i+1}) & \text{otherwise} \end{cases}.$$

Then, we define the *cylinder set* of the path ω, that is, the set of all infinite paths with prefix ω, by

$$C_s(\omega) \stackrel{\text{def}}{=} \{\omega' \in \mathrm{Path}_s \,|\, \omega \text{ is a prefix of } \omega'\}.$$

Finally, let Σ_s be the smallest σ-algebra on Path_s that contains all the sets $C_s(\omega)$ where ω ranges over finite paths in Path_s. For a given state $s \in S$, we define the probability measure Prob_s on Σ_s as the unique measure such that:

$$\mathrm{Prob}_s(C_s(\omega)) \stackrel{\text{def}}{=} p_s(\omega) \text{ for all finite } \omega \in \mathrm{Path}_s.$$

In order to reason about reward properties, we consider random variables over paths. We use $\mathrm{E}_s(X)$ to denote the usual expectation of the random variable $X : \mathrm{Path} \to \mathbb{R}$ with respect to the probability measure Prob_s.

2.1.3 Markov decision processes

The second basic model used in this thesis is the *Markov decision process (MDP)* [27] model, a discrete-time formalism that features both nondeterminism and probability. My presentation follows [81] and [84].

Definition 2.8. Let AP be a finite set of atomic propositions. A *labelled Markov decision process* is a tuple $M = (S, \bar{s}, A, p, l, r)$ where:

- S is the finite set of *states*;
- $\bar{S} \subseteq S$ is the set of *initial states*;
- A is the finite set of *actions*;
- $p : S \times A \to \mathsf{Dist}(S)$ is the *transition probability function*;
- $l : S \to 2^{AP}$ is the *labelling function*;

- $r : S \times A \times S \to \mathbb{R}$ is the *reward function*.

Similarly to the definition of DTMCs, this definition implicitly assumes that M is *time-homogeneous* (that is, S, A, p, l and r do not vary over time), and that S and A are discrete. Also, the transition probability and reward functions of M depend on the past only through the current state. While MDPs can generally have either discrete or continuous sets of decision periods, we also assume that all MDP models in this thesis use discrete time.

At the outset, the MDP is in some initial state $\bar{s} \in \bar{S}$. At each point of time, called *decision period*, an action $a \in A$ is chosen nondeterministically, and consequently the successor state s' of the current state $s \in S$ is determined by the probability distribution $p(s, a)$, that is, it is chosen with probability $p(s, a)(s')$. It is usual to assume the presence of a so-called *decision maker*, an entity that resolves this choice of actions. In each decision period, the decision maker receives the *reward* $r(s, a, s')$[2]. As for DTMCs, if a reward is positive, it is usually interpreted as *income*, otherwise as *cost*.

The syntax and semantics of MDPs are illustrated in Example 2.9.

Example 2.9. Consider the PRISM representation of an MDP model shown in Figure 2.3. This model is based on the DTMC model shown in Figure 2.1 and only differs in the definition of the `receiver` module. In comparison with the DTMC model, the guard of the first

```
mdp

const double P_SEND = 0.6;
const double P_RECV = 0.1;

...

module sender
  sending: [0..1];

  [] true -> P_SEND: (sending'=1) + (1-P_SEND): : (sending'=0);
  [send] sending=1 -> true;
endmodule

...
```

Figure 2.3: PRISM representation of a Markov decision process

transition of module `sender` has been weakened, allowing a nondeterministic choice between

[2]For most optimality criteria, the reward does not depend on the next state, in which case we use the notation $r(s, a)$ instead.

both of its commands whenever the guard conditions `sending=1` and `listening=1` of the synchronised commands hold.

Figure 2.4 shows the state-transition diagram of the MDP model. Compared to the

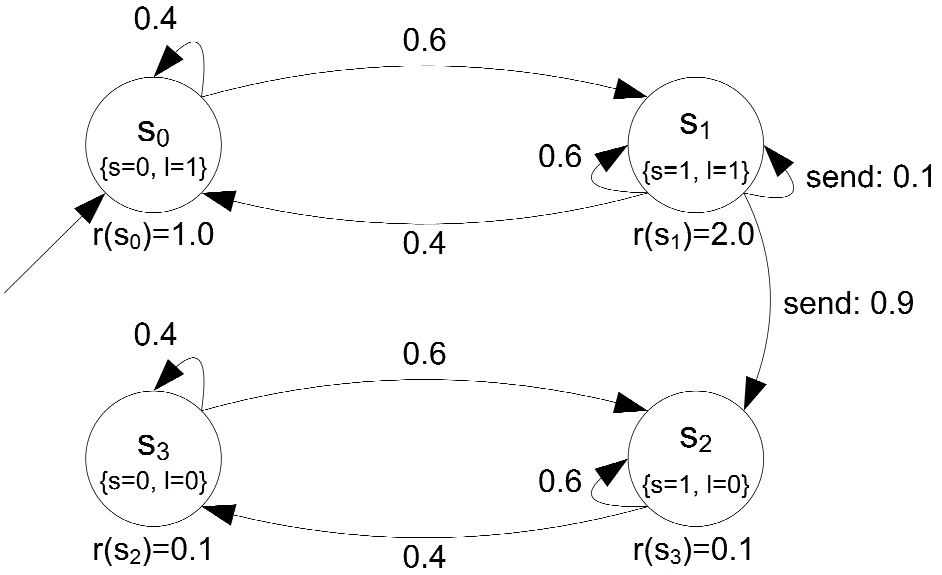


Figure 2.4: State-transition diagram for a Markov decision process

DTMC model, there are several more transitions, and a fourth state s_3 is reachable. Note that for some state pairs there is more than one corresponding transition, which is a characteristic feature for nondeterminism. Note that, for each state pair, the probabilities of all outgoing transitions under a fixed action sum up to 1.

Adversaries

The behaviour of the decision maker is controlled by a procedure called *policy*[3], which, for each state, determines the choice of action.

We distinguish policies based on two criteria: first, the way in which actions are chosen, and second, the extent to which they incorporate information about the past. A policy is *deterministic* if it selects actions with certainty and it is *randomised* if it specifies a probability distribution on the set of actions. A policy is *Markovian* if it depends – that is, its choices depend – on past states and actions only through the current state, and it is *history-dependent* if it depends on the whole *history* $h_t = (s_0, a_0, \ldots s_{t-1}, a_{t-1}, s_t)$ of past states s_i, $0 \leq i \leq t$, and actions a_j, $0 \leq j < t$, where t is the current decision period; we use H_t to denote the set

[3]In the literature, policies are also called "adversary", "plan", or "strategy".

of all histories h_t. A policy is *stationary* if it does not depend on the decision period t.

Paths and probability measures

An *infinite path* is an infinite sequence of states and actions $s_0, a_0, s_1, a_1, \ldots$ where $s_i \in S$, $a_i \in A$, and $p(s_i, a_i)(s_{i+1}) > 0$ for all $i \geq 0$. A *finite path* is a nonempty sequence of states $s_0, a_0, \ldots, s_{n-1}, a_{n-1}, s_n$ where $s_i \in S$, $a_j \in A$, and $p(s_j, a_j)(s_{j+1}) > 0$ for all $0 \leq i \leq n$ and $0 \leq j < n$. Let $\omega(i)$ denote the ith state, $\mathrm{step}(\omega, i)$ denote the ith action, and $|\omega|$ denote the length of the path ω. The set of all finite and infinite paths starting in state $s \in S$ is denoted Path_s and the set of all such paths for a given policy π is denoted Path_s^π. The set of all paths under a given policy π is denoted Path^π, and the set of all paths under any policy is denoted Path.

In order to describe the quantitative behaviour of an MDP with respect to a given property, we define a probability measure over sets of paths. First, we define the probability that, for a given policy π and state $s \in S$, the finite path $\omega = s_0, a_0, \ldots, s_{n-1}, a_{n-1}, s_n \in \mathrm{Path}_s^\pi$ is taken as

$$p_s^\pi(\omega) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } n = 0 \\ \prod_{i=0}^{n-1} p^\pi(s_i, a_i)(s_{i+1}) & \text{otherwise} \end{cases}.$$

Then, we define the *cylinder set* of the path ω, that is, the set of all infinite paths with prefix ω, as

$$C_s^\pi(\omega) \stackrel{\text{def}}{=} \{\omega' \in \mathrm{Path}^\pi \mid \omega \text{ is a prefix of } \omega'\}.$$

Finally, let Σ_s be the smallest σ-algebra on Path_s that contains all the sets $C_s(\omega)$ where ω ranges over finite paths in Path_s. For a given state $s \in S$ and policy π, we define the probability measure Prob_s^π on Σ_s as the unique measure such that:

$$\mathrm{Prob}_s^\pi(C_s^\pi(\omega)) \stackrel{\text{def}}{=} p_s^\pi(\omega) \text{ for all finite } \omega \in \mathrm{Path}_s.$$

In order to reason about reward properties, we consider random variables over paths. For a given policy π, we use $\mathrm{E}_s^\pi(X)$ to denote the usual expectation of the random variable $X : \mathrm{Path} \to \mathbb{R}$ with respect to the probability measure Prob_s^π.

2.1.4 Probabilistic Timed Automata

Probabilistic timed automata (PTA) [63] are a powerful modelling formalism for distributed systems that supports dense time, nondeterminism, and probabilistic choice. They are a generalisation of *timed automata* [6, 7, 28] that is obtained by adding a probabilistic transition relation. We include the notion of *urgent events* [64], a common feature of classical timed automata [24, 47]. Our presentation follows [61].

Syntax

Let $\mathcal{X}$ be a finite set of variables over $\mathbb{R}$, called *clocks*. A function $v : \mathcal{X} \to \mathbb{R}$ is referred to as a *clock valuation*. Let $\mathbb{R}^{\mathcal{X}}$ be the set of all clock valuations.

Let $Zones(\mathcal{X})$ denote the set of all *zones* over $\mathcal{X}$, which are conjunctions of atomic constraints of the form $x \lhd c$ for $x \in \mathcal{X}$, $\lhd \in \{\leq, =, \geq\}$, and $c \in \mathbb{N}$. A clock valuation $v \in \mathbb{R}^{\mathcal{X}}$ satisfies a clock constraint ζ, denoted $v \lhd \zeta$, if and only if ζ resolves to true after substituting each clock $x \in \mathcal{X}$ with the corresponding value $v(x)$.

Definition 2.10. A *probabilistic timed automaton* is a tuple $\mathcal{P} = (L, \bar{l}, \mathcal{X}, A, inv, enab, prob, lab, rew)$ where:

- L is a finite set of *locations*;
- $\bar{l} \in L$ is the *initial location*;
- X is a finite set of *clocks*;
- A is a finite set of *actions*, of which $A_u \subseteq A$ are *urgent*;
- $inv : L \to Zones(\mathcal{X})$ is the *invariant condition* function;
- $enab : L \times A \to Zones(\mathcal{X})$ is the *enabling condition* function;
- $prob \subseteq L \times A \to \mathsf{Dist}(2^{\mathcal{X}} \times L)$ is the *probabilistic transition function*;
- $lab : L \to 2^{AP}$ is the *labelling function*;
- $rew : L \times A \times L \to \mathbb{R}$ is the *reward function*.

To ensure that a finite semantics can be obtained later, the syntax implicitly restricts the PTA considered to be *diagonal-free* [62], that is, atomic constraints of the form $x - y \lhd c$ are not allowed. Additionally, in order to ensure the existence of a well-defined semantics, it is explicitly required that the PTAs considered are *structurally non-Zeno* [95], which can be checked syntactically [96].

Semantics

In each location of a probabilistic timed automaton, there is a nondeterministic choice between two types of transitions. *Delay transitions* correspond to the elapsing of time in a location. They are permitted as long as the invariant condition is satisfied and no urgent transitions (transitions under urgent actions) are enabled. *Action transitions* correspond to the execution of probabilistic transitions $(l, g, a, p) \in \mathit{prob}$. If the current location l satisfies the clock constraint g and the current action is a, then $p((X', l'))$ is the probability of resetting all clocks in X' to 0 and moving to the location l'.

This notion of a probabilistic timed automaton is strong enough to represent several higher-level features such as *urgent locations* and *integer variables*. In urgent locations, only action transitions are allowed, that is, such locations have to be left immediately without time passing. They can be modelled using an additional clock [24, 94]. Integer variables with bounded ranges, which can be tested within enabling conditions and reset by action transitions, can be represented by encoding their values within locations [93].

A state of a PTA is a pair of a location and a valuation for all clocks $(l, v) \in L \times \mathbb{R}^{\mathcal{X}}$ such that all invariant conditions of the location hold ($v \lhd \mathit{inv}(l)$). In any state (l, v), a certain amount of time $t \in \mathbb{R}$ can elapse, after which an action $a \in A$ is performed. The choice of t requires that, while time passes, the invariant $\mathit{inv}(l)$ remains continuously satisfied. Each action a can only be chosen if it is *enabled*, that is, the zone $\mathit{enab}(l, a)$ is satisfied by $v + t$. Once action a is chosen, a set of clocks to reset and a successor location are selected at random, according to the distribution $\mathit{prob}(l, a)$. We call each element $(X, l') \in 2^{\mathcal{X}} \times L$ in the support of $\mathit{prob}(l, a)$ an *edge* and, for convenience, assume that the set of such edges, denoted $\mathit{edges}(l, a)$, is an ordered list $\langle e_1, \ldots, e_n \rangle$.

The syntax and semantics of PTAs are illustrated in Example 2.11. A finite semantics is formally given in Section formally 2.1.5.

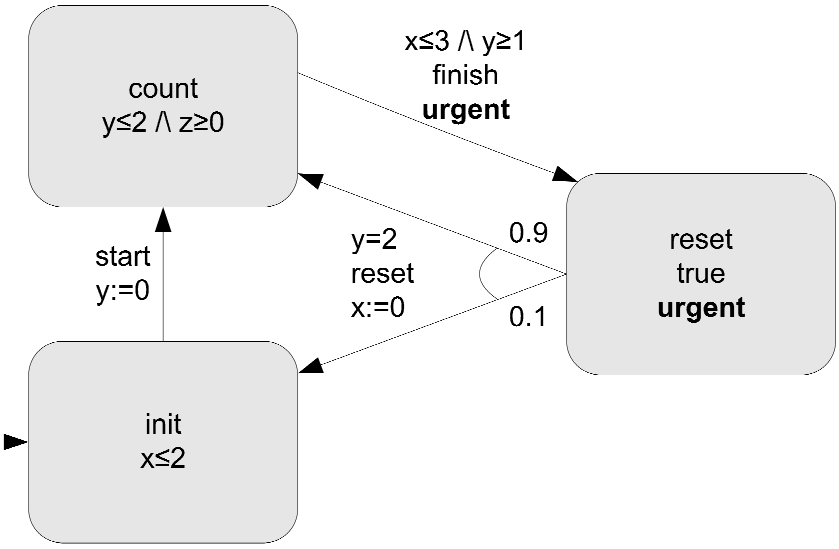


Figure 2.5: A probabilistic timed automaton

Example 2.11. Consider the PTA shown in Figure 2.5. There are three locations: *init*, *count*, and *reset*, which are drawn as boxes, where the initial location INIT is further emphasised by an incoming arrow without source. The locations are labelled with the invariant conditions $x \leq 2$, $y \leq 2 \wedge z \geq 0$, and true, respectively.

Transitions are indicated by one or more arrows, which are annotated with the transition's enabling condition, action, and set of clock resets. A probabilistic choice between different transitions is represented by a curve that links both transitions' arrows near their common source location. For example, from location *reset*, when the clock constraint $y = 2$ is met and the action *reset* occurs, the location is changed to *count* (with a probability of 0.9) or to *init* (with a probability of 0.1); in both cases, the clock x is reset to 0. Note that, for each location-action pair, the probabilities of outgoing transitions sum up to 1.

Each execution of the PTA starts in its initial location (here *init*) with all its clocks (here at least x, y, and z) set to 0. In the state *init*, initially both delay and event transitions are possible. For a delay transition, time can progress for up to two time units, until clock x has reached 2. For an event transition, whenever the event *start* occurs, a transition to location *count* can take place, thereby resetting clock y to 0.

For location *count*, an urgent event is defined, that is, this event has to be taken whenever both its invariant condition (here $x \leq 3 \wedge y \geq 1$) holds and its event (here $finish$) occurs. In this case, no delay transition and no other event transition for a non-urgent event may take

place. The location *reset* is defined to be an urgent location. That means that whenever an event transition is possible, no delay transition may occur.

2.1.5 Probabilistic Computation Tree Logic

The usual specification language for probabilistic model checking of DTMCs and MDPs is *Probabilistic Computation Tree Logic (PCTL)* [42]. PCTL is an extension of the non-probabilistic Computation Tree Logic (CTL) [23]. My presentation follows [9, 59, 84].

Syntax

Definition 2.12. The syntax of PCTL is defined as follows: let a be an atomic proposition, $\bowtie \in \{<, \leq, \geq, >\}$, $p \in [0,1]$, $r \in \mathbb{R}$, and $k \in \mathbb{N}$. Then, the set of well-formed PCTL formulae ϕ (where ϕ are state formulae and ψ are path formulae) is inductively defined as follows:

1. $\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{P}_{\bowtie p}\,[\psi] \mid \mathcal{R}_{\bowtie r}\,[\mathrm{C}^{\leq k}] \mid \mathcal{R}_{\bowtie r}\,[\mathrm{F}\,\phi]$ are state formulae;

2. $\psi ::= \mathcal{X}\,\phi \mid \phi\,\mathcal{U}\,\phi \mid \phi\,\mathcal{U}^{\leq k}\,\phi$ are path formulae.

The symbols $\mathcal{P}$ and $\mathcal{R}$ represent special operators for *probability* and *expected reward*, respectively. The symbols $\mathcal{X}$ and $\mathcal{U}$ represent the usual operators for *next* and *until*. For convenience, further operators are given by the following standard abstractions:

$$\bot \stackrel{\text{def}}{=} \neg\top$$
$$\varphi_1 \vee \varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2)$$
$$\varphi_1 \rightarrow \varphi_2 \stackrel{\text{def}}{=} \neg\varphi_1 \vee \varphi_2$$
$$\mathcal{F}\,\varphi \stackrel{\text{def}}{=} \top\,\mathcal{U}\,\varphi$$

For the evaluation of formulae, we require that unary operators have a higher precedence than binary operators, that is, arguments of unary operators match as small as possible subformulae and arguments of binary operators match as large as possible subformulae.

DTMC semantics

For a given path $\omega \in \mathrm{Path}$, we define random variables over paths as follows:

$$X_{\mathrm{C}^{\leq k}}(\omega) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } k = 0 \\ \sum_{i=0}^{k-1} r(\omega(i), \mathrm{step}(\omega, i), \omega(i+1)) & \text{otherwise} \end{cases};$$

$$X_{\mathrm{F}\,\varphi}(\omega) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } s_0 \vDash \varphi \\ \infty & \text{if } s_i \nvDash \varphi \text{ for all } i \in \mathbb{N} \cdot \\ \sum_{i=0}^{\min\{j \,|\, s_j \vDash \varphi\}-1} r(\omega(i), \mathrm{step}(\omega, i), \omega(i+1)) & \text{otherwise} \end{cases}$$

They describe the *cumulative reward* $X_{\mathrm{C}^{\leq k}}$ as the sum of all transition rewards along this path, and they describe the *reachability reward* as the sum of all transition rewards along this path until the first state satisfies the formula φ. Note that we define the reward of a path which does not reach a state satisfying φ to be ∞, even though the total reward of the path may not be infinite.

We define $p_s(\psi) \stackrel{\text{def}}{=} \mathrm{Prob}_s(\{\omega \in \mathrm{Path}_s \,|\, \omega \vDash \psi\})$. Note that this use of the notation p_s (over formulae) is different from that in section 2.1.2 (over paths).

Definition 2.13. The semantics of PCTL formulae for a labelled DTMC $M = (S, \bar{s}, p, l, r)$ is defined as follows:

1. A state $s \in S$ satisfies a state formula φ, denoted $s \vDash \varphi$, if the following holds:

$$\begin{aligned} s \vDash \top &\iff \text{true}; \\ s \vDash a &\iff a \in l(s) \text{ for all } a \in AP; \\ s \vDash \neg\varphi &\iff s \nvDash \varphi; \\ s \vDash \varphi_1 \wedge \varphi_2 &\iff s \vDash \varphi_1 \text{ and } s \vDash \varphi_2; \\ s \vDash \mathcal{P}_{\bowtie p}\,[\psi] &\iff p_s(\psi) \bowtie p; \\ s \vDash \mathcal{R}_{\bowtie r}\,[\mathrm{C}^{\leq k}] &\iff \mathrm{E}_s(X_{\mathrm{C}^{\leq k}}) \bowtie r; \\ s \vDash \mathcal{R}_{\bowtie r}\,[\mathrm{F}\,\varphi] &\iff \mathrm{E}_s(X_{\mathrm{F}\,\varphi}) \bowtie r; \end{aligned}$$

   where $X_{\mathrm{C}^{\leq k}}$ and $X_{\mathrm{F}\,\varphi}$ are the random variables defined above.

2. A path $\omega \in \text{Path}$ satisfies a path formula ψ, denoted $\omega \vDash \psi$, if the following holds[4]:

$$\begin{aligned}
\omega \vDash \mathcal{X}\,\varphi &\overset{\text{def}}{\iff} \omega(1) \vDash \varphi \text{ for all state formulae } \varphi;\\
\omega \vDash \varphi_1 \,\mathcal{U}\, \varphi_2 &\overset{\text{def}}{\iff} \omega^i \vDash \varphi_2 \text{ for some } i \geq 0 \text{ and}\\
&\qquad\quad \omega^j \vDash \varphi_1 \text{ for all } 0 \leq j < i;\\
\omega \vDash \varphi_1 \,\mathcal{U}^{\leq k}\, \varphi_2 &\overset{\text{def}}{\iff} \omega^i \vDash \varphi_2 \text{ for some } 0 \leq i \leq k \text{ and}\\
&\qquad\quad \omega^j \vDash \varphi_1 \text{ for all } 0 \leq j < i.
\end{aligned}$$

3. M satisfies a state formula φ, denoted $M \vDash \varphi$, if and only if $\bar{s} \vDash \varphi$.

The semantics of PCTL formulae with respect to a DTMC is illustrated in Example 2.14.

Example 2.14. Consider the DTMC from Example 2.7, which is shown in Figure 2.2. We would like to validate that the probability that the receiver eventually receives one of the messages transmitted by the sender (and turns off its radio) is at least 99 percent. In PCTL, this property can be expressed by the formula $\mathcal{P}_{\geq 0.99}\,[\mathcal{F}\,(\mathit{listening} = 0)]$. In PRISM, this is written as

```
P>=0.99 [F (listening=0)].
```

Also, we would like to compute the expected energy consumption of the receiver over ten time units. This property can be expressed in the PRISM specification language by the query expression $\mathcal{R}_{\text{energy}=?}\,[\text{C}^{\leq 10}]$, exceeding PCTL, where probability and reward operators are restricted to comparisons with constant values. In PRISM, this is written as

```
R{"energy"}=?  [C<=10].
```

The first formula would be evaluated to true and the second one to 4.61.

MDP semantics

Contrary to DTMCs, MDPs support nondeterminism, which appears in form of actions in the transition probability function and the reward function. For MDPs, the operators $\mathcal{P}$ and $\mathcal{R}$ quantify over all policies, that is, they refer to minimum and maximum probability and expected reward.

[4]Note that, for any path formula φ, state $s \in S$, the set of paths $\{\omega \in \text{Path}_s \mid \omega \vDash \varphi\}$ is measurable.

Similarly to p_s for the DTMC semantics, we define, for a given policy π, $p_s^\pi(\psi) \stackrel{\text{def}}{=} \text{Prob}_s^\pi(\{\omega \in \text{Path}_s^\pi \mid \omega \vDash \psi\})$. Note that this use of the notation p_s^π (over formulae) is different from that in section 2.1.3 (over paths).

Definition 2.15. The semantics of PCTL formulae for a labelled MDP $M = (S, \bar{s}, A, p, l, r)$ is defined as follows[5]: A state $s \in S$ satisfies a state formula φ, denoted $s \vDash \varphi$, if the following holds:

$$
\begin{aligned}
s \vDash \mathcal{P}_{\bowtie p}[\psi] &\iff^{\text{def}} p_s^\pi(\psi) \bowtie p \text{ for all policies } \pi \text{ of } M;\\
s \vDash \mathcal{R}_{\bowtie r}[\mathrm{C}^{\leq k}] &\iff^{\text{def}} \mathrm{E}_s^\pi(X_{\mathrm{C}^{\leq k}}) \bowtie r \text{ for all policies } \pi \text{ of } M;\\
s \vDash \mathcal{R}_{\bowtie r}[\mathrm{F}\,\varphi] &\iff^{\text{def}} \mathrm{E}_s^\pi(X_{\mathrm{F}\,\varphi}) \bowtie r \text{ for all policies } \pi \text{ of } M.
\end{aligned}
$$

Note that we define the reward of a path which does not reach a state satisfying φ to be ∞, even though the total reward of the path may not be infinite.

The semantics of PCTL formulae with respect to a MDP is illustrated in Example 2.16.

Example 2.16. Consider the MDP from Example 2.9, which is shown in Figure 2.4. We would like to evaluate the minimum probability that the receiver eventually receives one of the messages transmitted by the sender (and turns off its radio). In PCTL, this property can be expressed by the formula $\mathcal{P}_{\min=?}[\mathcal{F}(\mathit{listening} = 0)]$. In PRISM, this is written as

```
Pmin=?  [F (listening=0)].
```

Also, we would like to compute the maximum expected number of transmissions attempts of the sender until the receiver receives a message. This property can be expressed by the formula $\mathcal{R}_{\max(\text{transmissions})=?}[\mathrm{F}(\mathit{listening} = 0)]$. In PRISM, this is written as

```
R{"transmissions"}max=?  [F (listening=0)].
```

The first formula would be evaluated to 0.0 (for example, consider the policy that never choses the `send` action) and the second one to ∞.

PTA semantics

Probabilistic model checking requires a finite system model. Contrary to discrete-time Markov chains and Markov decision processes, probabilistic timed automata are not gen-

[5] All clauses that are identical for DTMCs and MDPs have been omitted.

erally finite, owing to their dense-time semantics (clocks are real-valued). In order to make automatic verification theoretically and practically feasible, several abstraction techniques are used.

A finite-state *integral semantics* for a subclass of probabilistic timed automata has been introduced in the *digital clocks* approach of Kwiatkowska et al. [56, 62]. In this approach, probabilistic timed automata are represented as Markov decision processes, thereby admitting probabilistic model checking techniques against PCTL properties that have been developed for MDPs. The authors have shown that this semantics preserves probabilistic reachability and expected reachability properties of non-Zeno, diagonal-free probabilistic timed automata.

The newer, game-based approach for obtaining an integral semantics [61] achieves smaller MDP models, but it can only be used to compute probabilistic reachability measures. The computation of expected reachability measures, needed for the evaluation of reward properties, is not possible.

For the class of non-Zeno, diagonal-free probabilistic timed automata, the digital clocks approach can be used to derive PRISM models directly. Hartmanns et al. [43, 44] discuss this topic from the objective of generating PRISM models for PTAs specified in the language *Modest*.

Following the presentation of [61], the finite semantics of a PTA is defined as follows.

Definition 2.17. Let $\mathcal{P} = (L, \bar{l}, \mathcal{X}, A, inv, enab, prob, lab, rew)$ be a PTA. The semantics of P is defined as the infinite-state MDP $[\![P]\!] = (S, \bar{s}, \mathbb{R} \times A, prob', lab', rew')$ where:

- $S = \{(l, v) \in L \times \mathbb{R} \mid v \lhd inv(l)\}$ and $\bar{s} = \{(\bar{l}, 0)\}$;
- $prob'((l, v), (t, a)) = \lambda$ if and only if $v + t' \lhd inv(l)$ for all $0 \leq t' \leq t$, $v + t \lhd enab(l, a)$ and, for any $(l', v') \in S$:

$$\lambda(l', v') = \sum \{prob(l, a)(X, l') \mid X \in 2^{\mathcal{X}} \wedge v' = (v + t)[X := 0]\};$$

- $lab'((l, v)) = lab(l)$;
- $rew'((l, v), a, (l', v')) = rew(l, a, l')$.

Each transition of the semantics of the PTA is a time-action pair (t, a), representing time passing for t time units, followed by a discrete a-labelled transition. If $p((l, v), (t, a))$

is defined and $edges(l,a) = \langle (l_1, X_1), \ldots, (l_n, X_n) \rangle$, we write $(l,v) \xrightarrow{t,a} \langle s_1, \ldots, s_n \rangle$ where $s_i = (l_i, (v+t)[X_i := 0])$ for all $1 \leq i \leq n$.

In order to represent sets of PTA states, we use the concept of a *symbolic state*: a pair $z = (l, \zeta)$, comprising a location l and a zone ζ over $\mathcal{X}$, represents the set of PTA states $\{(l,v) \mid v \lhd \zeta\}$. We use the notation $(l,v) \in (l,\zeta)$ to denote inclusion of a PTA state in a symbolic state.

For a symbolic state (l,ζ), an action a, and an edge $e = (X, l') \in edges(l,a)$, we define:

- $\mathrm{tsuc}(l,\zeta) \stackrel{\mathrm{def}}{=} (l, inv(l) \wedge \nwarrow \zeta)$ is the *time successor* of (l,ζ);
- $\mathrm{dsuc}[a,e](l,\zeta) \stackrel{\mathrm{def}}{=} (l', (\zeta \wedge enab(l,a))[X := 0] \wedge inv(l'))$ is the discrete successor of (l,ζ) with respect to e;
- $\mathrm{post}[a,e](l,\zeta) \stackrel{\mathrm{def}}{=} \mathrm{tsuc}(\mathrm{dsuc}[a,e](l,\zeta))$ is the post of (l,ζ) with respect to e

The *c-closure* of a zone ζ is obtained by removing any constraint that refers to integers greater than c. For a given c, there are only a finite number of c-closed zones. In this book, we assume that all zones are c-closed where c is the largest constant appearing in the PTA under study.

Representation in PRISM For the PTA from Example 2.11 (which is non-Zeno and diagonal-free), the PRISM model in Figure 2.6 for an MDP has been derived. The PRISM model consists of a single module. For each location, a constant of type integer is defined. For each clock, a variable is defined, with a range from 0 to the maximum value the clock is compared to plus one, as prescribed by [62]. For each transition, a command `[`*action*`]` *guard* $\rightarrow$ *assignments*`;` is defined as follows: For delay transitions, the action is empty, the guard comprises the source location, negations of all its urgent transitions' invariant conditions, and its enabling condition for the current time plus one; the assignment comprises an increment by one for each clock. For action transitions, the action label is as in the PTA, the guard comprises the source location, negations of all its other transitions' invariant conditions, and the transition's invariant condition; the assignment comprises the transition's clock resets. For example, consider location COUNT of the PTA. A delay transition from this location is only allowed when the invariant condition $x \leq 3 \wedge y \geq 1$ is not enabled. As the PTA contains three clocks, the transition has to increment all of them by one, up to the maximum value the respective clock is compared to within the PTA, which in this example is represented by

```
mdp

// states
const int INIT = 0;
const int COUNT = 1;
const int RESET = 2;

// maximum values clocks are compared plus one
const int MAX_x = 4;
const int MAX_y = 3;
const int MAX_z = 1;

module example
  location: [0..2] init INIT;

  // clocks
  x: [0..MAX_x];
  y: [0..MAX_y];
  z: [0..MAX_z];

  // delay transitions
  [] location=INIT & x+1<=2
      -> (location'=INIT) & (x'=min(x+1,MAX_x)) & (y'=min(y+1,MAX_y)) & (z'=min(z+1,MAX_z));
  [] location=COUNT & !(x<=3 & y>=1) & y+1<=2 & z+1>=0
      -> (location'=COUNT) & (x'=min(x+1,MAX_x)) & (y'=min(y+1,MAX_y)) & (z'=min(z+1,MAX_z));
  [] location=RESET
      -> (location'=RESET) & (x'=min(x+1,MAX_x)) & (y'=min(y+1,MAX_y)) & (z'=min(z+1,MAX_z));

  // action transitions
  [start] location=INIT -> (location'=COUNT) & (y'=0);
  [finish] location=COUNT & x<=3 & y>=1
      -> (location'=RESET);
  [reset] location=RESET & y=2
      -> 0.1: (location'=INIT) & (x'=0) + 0.9: (location'=COUNT) & (x'=0);
endmodule
```

Figure 2.6: PRISM representation of a Markov decision process obtained from the probabilistic timed automaton in Example 2.11

MAX_x for each clock x. This abstraction can be justified as follows: First, all atomic clock constraints are non-strict comparisons of a single clock and a non-negative integer; therefore incrementing clocks in steps of one suffices to cover all states of the PTA. Second, clock values in PTAs are only used for checking clock constraints, which only include values up to MAX_x; thus these comparisons always yield the same result for all values equal or greater than MAX_x; therefore it is sound to limit the maximum value of each clock to the maximum value it is compared to. Thus the PRISM command representing this transition is the following one:

```
  [] location=COUNT & !(x<=3 & y>=1) & y+1<=2 & z+1>=0
      -> (location'=COUNT) & (x'=min(x+1,MAX_x)) & (y'=min(y+1,MAX_y)) & (z'=min(z+1,MAX_z));
```

The reward functions of a PTA, which are defined with respect to a location and an action of the PTA, are translated in a similar way, based on corresponding state and action label in the PRISM model, respectively.

Timescale abstraction Unfortunately, the integral semantics contributes to the state explosion problem, as it leads to models of a size exponential in the number of clocks and the largest constant that the clocks are compared to. In order to cope with that, *timescale abstraction* can be used to reduce the size of a model. Alur et al. [5] have shown that the original model is a refinement of the reduced model as follows:

Proposition 2.18. Let $c \stackrel{\text{def}}{=} a \leq x \leq b$ be an atomic constraint in a timed automaton, with x a clock and $a, b \in \mathbb{N}$. Let $\mathcal{L}(c)$ denote the set of all executions of the automaton containing c.

1. Let k be the greatest common divisor of a and b. Define $c \stackrel{\text{def}}{=} a' \leq x \leq b'$ with $a' \stackrel{\text{def}}{=} a/k$ and $b' \stackrel{\text{def}}{=} b/k$. Let $\mathcal{L}(c)$ denote the set of all executions of the automaton containing c'. Then $\mathcal{L}(c) \subseteq \mathcal{L}(c')$.

2. Let $a'' \leq a$ and $b'' \leq b$. Define $c'' \stackrel{\text{def}}{=} a'' \leq x \leq b''$. Let $\mathcal{L}(c)$ denote the set of all executions of the automaton containing c''. Then $\mathcal{L}(c) \subseteq \mathcal{L}(c'')$.

They suggest to perform a refinement of the original model in two steps: First, lower bounds are rounded down and upper bounds are rounded up such that afterwards they have a large common factor. Following this, all bounds are divided by their greatest common divisor. This is equivalent to dividing all constants clocks are compared to by the value of a new time unit and then rounding lower bounds down and upper bounds up. For each of these operations, the set of all executions of the original model is a subset of all operations of the modified model. Therefore, the maximum and minimum probabilistic and expected reachability measures of the reduced model are upper and lower bounds of those for the original model.

2.1.6 Model-checking algorithms

To evaluate whether a DTMC or an MDP conforms to a PCTL specification, we use a set of algorithmic techniques known as *model checking*. Our presentation of the model-checking algorithms follows [9], [84], and [59].

As the model-checking algorithm for DTMCs is analogous to the one for MDPs, we only present the latter. DTMCs are deterministic and do not have a concept of policies. Thus, the computations in the DTMC algorithm can usually be obtained from the ones in the MDP algorithm by removing quantification over policies.

The algorithm takes an MDP $M = (S, \bar{s}, A, p, l, r)$ and a PCTL formula φ as input and terminates with an output of *true* if $M \vDash \varphi$ and *false* otherwise. Starting from atomic propositions, the algorithm recursively computes the sets of states satisfying the subformulae of φ, denoted $\mathrm{Sat}(\varphi)$.

Preliminaries

As MDPs are nondeterministic, determining probabilities and expected rewards requires computing maxima or minima over all policies – depending on whether the relational operator $\bowtie$ relates to a lower or upper bound. Let

$$
\begin{aligned}
p_s^{\max}(\psi) &\stackrel{\text{def}}{=} \sup_{\pi} \left\{ p_s^{\pi}(\psi) \right\}, \\
p_s^{\min}(\psi) &\stackrel{\text{def}}{=} \inf_{\pi} \left\{ p_s^{\pi}(\psi) \right\}, \\
e_s^{\max}(X) &\stackrel{\text{def}}{=} \sup_{\pi} \left\{ E_s^{\pi}(X) \right\}, \\
e_s^{\min}(X) &\stackrel{\text{def}}{=} \inf_{\pi} \left\{ E_s^{\pi}(X) \right\}.
\end{aligned}
$$

If $\bowtie \in \{<, \leq\}$, then:

$$
\begin{aligned}
\mathrm{Sat}(\mathcal{P}_{\bowtie p}\,[\psi]) &:= \{ s \in S \mid p_s^{\max}(\psi) \bowtie p \}, \\
\mathrm{Sat}(\mathcal{R}_{\bowtie r}\,[\varphi]) &:= \{ s \in S \mid e_s^{\max}(X_\varphi) \bowtie p \}.
\end{aligned}
$$

If $\bowtie \in \{\geq, >\}$, then:

$$\mathrm{Sat}(\mathcal{P}_{\bowtie p}\,[\psi]) := \{s \in S \,|\, p_s^{\min}(\psi) \bowtie p\},$$
$$\mathrm{Sat}(\mathcal{R}_{\bowtie r}\,[\varphi]) := \{s \in S \,|\, e_s^{\min}(X_\varphi) \bowtie p\},$$

For the sake of brevity, we write $p_s^{\max}$, $p_s^{\min}$, $e_s^{\max}$, and $e_s^{\min}$ to abbreviate $p_s^{\max}(\psi)$, $p_s^{\min}(\psi)$, $e_s^{\max}(X)$, and $e_s^{\min}(X)$, respectively.

The remainder of this subsection comprises the computation of $\mathrm{Sat}(\varphi)$ for the different cases of PCTL formulae φ. We show the computation of maximum probabilities and maximum expected rewards, but omit the minimum cases, which are completely analogous.

Propositional formulae

For propositional formulae, the computation of the set of satisfying states is straightforward:

$$\mathrm{Sat}(\top) := S;$$
$$\mathrm{Sat}(a) := \{s \,|\, a \in l(s)\} \text{ for all } a \in AP;$$
$$\mathrm{Sat}(\neg\varphi) := S \backslash \mathrm{Sat}(\varphi);$$
$$\mathrm{Sat}(\varphi_1 \wedge \varphi_2) := \mathrm{Sat}(\varphi_1) \cap \mathrm{Sat}(\varphi_2).$$

Next-state formulae

In order to compute $\mathrm{Sat}(\mathcal{P}_{\bowtie p}\,[\mathcal{X}\,\varphi])$, we only need to consider direct successor states of s that satisfy φ, thus we have:

$$p_s^{\max}(\mathcal{X}\,\varphi) := \max_{a \in A} \left\{ \sum_{s' \in \mathrm{Sat}(\varphi)} p(s,a)(s') \right\}.$$

Until formulae

In order to compute $\mathrm{Sat}(\mathcal{P}_{\bowtie p}\,[\varphi_1\,\mathcal{U}\,\varphi_2])$, we need to compute $p_s^{\max}(\varphi_1\,\mathcal{U}\,\varphi_2)$. We first divide the set of states S into three disjoint subsets. The sets S^{no} and S^{yes} contain all states with probability 0 and 1, respectively, and the set $S^{?}$ contains the remaining states. If we are

interested in $p_s^{\max}(\varphi_1 \,\mathcal{U}\, \varphi_2)$, we have:

$$S^{\text{no}} := \text{Prob0A}(\text{Sat}(\varphi_1), \text{Sat}(\varphi_2)),$$
$$S^{\text{yes}} := \text{Prob1E}(\text{Sat}(\varphi_1), \text{Sat}(\varphi_2)),$$
$$S^{?} := S \backslash (S^{\text{no}} \cup S^{\text{yes}}).$$

The functions Prob0A and Prob1E compute the set of states with probability 0 for all policies and the set of states with probability 1 for some policy, respectively. They are calculated using *precomputation algorithms* (see, for example, [84]).

For $s \in S^{\text{no}}$ and $s \in S^{\text{yes}}$, $p_s^{\max}(\varphi_1 \,\mathcal{U}^{\leq k}\, \varphi_2)$ is trivially 0 or 1, respectively. For $s \in S^{?}$, the probabilities are defined recursively. For maximum probabilities, we have $p_s^{\max}(\varphi_1 \,\mathcal{U}\, \varphi_2) = \lim_{n\to\infty} p_s^{\max(n)}(\varphi_1 \,\mathcal{U}\, \varphi_2)$ where:

$$p_s^{\max(n)} := \begin{cases} 1 & \text{if } s \in S^{\text{yes}} \\ 0 & \text{if } s \in S^{\text{no}} \\ 0 & \text{if } s \in S^{?} \text{ and } n = 0 \\ \max_{a \in A} \left\{ \sum_{s' \in S} p(s,a)(s') \cdot p_s^{\max(n-1)} \right\} & \text{if } s \in S^{?} \text{ and } n > 0 \end{cases}.$$

Bounded-until formulae

The computation of $\text{Sat}(\mathcal{P}_{\bowtie p}\,[\varphi_1 \,\mathcal{U}^{\leq k}\, \varphi_2])$ is similar to the case of unbounded-until formulae. We first divide the set of states S into three disjoint subsets:

$$S^{\text{no}} := S \backslash (\text{Sat}(\varphi_1) \cup \text{Sat}(\varphi_2)),$$
$$S^{\text{yes}} := \text{Sat}(\varphi_2),$$
$$S^{?} := S \backslash (S^{\text{no}} \cup S^{\text{yes}}).$$

For $s \in S^{\text{no}}$ and $s \in S^{\text{yes}}$, $p_s^{\max}(\varphi_1 \,\mathcal{U}^{\leq k}\, \varphi_2)$ is trivially 0 or 1, respectively. For $s \in S^{?}$, the probabilities are defined recursively. If $k = 0$, then $p_s^{\max}(\varphi_1 \,\mathcal{U}^{\leq k}\, \varphi_2) = 0$. If $k > 0$, then:

$$p_s^{\max}(\varphi_1 \,\mathcal{U}^{\leq k}\, \varphi_2) := \max_{a \in A} \left\{ \sum_{s' \in S} p(s,a)(s') \cdot p_{s'}^{\max}(\varphi_1 \,\mathcal{U}^{\leq k-1}\, \varphi_2) \right\}.$$

Cumulated-reward formulae

In order to compute $\mathrm{Sat}(\mathcal{R}_{\bowtie r}\,[\mathrm{C}^{\leq k}])$, we use recursion over k. If $k = 0$, then $e_s^{\max}(X_{\mathrm{C}^{\leq k}}) = 0$. If $k > 0$, then:

$$e_s^{\max}(X_{\mathrm{C}^{\leq k}}) := \max_{a \in A} \left\{ \sum_{s' \in S} p(s,a)(s') \cdot \big(r(s,a,s') + e_{s'}^{\max}(X_{\mathrm{C}^{\leq k-1}})\big) \right\}.$$

Reachability-reward formulae

In order to compute $\mathrm{Sat}(\mathcal{R}_{\bowtie r}\,[\mathrm{F}\,\varphi])$, we again divide the set of states S into three disjoint subsets. If we are interested in $e_s^{\max}(X_{\mathrm{F}\,\varphi})$, we have:

$$\begin{aligned} S^0 &:= \mathrm{Sat}(\varphi), \\ S^\infty &:= \mathrm{Sat}(\neg \mathcal{P}_{\geq 1}\,[\mathrm{true}\,\mathcal{U}\,\varphi]), \\ S^? &:= S \backslash (S^0 \cup S^\infty). \end{aligned}$$

Similar to the case of unbounded-until formulae, for $s \in S^0$ and $s \in S^\infty$, $e_s^{\max}(X_{\mathrm{F}\,\varphi})$ is trivially 0 or ∞, respectively. For $s \in S^?$, the expected rewards are defined recursively. For maximum expected rewards, we have $e_s^{\max}(X_{\mathrm{F}\,\varphi}) = \lim_{n\to\infty} e_s^{\max(n)}(X_{\mathrm{F}\,\varphi})$ where:

$$e_s^{\max(n)} := \begin{cases} \infty & \text{if } s \in S^\infty \\ 0 & \text{if } s \in S^0 \\ 0 & \text{if } s \in S^? \text{ and } n = 0 \\ \max_{a \in A} \Big\{ \sum_{s' \in S} p(s,a)(s') & \\ \quad \cdot \big(r(s,a,s') + e_s^{\max(n-1)}\big) \Big\} & \text{if } s \in S^? \text{ and } n > 0 \end{cases} .$$

2.1.7 Quantitative analysis

Qualitative properties, that is, properties which evaluate to boolean truth values, can be expressed in PCTL and used for checking DTMCs and MDPs (as described in Definitions 2.13 and 2.15).

If the outermost operator of a PCTL formula is $\mathcal{P}_{\bowtie p}$ or $\mathcal{R}_{\bowtie r}$, it can be replaced by an operator that refers to the actual probability or expected reward value, resulting in a

quantitative property. For MDPs, these values can vary for different policies, and therefore quantitative operators for MDPs specifically refer to minimum and maximum values over all policies.

For a state $s \in S$ of a DTMC, the new operators $\mathcal{P}_{=?}$ and $\mathcal{R}_{=?}$ are defined as follows:

$$\mathcal{P}_{=?}[\psi](s) \stackrel{\text{def}}{=} p_s(\psi);$$
$$\mathcal{R}_{=?}[\phi](s) \stackrel{\text{def}}{=} \mathrm{E}_s(X_\varphi).$$

For a state $s \in S$ and policies π of an MDP, the new operators $\mathcal{P}_{\min=?}$, $\mathcal{P}_{\max=?}$, $\mathcal{R}_{\min=?}$, and $\mathcal{R}_{\max=?}$ are defined as follows:

$$\mathcal{P}_{\min=?}[\psi](s) \stackrel{\text{def}}{=} \inf_\pi \left\{p_s^\pi(\psi)\right\};$$
$$\mathcal{P}_{\max=?}[\psi](s) \stackrel{\text{def}}{=} \sup_\pi \left\{p_s^\pi(\psi)\right\};$$
$$\mathcal{R}_{\min=?}[\phi](s) \stackrel{\text{def}}{=} \inf_\pi \left\{\mathrm{E}_s^\pi(X_\varphi)\right\};$$
$$\mathcal{R}_{\max=?}[\phi](s) \stackrel{\text{def}}{=} \sup_\pi \left\{\mathrm{E}_s^\pi(X_\varphi)\right\}.$$

The analysis of formulae using these new operators does not require additional computational effort, as the value of the probability or expected reward in a quantitative formula is calculated in the same way as in a probability threshold formula.

2.1.8 Complexity

As all temporal operators of PCTL are defined recursively, in the worst case each property has to be computed for each of its logical operators and for each state of the model. To evaluate until and reachability-reward formulae for DTMCs, systems of linear equations (in the worst case, one for each state), have to be solved. For MDPs, linear optimisation problems (in the worst case, one equation for each state and each action), have to be solved.

Thus, the time required for model-checking a PCTL formula against a DTMC or an MDP is linear in the size of the formula and polynomial in the size of the model [84]. For a DTMC, the latter is defined as the number of states; for an MDP, it is defined as the number of nondeterministic choices, that is, $|S| \cdot |A|$.

The size of the model, that is, the number of its states is exponential in the number

of variables that comprise each state. Even for relatively simple models, this can result in intractably large state spaces. This is commonly referred to as the *state-space explosion problem* and is considered a major weakness of model checking.

2.1.9 PRISM

There are several probabilistic model checkers, that is, tools that allow to check whether a given probabilistic model satisfies a given temporal-logic property. The probabilistic model checker *PRISM* [60, 80] has been developed at the University of Birmingham, at the University of Oxford, and with many external contributions, since 1999.

For the description of probabilistic models PRISM uses an input language that has been derived from *Reactive Modules* [8], a language of process-algebraic expressions. In PRISM, models are described as the parallel composition of a set of modules, with each module containing a set of commands describing transitions. Each command consists of an optional action and boolean formulae over simple arithmetic expressions of variables and constants for antecedent and consequent. Variables can be local with respect to a module or global with respect to the whole model. A transition from one state to another corresponds a choice of all enabled commands. A command is enabled in a state when its antescedent holds and all other commands with the same action synchronously hold. Depending on the type of model, the choice of transition is probabilistic, nondeterministic, or both.

In contrast to other probabilistic model checkers, PRISM features state and transition rewards, quantitative analysis, symbolic data structures, and symmetry reduction. PRISM has proven to be successful in a wide range of case studies [57, 58, 80].

PRISM supports all types of models and all types of PCTL formulae used in this thesis. Probabilistic timed automata, under some restrictions, can be directly modelled as Markov decision processes [27] via the integral semantics. Due to the compositionality property of the integral semantics [62], a parallel composition of probabilistic timed automata can be modelled as the parallel composition of their respective Markov decision processes.

To evaluate PCTL formulae for a given DTMC or MDP, PRISM uses graph-theoretical techniques to analyse reachability and numerical solution techniques to solve linear equation systems and linear optimisation problems. For linear equation systems, techniques such as Jacobi, Gauss-Seidel, and successive over-relaxation are used; for linear optimisation problems,

techniques such as value iteration are employed. Detailed presentations of these techniques can be found, for example, in [59] and [84].

DTMCs and MDPs are usually represented using square, real-valued matrices. Due to the state-space explosion problem, these matrices are often very large. However, they typically contain relatively few non-zero entries. In order to exploit this, PRISM manages these models using a compact data structure called *sparse matrices*, which stores only non-zero entries explicitly.

The model-checking algorithms use matrix-vector multiplications much more frequently than vector-matrix multiplications. Therefore, we use *row-major* sparse matrics, a variant where the entries are ordered by row index. The entries of sparse matrices for DTMCs are stored using separate rows for row index, column index, and value. There are pointers from each entry in the row-index array to one in the column-index array and from each entry in the column-index array to one in the value array. In order to represent MDPs, an additional array for non-deterministic choice is added between row-index and column-index array. More details about the implementation can be found, for example, in [78].

In order to reduce the memory requirements even further, PRISM supports symbolic data structures based on *binary decision diagrams (BDDs)* and in particular a generalisation of them, *multi-terminal BDDs (MTBDDs)*. PRISM provides a choice of three computation engines [60]: First, the "MTBDD" engine, which uses a symbolic representation, has the lowest memory requirements and is most suitable for large models containing a large degree of regularity. Second, the "sparse" engine, which uses a sparse-matrix representation, is the fastest engine but has the highest memory requirements and is most suitable for irregular models. Third, the "hybrid" engine, which uses extensions of MTBDDs, is faster than that engine and requires less memory than the "sparse" engine.

2.2 Wireless network protocols

In this section, we give a brief overview of wireless network protocols. My presentation follows [91].

Computer networks can be classified using different criteria, most importantly transmission technology and scale. In terms of transmission technology, one distinguishes broadcast

links and point-to-point links. With broadcast communication, each message sent by one node is received by all node of the network, while with point-to-point communication, each message is received by exactly one receiver.

In wireless networks, all transmissions are via broadcast, and they are received by all network node that are listening to the channel and located within sufficient range to distinguish the transmitted signal from background noise. This depends on characteristics of the channel (for instance, medium), properties of the sender's radio (for instance, signal strength), and properties of the receiver's radio (for instance, receiver sensitivity).

With respect to scale, different types of networks are distinguished by their communication range. For wireless networks, Table 2.1 shows the relevant classes and the respective communication standards. The lowest range have wireless personal area networks (WPANs)

Range	**Category**	**Communication standards**
1 m	wireless personal area networks	Bluetooth, IEEE 802.15.4, ZigBee
1 km	wireless local area networks	IEEE 802.11
1000 km	wireless wide area networks	GSM, UMTS, LTE, IEEE 802.16

Table 2.1: Classification of wireless networks with respect to scale [91]

with a few metres; the main WPAN communication standards are Bluetooth, IEEE 802.15.4, and ZigBee. A higher range have wireless local area networks (WLANs) with up to several hundred metres; the main WLAN communication standard is IEEE 802.11. The highest range have wireless wide area networks (WWANs) with up to 1,000 kilometres; the main WWAN communication standards are GSM, UMTS, LTE, and IEEE 802.16 (WiMAX).

2.2.1 Medium access control

Network protocols can be distinguished by the approaches they use for medium access control, that is, to regulate which node may transmit data at any given time. Rana et al.'s paper [82] provides a good overview of existing mechanisms and distinguishes three types of *schemes*. In *random access* schemes (for instance, IEEE 802.11[6] and ALOHA[7]), the medium is accessed randomly whenever required, usually employing a contention resolution mechanism such as CSMA-CA. In *scheduled access* schemes (for instance, STDMA[8] and CATA[9]), access to the

[6]IEEE 802.11 is a standard for wireless local area networks.

[7]ALOHA is not an acronym but stands for the protocol's origin at the University of Hawaii and is described in [1].

[8]STDMA stands for *Self-Organized Time Division Multiple Access* and is described in [40].

[9]CATA stands for *Collision Avoidance Time Allocation* and is described in [92].

medium follows a pre-determined schedule. There also exist several *hybrid access* schemes (for instance, ADAPT[10] and Z-MAC[11]), which combine elements of both scheduled and random access.

For scheduled and hybrid access schemes, the underlying time schedule splits time into slots. In each slot, at most one node in the network is allowed to transmit. An agreed set of mechanisms for managing and allocating slots to network nodes is called *slot allocation protocol*. Slot allocation protocols where allocation is based not just on fixed schedules and pure randomness, but also depends on the behaviour of network nodes, are called *dynamic*.

In dynamic slot allocation protocols, the way in which nodes actively influence the allocation of time slots is called *policy*. Each node uses a policy to control when requests are made or acknowledged. Policies can be deterministic, nondeterministic or even random. If a policy takes into account contextual factors, it is called *adaptive*. The main purpose of adaptive policies is to optimise bandwidth usage.

2.2.2 Flooding and gossiping

An important task in wireless networking is routing, that is, transporting transmitted data packets from the sender to its destination. This involves using intermediate nodes to receive and forward data. In order to choose suitable paths through the network, the network topology is usually taken into account.

A very simple routing mechanism for wireless networks is *flooding*, where each node broadcasts each incoming packet to each other node. This involves a lot of redundancy and is therefore unwanted for many applications, but very useful for others such as wireless networks with frequently unavailable nodes, unknown topologies, or unreliable transmissions.

There exist several variants of flooding, which use mechanisms to limit the retransmission of packets, for example by imposing a maximum number of retransmission for each packet or by logging the packets retransmitted by each station. Variants that resemble communication in social networks, especially periodic exchanges of data where messages are forwarded with a certain probability, are referred to as *gossip protocols* [35, 55], although *flooding* and *gossiping* are often used synonymously.

[10] ADAPT stands for *A Dynamically Adaptive Protocol for Transmission* and is described in [74].
[11] Z-MAC stands for *Zebra MAC* and is described in [83].

2.3 Discrete-event simulation

In this section, we give a brief introduction to discrete-event simulation, its application to wireless networks, and the tool Castalia.

Simulation [10, 14] is a non-formal method for the automatic validation and quantitative analysis of systems. Its objective is, based on a mathematical model of a system that can be controlled by parameters and initial conditions, to generate representative scenarios for the systems's behaviour. The quantitative properties that can be analysed are simpler than those that can be analysed using probabilistic model checking. Alternatively to simulating a whole system it is also possible to simulate only a subset of a system's components, for example stimuli of some inputs. Simulation is widely used in many areas of science, technology, economics, and others.

Simulation is preferred over analytical methods where a complete enumeration of all model states is prohibitive, where obtaining a compact mathematical representation or its solution is infeasible, and where modelling assumptions of analytical models can lead to inaccurate results. However, the results of simulation-based analyses are only as good as the model and the data used. Simulation results are only correct for the simulation runs conducted, while model checking results are correct with respect to the full behaviour of the model. Establishing reliable results usually requires several simulation runs, and, depending on the complexity of the model and the number of variables to be optimised, the resource requirements can be very high.

In discrete-event simulation [14], a simulation run is described by the state of the model at discrete time steps, where state changes are called *events*. Discrete-event simulation requires a clock, a set of pending events, random-number generators, ending condition, and statistics/reporting.

2.3.1 Monte-Carlo simulation

Monte-Carlo simulation [10] is a simulation method that uses sampling of probability distributions to determine input parameters of a simulation. For stochastic processes that are very large or very complex, Monte-Carlo simulation is often still feasible when other analysis are not viable.

2.3.2 Wireless-network simulation

Wireless networks often consist of many components and their complex interactions. Network simulators [38] can typically represent the network topology and properties of nodes, as well as links and traffic between nodes. Often, hierarchies of nodes (for example, masters and slaves), types of nodes (for example, hubs and routers), and protocol details can also be specified. The traffic between nodes is usually described by environment parameters and probability distributions. Models for the wireless channel may include fading, loss, interference, and mobility. Models for the radio may include reception probability (based on transmission power, reception sensitivity, modulation type, medium/environment, power consumption, delay).

Wireless networks simulators are usually based on discrete-event simulation. Three of the most widely used wireless-network simulators are *ns-2* [76], *OPNeT Modeler* [77], and *GloMoSim* [68]. All three are based on discrete-event simulation.

As the main objective of simulation is the generation of realistic scenarios for a system's behaviour, the quality of wireless-network simulators varies. Several general weaknesses of wireless-network simulators have been shown in the literature [20, 35, 52]. Their underlying mathematical models are often non-transparent, unrealistic, and not validated against empirical data. Consequently, simulation results vary widely between different simulators and also compared to field experiments [35, 77]. Furthermore, simulators generally do not adequately support nondeterminism, which is sometimes interpreted probabilistically, again leading to unrealistic results.

Cavin et al. [20] say that a good wireless-network simulator has to be able to represent realistic protocols that include interaction over more than one network layer, to use realistic parameters for connectivity, power ranges, and mobility schemes, and to use realistic initial conditions. For a simple algorithm where each node just forwards each message received for the first time to its neighbours, they report that the three simulators ns-2, OPNeT Modeler and GloMoSim all produce "very different" and "barely comparable" results. As a general reason for the lack of comparability, the authors establish the different subsets of conceivable environment parameters and possible abstractions used by each implementation of a wireless-network simulator. However, other reasons mentioned include individual weaknesses of the

simulators, such as the cumbersome requirement in OPNET Modeler to define each feature as a finite state machine, missing documentation, inconsistent code, unavailability of generalised analysis tools, a steep learning curve in ns-2, and particular difficulties to model protocols over multiple network layers in GloMoSim. The authors also note that, in each new release of one of these simulators, new errors in the implementation of lower network layers are corrected, even for such elementary network protocols such as IEEE 802.11.

Kotz et al. [52] criticise the use of unrealistically simple assumptions in the radio models of many simulators, such as "the world is two dimensional", "a radio's transmission area is roughly circular", "all radios have equal range", "if I can hear you, you can hear me", "if I can hear you at all, I can hear you perfectly", and "signal strength is a simple function of distance". Realistic features such as hills, obstacles, link asymmetries, and unpredictable fading are often ignored. The authors recommend the use of asymmetric links, better exploration of model parameters, and the consideration of hills and valleys. More generally, they recommend the validation with real input data, and supporting the use of the real system's code in simulators. Models should be compatible with different environments and optimally combine physical and link layer.

Fehnker and Gao [35] combine probabilistic model checking (PRISM) with Monte-Carlo simulation (MATLAB) for a comparative study of simple gossip protocols. While model checking gave very precise performance results for small networks, simulation was able to show performance effects of large networks with sufficient detail.

Zuniga and Krishnamachari [102] describe state-of-the-art models for wireless channel and radio, which have been successfully validated against empirical data. In particular, connectivity is not just considered discretely (with the states *disconnected* and *connected*), but also the transitional region between them, which is critical for the modelling of unreliable links, has been represented very realistically.

2.3.3 Castalia

The wireless network simulator Castalia [15, 79] has been developed at National ICT Australia since 2006. It has been implemented in the simulation framework OMNeT++. In contrast to other wireless network simulators, Castalia features a more realistic channel and radio model [86, 102]; flexible modelling of physical processes; device noise, bias, and power

consumption; node clock drift; resource monitoring (for example memory and CPU time); and implementations of several MAC and routing protocols. It is not aimed at specific applications or for simulating or emulating specific code, but as a generic platform for the validation of algorithms.

Castalia's channel model represents fading phenomena more realistically, including the temporal variation of fading. It uses *lognormal shadowing model*[12] [102] to model average path loss and utilises empirically measured data to model the temporal variation of path loss.

Castalia's radio model dynamically calculates packet reception probabilities depending on interference and transmission power at runtime.

In order to simulate a network protocol using Castalia, a description of the protocol in NED (the input language of OMNeT++) and an implementation of its behaviour in C++ are needed. To date, Castalia already supports a range of different network protocols.

2.4 A sample device

In this section, we provide information about a wireless communication device, which is referred to in the Chapters 4, 5, and 6.

In order to reason about energy consumption of a device, it is necessary to have reliable data about the device's power states and transitions. In this thesis, we have studied energy characteristics on the example of the IEEE 802.15.4/ZigBee transceiver device Texas Instruments CC2520, which we have modelled using information from its datasheet [49].

This device provides two power states for receiving, one for transmitting, and three for the radio turned off, as shown in Tables 2.2 and 2.3. The transition times are explicitly

Operation mode	**Typical power consumption**
receiving (waiting for synchronisation)	66.9mW
receiving (receiving frame)	55.5mW
transmitting	77.4mW
active mode	4.8mW
low-power mode 1 (keeping state)	525μW
low-power mode 2 (not keeping state)	90nW

Table 2.2: Power levels of Texas Instruments CC2520 [49]

[12] In the literature, the lognormal shadowing model is also referred to as "log-distance path loss model".

Operation mode	**Time**
low-power mode 2 → AM	0.3ms
low-power mode 1 → AM	0.2ms
active mode → RX	192μs
active mode → TX	192μs
receiving → TX turnaround	192μs
transmitting → RX turnaround	192μs

Table 2.3: Timing behaviour of Texas Instruments CC2520 [49]

stated in the datasheet. The power level P of each power state has been calculated from the default supply voltage V of 3.0 volt and the state's current consumption I, using Joule's law as $P = V \cdot I$.

Chapter 3

Combining simulation and model checking for wireless sensor networks

In this chapter, we explore ways of combining simulation and probabilistic model checking in order to improve the design of wireless sensor networks (WSNs) and their protocols. Algorithms for automated generation of models for both analysis methods have been developed and implemented in a tool, which has been successfully applied to different scenarios of flooding and gossip protocols. This work has been carried out jointly with Annabelle McIver, Ansgar Fehnker, and Athanassios Boulis.

Previously, Fehnker and Gao [35] had proposed to build a formal model and a simulation model from the same description and apply both model checking and Monte-Carlo simulation, in order to obtain more realistic results than with current approaches.

The work on the modelling framework has previously been published at MeMoT 2007[1] [33] and as a book chapter in the Lecture Notes in Computer Science series [34]. The graphical specification tool CaVi has been presented at NICTA Techfest 2007 [32] and at QEST 2008[2] [16]. These presentations have been extended by a new, more detailed example of the theoretical framework and the tool.

[1] MeMoT 2007 stands for the Workshop on Methods, Models and Tools for Fault Tolerance at the 7th International Conference on Integrated Formal Methods (IFM 2007).

[2] QEST 2008 stands for the 5th International Conference on the Quantitative Evaluation of Systems.

Formal methods provide powerful specification formalisms and exhaustive analysis (exact results with respect to models), for example, via model checking, but suffer from state space explosion.

Simulation supports larger models, but its semantics is often non-transparent or unintuitive (cf. Section 2.3.2). Its analysis is non-exhaustive and costly due to the high number of simulation runs required.

We present the graphical specification tool CaVi, which can generate PRISM models and perform Monte Carlo simulation of Castalia models, based on a uniform description and a generic model template for each tool. It uses realistic wireless communication models and, if the translation to PRISM is used, has a clear, formal semantics. The model size depends on the analysis mechanism used. Analyses are exhaustive if the analysis tool used is PRISM.

Monte Carlo simulation of PRISM models relies on the clear semantics available via CaVi and PRISM, but keeps models small and simple, thus further reducing the verification time.

This chapter is divided into four sections. The next section introduces different interference models and the analytical model for modelling wireless communication. Section 3.2 describes the graphical specification tool CaVi and its coupling to the probabilistic model checker PRISM and the wireless-sensor-network simulator Castalia. Section 3.3 demonstrates the modelling framework and the tool on an example. Section 3.4 concludes this chapter.

3.1 Modelling wireless communication

One of the main weaknesses of both formal and non-formal approaches in wireless-network modelling is the use of oversimplified communication models, which lead to unrealistic analysis results. Wireless communication has two main aspects: channel and radio. Hereafter, the term *radio* refers to the physical properties of sending and receiving hardware, including transmission power levels, reception sensitivity, modulation type, and data rate. The term *channel* refers to properties of the transmitted signals such as signal strength, fading, loss, and interference; it also includes physical properties of the environment that influence signal transmission, including noise bandwidth, obstacles between sender and receiver, and asymmetric links.

3.1.1 Interference models

Interference models describe whether a message transmitted by a station in a wireless network can be received by another station of that network. This depends on various factors relating to channel and radio, including geographical positions and signal strengths of sending and receiving stations.

In order to illustrate this, we consider simple networks of two stations that start transmissions at the same time. Let us assume that the physical properties of the stations' radios are identical (that is, data rate, signal strength, receiver sensitivity, etc., are equal for all of them). Let us also assume that the channel is uniform (that is, background noise, fading, etc., are the same for the whole network). Figure 3.1 shows three different scenarios. A

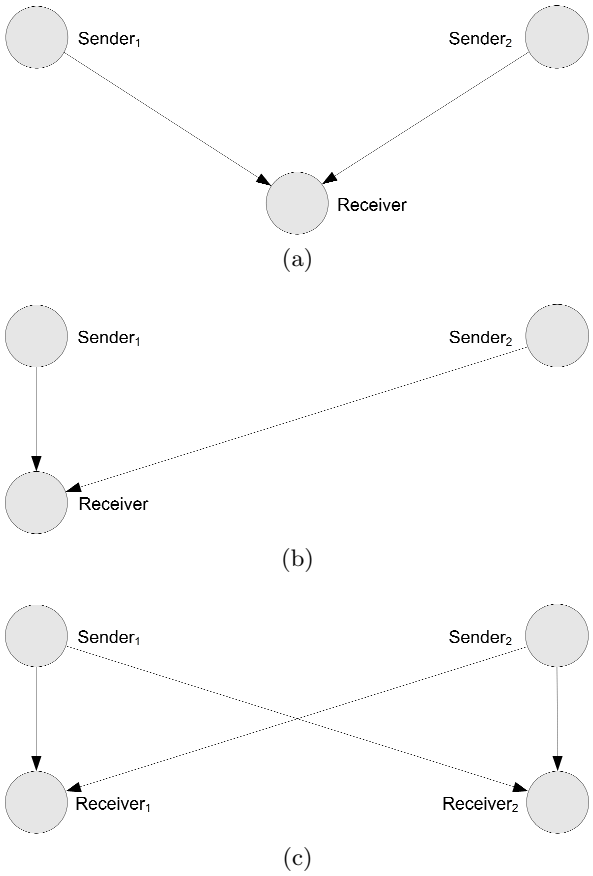


Figure 3.1: Scenarios of sending and receiving stations

single receiving station with equal distance to each sending station (Figure 3.1(a)) receives both sending stations' transmissions at the same signal strength and is thus unable to distinguish them using this measure. A single receiving station with asymmetric distances to the sending stations (Figure 3.1(b)) receives Sender_1's transmission with a higher signal strength

than Sender_2's transmission, thus could decode Sender_1's message by regarding Sender_2's transmission as noise. When there are two receiving stations (Figure 3.1(c)), the interference situation is identical to the previous one. Note that interference also takes place when there is only one station sending and the receiver has to distinguish that station's transmission from the background noise.

Hence, whenever two stations in a wireless network transmit at the same time, these transmissions interfere. There are two common models for interpreting this. The first one is the *collision-free model*: it represents the best-case assumption that collisions never occur. The second one is the *collision model*: it represents the worst-case assumption that collisions always occur. More realistic models take into account the different sources of interference as well as the capabilities of the receiver. In the so-called *additive-interference model*, a collision occurs when a receiving station cannot clearly distinguish between a signal and noise (depending on topology, number and strength of incoming transmissions, and sensitivity of reception).

In all models, sending and receiving stations do not directly notice whether collisions occur. Nevertheless, they can conclude this if acknowledgements for sent messages are part of the communication protocol.

3.1.2 An analytical model for channel and radio

One of the best analytical models for wireless communication has been developed by Zuniga and Krishnamachari [102] and empirically validated by Seada et al. [86]. In this additive-interference model, the probability that a node receives a message depends on various parameters of both channel and radio. In the remainder of this subsection, we follow the presentations in [34] and [102]. Most of the formulae have previously been published in [34].

We first introduce abstractions for the wireless channel. In order to describe the rate at which the quality of a radio signal decreases after its propagation, we used the established *lognormal shadowing model*. The power of a received signal depends on the power of the transmitted signal and the power lost during transmission; in wireless networks, the latter is commonly referred to as *path loss*. The path loss is influenced by the distance between sender and receiver and the environmental factors such as propagation medium, terrain, and obstacles.

Let i and j be two nodes in an arbitrary network, located a distance $d(i,j)$ apart from each other; let tx_j be the transmission strength of node j in decibel. Let d_0 be a reference distance; let $PL(d_0)$ be the path loss in decibel at this reference distance, representing a generic quality loss due to a signal transmission. Let PLE be the *path loss exponent*, describing the change in signal strength with respect to distance.

According to the definition of the unit decibel, the ratio in decibel (dB) v_{dB} of a value v to a reference value v_0 is given as ten times the decadic logarithm, that is, $v_{dB} \stackrel{\text{def}}{=} 10 \log_{10}(d/d_0)$. As all quantities in this formula are given in decibel, the distance-dependent path loss in decibel is computed as ten times the decadic logarithm of the distance ratio $d(i,j)/d_0$.

Definition 3.1. The *signal strength for a received message* in decibel is defined as

$$rxdB_{i,j} \stackrel{\text{def}}{=} tx_j - PL(d_0) - 10\,PLE\,\log_{10}(d(i,j)/d_0).$$

From this, the power at the receiver can be computed by

$$rx_{i,j} \stackrel{\text{def}}{=} 10^{rxdB_{i,j}/10}. \tag{3.1}$$

The background noise comprises the constant noise $nbgN$ inherent to the channel and the noise originating from neighbouring nodes' transmissions. Let $send_i$ be a function that returns 1 if node i is transmitting a message and 0 otherwise.

Definition 3.2. The *background noise* at receiver i with respect to a message transmitted by j is defined as

$$bgN_{i,j} \stackrel{\text{def}}{=} nbgN + \sum_{k \neq i,j} rx_{i,k} \cdot send_k.$$

The quality of an incoming signal with respect to existing noise at the receiver is commonly described using the signal-to-noise ratio, which is formulated as the ratio of received-signal strength to background noise as follows.

Definition 3.3. The *signal-to-noise ratio* for a transmission from j to i is defined by

$$SNR_{i,j} \stackrel{\text{def}}{=} \frac{rx_{i,j}}{bgN_{i,j}}.$$

In order to compute link probabilities, abstractions for the wireless radio need to be computed. Let nDR be the data rate, nBW be the noise bandwidth, and f be the frame size in bits. For FSK modulation, the probability for a bit error is known [102] as $\frac{1}{2} \cdot \mathrm{e}^{-\frac{1}{2} \cdot \frac{nBW}{nDR} SNR_{i,j}})$. Consequently, the *threshold-free reception probability* for a message of length f transmitted from j to i can be computed by

$$snr2prob(SNR_{i,j}) = (1 - \frac{1}{2} \cdot \mathrm{e}^{-\frac{1}{2} \cdot \frac{nBW}{nDR} SNR_{i,j}})^{8f}. \tag{3.2}$$

According to the interference model by Zuniga and Krishnamachari [102], messages cannot be distinguished from background noise if the signal-to-noise ratio is below a given threshold, which is specific to data rate, noise bandwidth, frame size, and modulation type; in this case, the reception probability of a message is zero. Let nTP be the minimum packet-reception probability, that is, if the probability to receive a message of length f is less than nTP, it is assumed that the receiver cannot distinguish it from background noise.

Definition 3.4. For FSK modulation, the *signal-to-noise-ratio threshold* is defined by

$$\Delta_{i,j} \stackrel{\text{def}}{=} -2\frac{nDR}{nBW} \ln\left(2(1 - nTP^{\frac{1}{8f}})\right).$$

Finally, considering the signal-to-noise-ratio threshold and the threshold-free reception probability, the respective *reception probability* for i receiving a message from j is given by

$$precv_{i,j} \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } SNR_{i,j} < \Delta_{i,j} \\ snr2prob(SNR_{i,j}) & \text{otherwise} \end{cases}. \tag{3.3}$$

As the computation of the probability $precv_{i,j}$ considers transmissions from nodes other than j as noise, it is a conditional probability. In particular, it is not possible to have $precv_{i,j} > 0$ and $precv_{i,k} > 0$ with $j \neq k$. Thus the probability P_i that i receives a message from any other node can simply be computed by

$$P_i = \sum_{j \neq i} precv_{i,j} \cdot send_j. \tag{3.4}$$

3.2 Interfacing wireless communication models with simulation and probabilistic model checking

3.2.1 The graphical specification tool CaVi

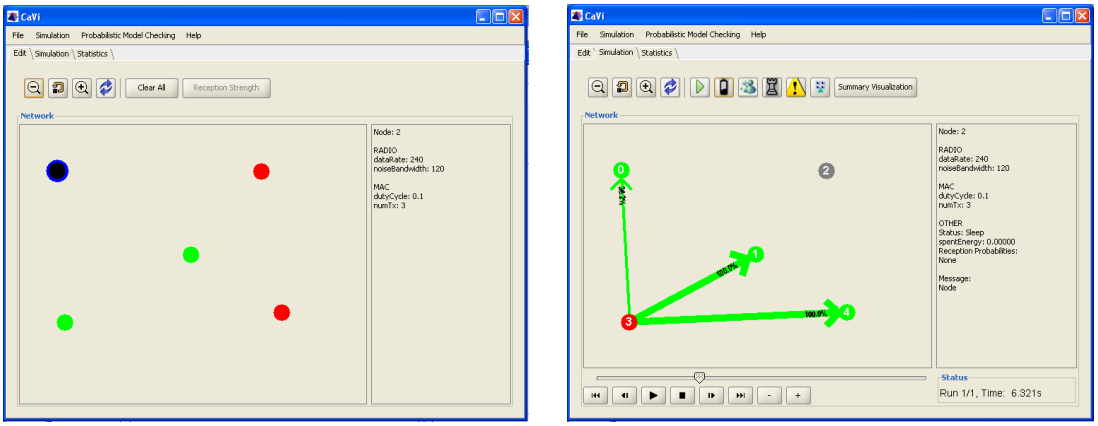
The wireless communication models presented in the previous section have been integrated into a new tool for graphical specification of wireless networks: CaVi, which stands for "Castalia visualiser". CaVi has been implemented in the programming language Java. The representation of nodes, networks, and other relevant entities is based on data structures from the Java Collections Framework (such as lists, sets, trees, and queues). The graphical user interface is based on the Java Swing toolkit. In order to store nodes and networks as well as parameters of channel and simulation environment, XML representations are used. In order to read and write model files and result files of Castalia ad PRISM, parsers based on regular expressions are used.

Based on functionality of PRISM and Castalia, CaVi provides several features to increase efficiency and correctness in design and analysis of wireless sensor networks. Moreover, it can automatically generate models for PRISM and Castalia from graphical specifications of wireless sensor networks.

In the design stage, new networks can be created and modified in a drag-and-drop fashion. Besides the graphical specification of nodes' geographical positions, several performance measures can be visualised on-the-fly during specification of a network, most importantly worst-case and best-case reception probabilities, which are calculated using Equation 3.3. In the analysis stage, simulation and model-checking of the specified networks can be performed and the results can be visualised.

Figure 3.2 shows some of CaVi's visual capabilities. In Figure 3.2(a), the top-left node has been selected such that its reception probability with respect to messages sent by each of the other nodes is indicated by green, yellow, and red colours (corresponding to high, medium, and low, respectively). In Figure 3.2(b), the step-by-step simulation mode is shown; it can be seen that the bottom-left node is sending to all other nodes except the top-right one, which is out of range; the reception quality for each individual link is highlighted by the thickness of the respective arrow and by a percentage value.

Given the graphical representation of a wireless sensor network, aspects of realistic chan-



(a) (b)

Figure 3.2: Visualisation of performance indicators in CaVi (taken from [34])

nel models such as distance and density can be intuitively specified using just graphical means. Aspects of realistic radio models such as transmission power and reception sensitivity are beyond the capabilities of the graphical user interface. However, they can be easily specified using separate dialog boxes; this includes parameters of individual nodes, whole networks, and the environment.

For the first time, realistic wireless channel and radio models based on [102] have been integrated into probabilistic model checking in [33]. By automatically generating PRISM models for flooding and gossip protocols, CaVi removes the hitherto lengthy and error-prone tasks of computing and incorporating probabilistic abstractions into existing models. For other protocols, which are not yet supported directly by CaVi, the remaining manual modifications are much smaller and less expensive when using CaVi than without. With Castalia, a powerful discrete-event simulator specifically designed for wireless sensor networks, many critical issues (such as suboptimal node distances or collision-prone network topologies) can already be identified at the design stage, and model-checking results can be validated quickly.

Figure 3.3 shows how CaVi, Castalia, and PRISM are connected to each other. A typical workflow for developing a new wireless sensor network may consist of the following steps: first, CaVi is used to define a graphical specification of the network. Afterwards, static aspects of the design (that is, assigning the geographical positions of the nodes and physical parameters of the radio) can be validated using the visual performance indicator provided. Dynamic aspects such as execution details of the protocol can be validated using the integrated simulation with Castalia. Following this, a PRISM model for a specific protocol

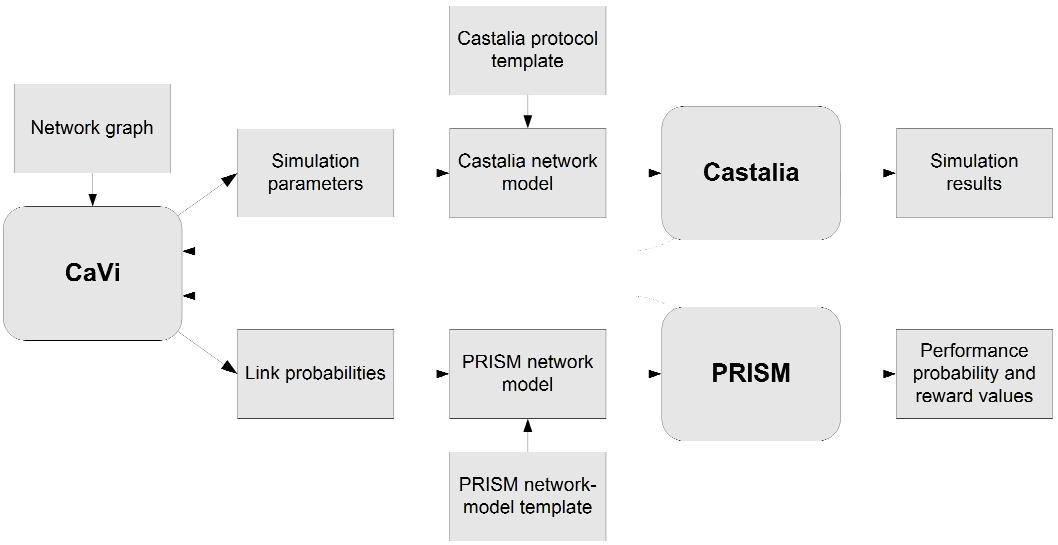


Figure 3.3: Interconnection of CaVi with PRISM and Castalia (based on Figure 8 from [34])

can be generated from an existing template (to date, flooding and gossiping are supported). Finally, PRISM can be used for further, formal analysis.

The latest version of CaVi can be obtained from National ICT Australia. Older versions are also available at `https://cgi.cse.unsw.edu.au/~formalmethods/wiki/pewna/index.cgi/TasteOfResearch`.

3.2.2 Translation to PRISM

In order to use the probabilistic model checker PRISM for the analysis of a wireless network specified via CaVi, a generic PRISM model is instrumented with the presented probabilistic abstractions for wireless channel and radio. For the computation of concrete values for these abstractions, all relevant channel and radio parameters of the network, for example signal strength and reception sensitivity, as well as the geographical positions of all nodes, are taken into account.

The basic generic PRISM model used for all the protocols is as follows:

```
const recvpi

module nodei
  activei: [0..1] init 1;
  sendi: [0..1] init 0;
  [tick] sendi=0 & activei=1 -> recvpi: (sendi'=1) & (activei'=1)
```

```
                              + (1-recvpi): (sendi'=0) & (activei'=1);
  [tick] sendi=1 & activei=1 -> sendi'=0 & activei'=0;
  [tick] activei=0 -> sendi'=0 & activei'=0;
endmodule
```

For each node i, a module `node`i is included. When the variable `active`i equals 0, the node is inactive and can only listen but not send; if it equals 1, the node is active and can both listen and send. If node i has received a message to be forwarded, the variable `send`i equals 1; otherwise, it equals 0. In this model, all nodes receive and attempt to forward messages synchronously with the action `tick`. At each point in time, the probability that node i receives a message to be forwarded is given by the constant `recvp`i. Note that CaVi can also generate models where sending takes places asynchronously, which will be used in Chapters 4 and 6. For a discussion of synchronous versus asynchronous communication links, also see [16].

As not all arithmetic functions used in our analytical model are available in PRISM, we use precomputed constants for most parameters except for the reception probabilities $precv_{i,j}$, which are conditional probabilties and thus are realised using PRISM's `formula` construct.

First, the signal strength for a received message $rx_{i,j}$ is denoted in the PRISM model by `linRxSignal_`j`_`i, and appears as a constant declaration for each pair of nodes. For example, for nodes 1 and 2, we have:

```
const double linRxSignal_2_1 = 3.162277660168379E-7;
```

Second, the signal-to-noise ratio $SNR(i, j)$ is denoted `snr_`j`_`i, and appears as a formula declaration for each pair of nodes. For example, for nodes 1 and 2 in a four-node model, we have:

```
formula snr_2_1 =
    (linRxSignal_2_1*send2)/(linRxSignal_0_1*send0 + linRxSignal_3_1*send3 + 1.0E-10);
```

Third, the reception probability $precv_{i,j}$ is denoted `Preceive_`j`_`i, and appears as a formula declaration for each pair of nodes. For example, for nodes 1 and 2, we have:

```
formula Preceive_2_1 = func(max,0,(snr_2_1>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_2_1)), 8 * 25):0);
```

Fourth, the probability P_i that a node i receives a message from any other node is denoted `recvp`i, and appears as a formula declaration for each node. Here, we take the precaution of ensuring that the sum does not exceed 1, which sometimes happens due to rounding errors. For example, for node 1, we have:

```
formula recvp1 = func(min,1,Preceive_0_1+Preceive_2_1+Preceive_3_1);
```

3.2.3 Translation to Castalia

In order to use the wireless-network simulator Castalia for the analysis of a wireless network specified via CaVi, all relevant channel and radio parameters of the network, for example signal strength and reception sensitivity, as well as the geographical positions of all nodes are exported into parameter files for Castalia. Depending on the network protocol that shall be simulated with Castalia, the respective C++ implementation is chosen.

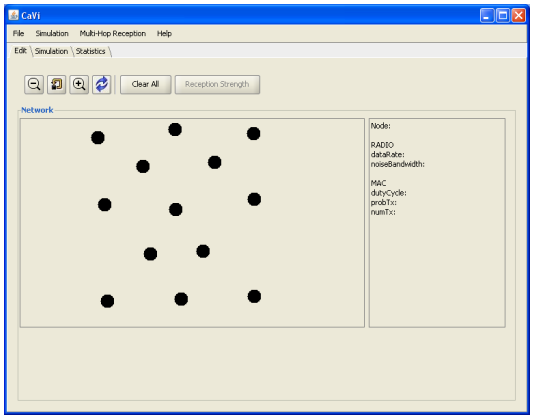
Castalia executes a given protocol description for a given network by performing a discrete-event simulation. This involves generating sequences of events that are valid with respect to Castalia's implementation of the protocol. For the simulation of the channel and radio model, as well as for random choices required by protocols, random numbers are generated to guide probabilistic choices.

3.3 Example

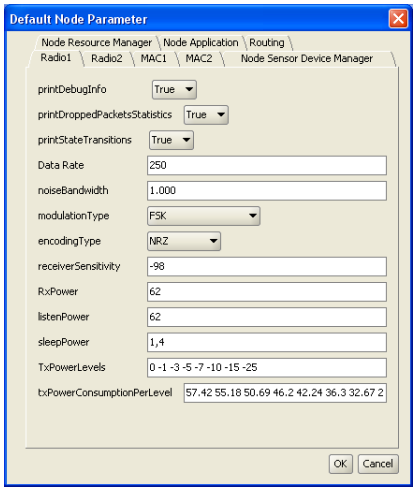
Using the example of a simple generic gossip protocol, we demonstrate how the modelling framework and the graphical specification tool CaVi are used.

First, models are specified graphically in a drag-and-drop fashion using CaVi, see Figure 3.4a. Then parameters of the nodes, the network, and the simulation engine are chosen in separate dialogs, as in Figure 3.4b for the default node behaviour.

During the graphical specification of a network, an instant visualisation of best-case and worst-case reception probabilities, based on the underlying analytical model from Section 3.1.2, is available, as shown in Figure 3.4.



(a) Model specification in CaVi



(b) Example for node parameterisation

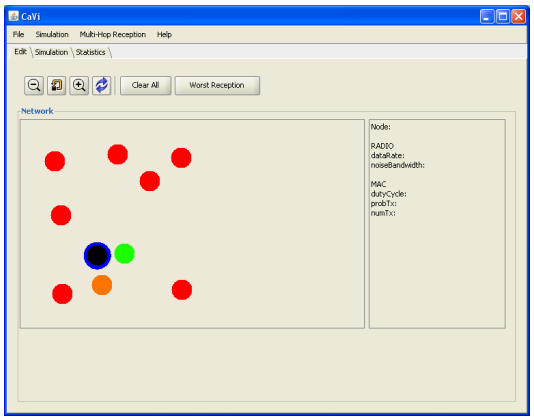


Figure 3.4: Instant visualisation of reception probabilities in CaVi

Under the simulation tab, the simulator Castalia can be invoked, and traces from its runs can be stepped through, as demonstrated in Figure 3.5.

From the CaVi model, the following PRISM model is generated.

```
nondeterministic

const double PsendingNode0 = 1.0;
...
const double PsendingNode9 = 1.0;

const double linRxSignal_0_1 = 5.598171297424829E-9;
const double linRxSignal_0_2 = 3.0939947607558865E-7;
const double linRxSignal_0_3 = 4.204565781341186E-8;
```

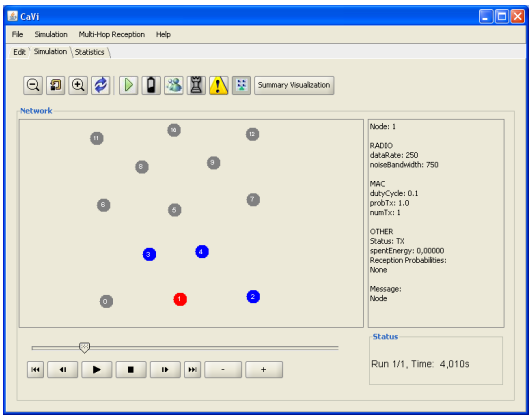


Figure 3.5: Visualisation of simulation results in CaVi

```
...
const double linRxSignal_1_0 = 5.598171297424828E-8;
const double linRxSignal_1_2 = 2.0350724876354692E-7;
const double linRxSignal_1_3 = 2.4061187321421144E-8;
...
const double linRxSignal_9_8 = 1.250076735937101E-5;

formula snr_0_1 = (linRxSignal_0_1*send0)/(linRxSignal_2_1*send2 + linRxSignal_3_1*send3
                + linRxSignal_4_1*send4 + linRxSignal_5_1*send5 + linRxSignal_6_1*send6
                + linRxSignal_7_1*send7 + linRxSignal_8_1*send8 + linRxSignal_9_1*send9 + 1.0E-10);
...
formula snr_9_8 = (linRxSignal_9_8*send9)/(linRxSignal_0_8*send0 + linRxSignal_1_8*send1
                + linRxSignal_2_8*send2 + linRxSignal_3_8*send3 + linRxSignal_4_8*send4
                + linRxSignal_5_8*send5 + linRxSignal_6_8*send6 + linRxSignal_7_8*send7 + 1.0E-10);

formula Preceive_0_1 = func(max,0,(snr_0_1>=2.059654263000424)?func(pow,(1-0.5
                        *func(pow,2.71828,-0.5 *3.0 * snr_0_1)), 8 * 25):0);
...
formula Preceive_9_8 = func(max,0,(snr_9_8>=2.059654263000424)?func(pow,(1-0.5
                        *func(pow,2.71828,-0.5 *3.0 * snr_9_8)), 8 * 25):0);

formula recvp0 = func(min,1,Preceive_1_0+Preceive_2_0+Preceive_3_0+Preceive_4_0+Preceive_5_0
                    +Preceive_6_0+Preceive_7_0+Preceive_8_0+Preceive_9_0);
...
formula recvp9 = func(min,1,Preceive_0_9+Preceive_1_9+Preceive_2_9+Preceive_3_9+Preceive_4_9
                    +Preceive_5_9+Preceive_6_9+Preceive_7_9+Preceive_8_9);
```

```
module node0
        active0:[0..2] init 1;
        send0:  [0..1] init 0;

[tick] send0=0&active0=1 -> PsendingNode0:(send0'=1)&(active0'=1)
                          +(1-PsendingNode0):(send0'=0)&(active0'=0);
[tick] send0=1&active0=1 -> send0'=0&active0'=0;
[tick] active0=0 -> send0'=0&active0'=0;
endmodule
...
module node9
        active9:[0..2] init 1;
        send9:  [0..1] init 0;

[tick] send9=0&active9=1 -> recvp9*PsendingNode9:(send9'=1)&(active9'=1)
                          + recvp9*(1 -PsendingNode9):(send9'=0)&(active9'=0)
                          + (1 - recvp9):(send9'=0)&(active9'=1);
[tick] send9=0&active9=1 -> recvp9*PsendingNode9:(active9'=2)
                          + recvp9*(1 -PsendingNode9):(send9'=0)&(active9'=0)
                          + (1 - recvp9):(send9'=0)&(active9'=1);
[tick] send9=1&active9=1 -> send9'=0&active9'=0;
[tick] active9=0 -> send9'=0&active9'=0;
[tick] active9=2 ->active9'=2;
[]     active9=2 ->send9'=1&active9'=1;
endmodule
```

The generated PRISM model follows the structure described in Section 3.2.2. Constants have been declared for signal strengths, signal-to-noise ratios, and reception probabilities according to the specified network topology, the chosen parameter values, and the underlying analytical model.

3.4 Conclusion

We have presented an analytical framework and a graphical specification tool for uniform modelling and analysis of wireless sensor networks. This is the first framework that has used realistic wireless channel and radio models in conjunction with probabilistic model

checking. The visualisation of performance measures based on on-the-fly computations and the integration of simulation and model checking tools have greatly improved design and analysis of wireless sensor networks. Owing to the uniform wireless communication models used, the PRISM model can automatically be enriched with realistic interference models, and even large models can quickly be validated using Castalia.

However, the analysis of PRISM models from wireless sensor networks specified in CaVi is limited to a network size of about 20 nodes (depending on topology and protocol). For larger networks, PRISM is unable to process the code of the generated model and thus building a state-transition model is not possible. This is due to the large size of the model code, with the number of lines being quadratic and the length of each line being linear in the number of nodes, thus the model being cubic in the number of nodes. For the future, two approaches seem promising to resolve this issue: first, adding support for more generic user-defined formulae and arithmetic functions or even for a powerful metalanguage such as XRM [26] to PRISM is likely to alleviate the parsing problem; second, implementing an improved mechanism for parsing model code and building models is likely to overcome further building problems for the large but generic models CaVi produces.

While probabilistic model checking is generally limited to the analysis of small models, it is able to analyse the precise impact of individual network parameters. Instead, Castalia would need many runs to reach an accuracy easily achievable with formal methods, but it can quickly predict general trends about the behaviour of large networks and protocols.

In order to improve the feasibility of probabilistic model checking, further work on abstraction and refinement is needed, in particular to exploit symmetries in networks where many nodes have similar physical properties or execute the same protocol.

With the continuing use of CaVi, it would be beneficial to establish an equivalence between CaVi/Castalia specifications on one hand and PRISM models on the other hand. A unifying language for describing these models, for example based on McIver's approach for action systems [70], applied to this study in [34], might be appropriate for this.

Also, the wireless communication models used should be validated through further field experiments, simulation, and analytical work.

Chapter 4

Semi-formal analysis of spatial properties for wireless sensor networks

In this chapter, we present a methodology for modelling and analysing spatial properties of wireless sensor networks. Based on the realistic wireless channel and radio models presented in Chapter 3, we analyse performance and energy properties for different network topologies and sizes. We also evaluate the suitability of probabilistic model checking for the analysis of spatial properties.

The aims of this study are as follows: first, to show the feasibility of the approach presented in Chapter 3; second, to demonstrate its effectiveness in systematically assessing different network topologies; and, third, to develop robust and reliable spatial models that can be used to enhance classic, non-interference models for quantitative analysis of the protocols studied in Chapters 5 and 6.

This work extends the hybrid approach presented in Chapter 3 to the broader analysis of spatial characteristics of wireless sensor networks. We consider both performance and energy properties of network topologies and present a methodology for comparing the quality of forwarding topologies for gossip protocols. For this purpose, we take the gossip protocol included in CaVi and compare the effect of different network topologies and parameterisations.

On the example of the IEEE 802.15.4/ZigBee transceiver Texas Instruments CC2520, the impact of network topology on energy properties (energy consumption, battery lifetime, network lifetime) has been studied. Using our tool CaVi, spatial characteristics of networks as well as advanced physical characteristics of individual network nodes such as transmission power level and reception sensitivity have been modelled.

This chapter is divided into three sections. The next section describes the network topologies considered, modelling assumptions made, and the adaptation of the wireless channel and radio models from Chapter 3. Section 4.2 describes experiments performed and discusses their outcomes. The last section concludes this chapter.

4.1 Modelling

This section describes the models analysed in this study. It describes network topologies, wireless channel and radio configuration, modelling assumptions, and the process of obtaining input models for the probabilistic model checker PRISM.

For each network topology and each node distance, a model was specified graphically within CaVi, including physical properties of nodes and network, which CaVi supports as model parameters.

For the interactive evaluation of different network topologies, Castalia features integrated in CaVi were used to simulate the respective networks. This included the computation of simulation runs for different routing scenarios and their instant visualisation in CaVi (as shown in Figure 3.5). This way, the suitability of network layouts with respect to reception probabilities, routing paths, and energy consumption, could be determined quickly.

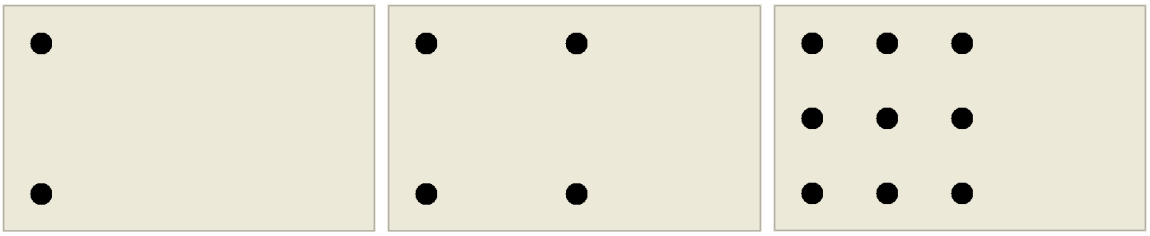
As Castalia uses C++ implementations for all protocols, the semantics of which cannot easily be compared with those of the PRISM models we intended to use, we did not further use the simulation capabilities of CaVi.

4.1.1 Network topologies

The network topologies considered in this study can be differentiated by three main aspects: first, the number of nodes; second, the spatial distribution of nodes; and, third, the distance between pairs of nodes. In this work, we consider square-shaped grids, cross-shaped grids,

and randomly-arranged distributions of two, four and nine nodes, with node distances of one, two and four metres. These parameters uniquely define networks that are either square-shaped or cross-shaped; node positions in randomly-arranged networks can of course vary. Other combinations of network size, network shape, and node distance may exist but are not considered in this study.

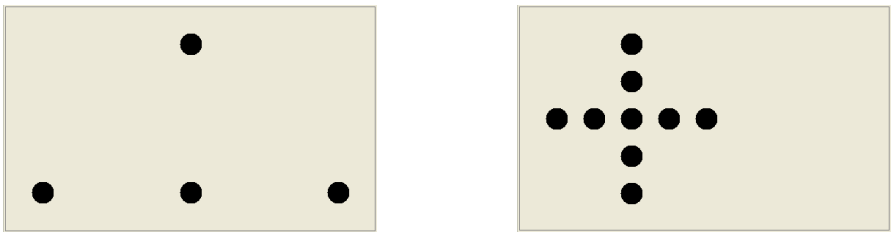
Below, all topologies are visualised using two-dimensional representations of their nodes. Topologies that differ only by node distance have identical representations. Figure 4.1 shows the square-shaped grid topologies.



(a) 2-square (b) 4-square (c) 9-square

Figure 4.1: Square-shaped grid topologies considered

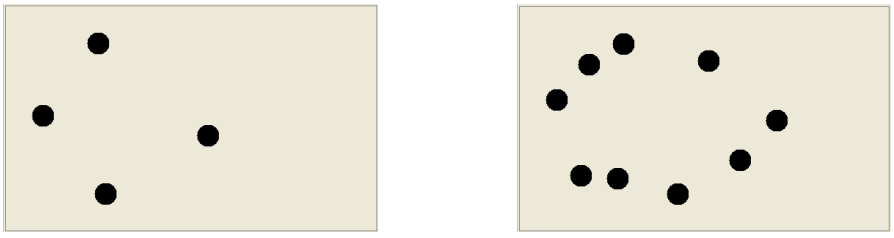
Figure 4.2 shows the cross-shaped topologies.



(a) 4-cross (b) 9-cross

Figure 4.2: Cross-shaped grid topologies considered

Figure 4.3 shows the randomly-arranged topologies.



(a) 4-random (b) 9-random

Figure 4.3: Randomly-arranged topologies considered

4.1.2 Modelling assumptions

While the models for this study have been chosen in a way such that experimental results derived from them are as generally applicable as possible, a number of modelling assumptions still had to be made.

The first assumptions relate to the channel, which usually describes the medium as well as fading and noise phenomena. We assume a lossy channel, that is, collisions can occur and messages can be lost. This lossy channel is implicitly incorporated in the computation of reception probabilities. We also use an additive interference model, as indicated by the wireless models from Chapter 3. This implies the use of *frequency-shift keying*[1], as it is the only supported modulation scheme for this interference model.

The next assumptions relate to the radio, which describes the physical transmission and reception parameters of devices such as transmission power, data rate, and noise bandwidth (noise can originate from obstacles, walls, and sources external to our network). We assume a data rate of 250 kbit/s, a noise bandwidth of 1 MHz, a receiver sensitivity of -98 dB, and a threshold probability of 0.01, which according to the literature [102] are realistic values.

4.1.3 Realistic wireless channel and radio models

In order to overcome the known weaknesses of existing models for wireless communication, we take into account the realistic wireless models introduced in Chapter 3.

In order to obtain PRISM models, three steps are necessary: in the first step, the networks are modelled using our tool CaVi. In the following step, PRISM code for the gossip protocol is generated by CaVi. In the final step, further adaptations of the PRISM models to support model parameters for channel utilisation and sending probability are made.

The main problem when generating PRISM models using CaVi was the unrealistically frequent occurence of global minimum and maximum values for signal-to-noise ratio and reception probability. This is because, first, signal-to-noise ratios are sensitive to the sending activities of all nodes in the vicinity of the receiver; second, the networks considered have a particularly high node density as the sample device is a ZigBee transceiver; and, third, it is implicitly (and incorrectly) assumed that all nodes always send, leading to high noise and thus low signal-to-noise ratios, ultimately to reception probabilities of zero.

[1] Frequency-shift keying is a modulation scheme for transmitting digital data.

One way to mitigate this problem was to manually adjust the model parameters influencing signal-to-noise ratios (and thus, indirectly, also influencing reception probabilities), using the instant visualisation of reception probabilities in CaVi. These model parameters are: number of nodes in the network, channel utilisation for each node, distance between pairs of nodes, data rate of each node, and noise bandwidth of each node. First, we selected realistic grid shapes and node distances; this way, by increasing node distances, signal-to-noise ratios could often be increased, too (but at the price of sometimes unrealistically high signal-to-noise ratios). Second, we chose sensible values for channel utilisation and sending probability.

In order to completely resolve this problem and make generated models robust to small parameters changes, we replaced the hitherto constant sending inclination $send_{i,j}$ with a new probabilistic abstraction that depends on channel utilisation $utilisation_i$ and network size n as follows:

$$send_{i,j} \stackrel{\text{def}}{=} utilisation_i/n.$$

We allow asynchronous sending of messages and also all reception probabilities considered are for the multi-hop case, that is, transmissions can be direct or indirect via arbitrarily many intermediate nodes. The PRISM models are given in Appendix A. In order to support asynchronous sending, the model needs to support nondeterminism, and thus has been modelled as an MDP.

The module for the additive-interference model is listed in Appendix section A.1. It uses the wireless channel and radio models introduced in Chapter 3. In this module, new parameters `NETWORK_SIZE` for network size and `utilisation`i, $i = 0, \ldots,$ `NETWORK_SIZE` $- 1$, for *channel utilisation* have been introduced. A new parameter `p_send` for *sending inclination* has been introduced to parameterise the already designated parameters `PsendingNode`$i, i = 0, \ldots,$ `NETWORK_SIZE` $- 1$. Channel utilisation is defined as the degree to which node i uses its proportion of the network's bandwidth. For our experiments, all nodes have the same channel utilisation. Sending inclination is defined as the probability for a node to perform a send action when it is able to do so. Messages to be forwarded are sent to the nodes at any time, that is, reception of messages takes place asynchronously (either synchronised with the action `tick` or unsynchronised). Nodes forward received messages at the same time, that is,

all messages are sent synchronously (with the action `tick`).

An ancillary module that allows reasoning about the passing of time is listed in Appendix section A.4. This module is in parallel composition to the other modules of the model such that it generates `tick` actions when time passes.

4.2 Experiments and results

In this section, we present our analysis methodology, experiments, and results. Using discrete-event simulation with realistic wireless channel and radio models as presented in Chapter 3, we investigated the impact of spatial network characteristics on performance and energy properties of a gossip protocol.

For all experiments, we consider channels of 250 kbit/s bandwidth, as this is typical for the sample device considered. In this Chapter, we assume that, with respect to the physical layer, all frames considered have a size of 200 bits.

For all experiments, we used version 3.3 of the probabilistic model checker PRISM, in particular its sparse-matrix-based verification engine, on an Intel Core 2 Duo CPU (model number P9600, two cores, 2.66 GHz clock rate) with 2 GB of memory.

4.2.1 Specification of rewards

In order to evaluate the expected number of collisions, the expected time for a correct transmission, and the expected energy consumption, PRISM supports *reachability rewards* [54, 69]. In the PRISM property specification language [59], which extends the probabilistic temporal logic PCTL with support for reward-based measures, each formula describes the expected value of a reward function that is defined over states and transitions. In PRISM, both types of reward functions are described using so-called *reward structures*.

The reward structures for this chapter's models are listed in Appendix section A.5.

Simple reward structures and formulae

Expected message throughput The main performance property in the context of routing or gossiping is *message throughput*, which describes the number of messages forwarded.

This property has been implemented in the reward structure `throughput`. The reward definition refers to node 3 as the destination node of all messages forwarded.

Expected number of messages sent The performance property *number of messages sent* describes the number of messages in total sent by nodes of the network. It has been implemented in the reward structure `sent`. The reward definition refers to node 3 as the destination node of all messages forwarded.

Expected energy consumed The energy experiments are based on the IEEE 802.15.4/ZigBee device specified in Section 2.4. The *energy consumption* describes the amount of energy in nWh consumed during the transmission of one frame. As each frame has a length of 200 bit, the rewards represent the energy consumed during a time frame of 0.8 ms. For the device considered, the reward values have been implemented in the reward structure `energy` and are given in μWs.

Expected time passed without completion The *time passed without completion* describes the number of transitions until station i has completed its attempts to forward the message, either by successful forwarding it or by choosing not to forward it. It has been implemented in the reward structure `time`. As for the throughput reward, this reward definition refers to node 3.

Derived reward structures and formulae

In order to describe multidimensional Quality of Service properties, basic performance properties can be combined: for example, throughput with loss, throughput with delay, or throughput with loss and delay. As constrained optimisation problems cannot be solved within PRISM, such properties are implemented using composite reward functions.

Expected effective throughput The *effective throughput* describes the number of new messages forwarded, that is, messages lost are not counted.

Let t be the expected throughput and l be the expected number of lost messages. As we assume that lost messages are resent, the expected number of messages to be sent in each time slot is $t + l$. Thus, forwarding one message requires $\frac{t+l}{t}$ attempts, or – in other words

– the expected fraction of new messages is $\frac{t}{t+l}$. Hence, the *expected effective throughput* e is defined as the ratio of expected throughput t to the expected number of transmission attempts per message, given by

$$e \stackrel{\text{def}}{=} \frac{t^2}{t+l}.$$

4.2.2 Specification of performance properties

Quantitative performance measures of MDP models are described using formulae of the PRISM property specification language [59], which extends the probabilistic temporal logic PCTL with support for reward-based measures. The properties evaluated in this Chapter are expressed either as a single formula (for instance, throughput) or as an arithmetic expression over two formulae for effective throughput. Each formula describes a probabilistic reachability property or an expected reachability property for a reward function that is defined over states and transitions. As our models are nondeterministic, probabilistic properties typically refer to minimum and maximum probabilities over all possible policies, rather than to one single probability as in the deterministic case. The remainder of this subsection describes all performance properties used in this work.

In order to evaluate the expected throughput and expected effective throughput until a maximum time of `TIME_MAX` has been reached, we define the expected reachability properties T1 and T2 as follows:

T1 Maximum expected throughput until a time limit reached:

$$\mathcal{R}_{\max(\text{throughput})=?}[\text{F}\, time = TIME\_MAX].$$

T2 Maximum expected effective throughput until time limit reached:

$$\frac{t^2}{s}$$

where t and s represents the throughput (as described by property T1) and the number of messages sent, respectively.

In order to evaluate the expected propagation probability, we first define the subformulae `complete` and `all_complete`:

```
formula all_complete = active0=0 & active1=0 & active2=0 & active3=0;
formula complete = active3=0;
```

Now, we define the probabilistic reachability properties PP1 and PP2 as follows:

PP1 Minimum propagation probability for the last station completing its transmission:

$$\mathcal{P}_{\min=?}[\mathrm{F}\, \mathit{complete}].$$

PP2 Minimum propagation probability for all stations completing their transmissions:

$$\mathcal{P}_{\min=?}[\mathrm{F}\, \mathit{all\_complete}].$$

In order to evaluate the propagation time, we define the expected reachability properties PT1 and PT2 as follows:

PT1 Minimum expected propagation time for the last station completing its transmission:

$$\mathcal{R}_{\min(\mathrm{time})=?}[\mathrm{F}\, \mathit{complete}].$$

PT2 Minimum expected propagation time for all stations completing their transmissions:

$$\mathcal{R}_{\min(\mathrm{time})=?}[\mathrm{F}\, \mathit{all\_complete}].$$

In order to evaluate the energy consumed, we defined the expected reachability properties E1, E2, and E3 as follows:

E1 Minimum expected energy consumption for the last station completing its transmission:

$$\mathcal{R}_{\min(\mathrm{energy})=?}[\mathrm{F}\, \mathit{complete}].$$

E2 Minimum expected energy consumption for all stations completing their transmissions:

$$\mathcal{R}_{\min(\mathrm{energy})=?}[\mathrm{F}\, \mathit{all\_complete}].$$

E3 Minimum expected energy consumption until time limit reached:

$$\mathcal{R}_{\min(\text{energy})=?}[\text{F}\, time = TIME\_MAX].$$

4.2.3 Scalability of the analyses

To reason about the scalability of the analyses conducted, space and time requirements have been evaluated with respect to frame and buffer size.

In Table 4.1, the minimum expected throughput (property T1) until a time limit `TIME_MAX` of 20 frame lengths is reached is computed for different network topologies, after which the property is considered to have failed. The results show that space and time requirements are

Nodes	States	Space	Verification time
2	59	2.8 kB	0.0 s
4	515	9.0 kB	0.0 s
9	124661	190 kB	0.24 s

Table 4.1: Impact of network size on model size and verification time (scenario: `utilisation`=0.5)

very low for all network sizes considered. Therefore, the analysis of models up to nine nodes is feasible.

In Table 4.2, the minimum expected energy consumption (property E1) for the last station to complete is computed for different network topologies, considering a time limit of 100,000 time steps. The results show that space and time requirements are very low for all network

Nodes	States	Space	Verification time
2	59	2.9 kB	0.0 s
4	515	9.1 kB	0.0 s
9	124661	190 kB	0.41 s

Table 4.2: Impact of network size on model size and verification time (scenario: `utilisation`=0.5)

sizes considered. The analysis of larger models would generally be possible, but for this protocol PRISM can currently not deal with CaVi-generated models of more than 12 nodes, as described in Section 3.4.

4.2.4 Analysing performance characteristics

Networks of square-shaped grids, cross-shaped grids, and randomly-arranged distributions of two, four and nine nodes, with node distances of one, two and four metres have been anal-

ysed with respect to four classes of performance properties regarding expected throughput, propagation probability, expected propagation time, and expected energy consumption.

The first group of experiments investigates the influence of spatial characteristics on expected throughput. For different combinations of network size, topology, and distance between nodes, where $n\,top\,d$ stands for a network of topology *top*, size n, and node distance d in metres, Figure 4.4 and Figure 4.5 show the expected throughput and the expected effective throughput, respectively, with respect to different values for channel utilisation.

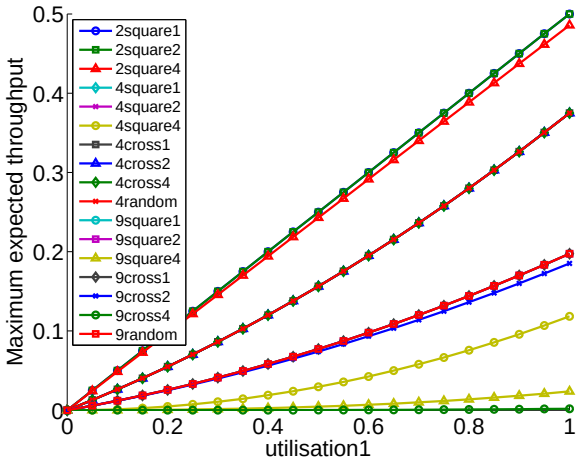


Figure 4.4: Maximum throughput over time

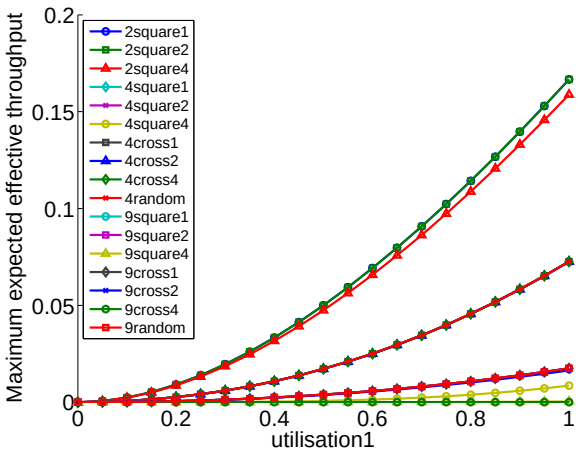


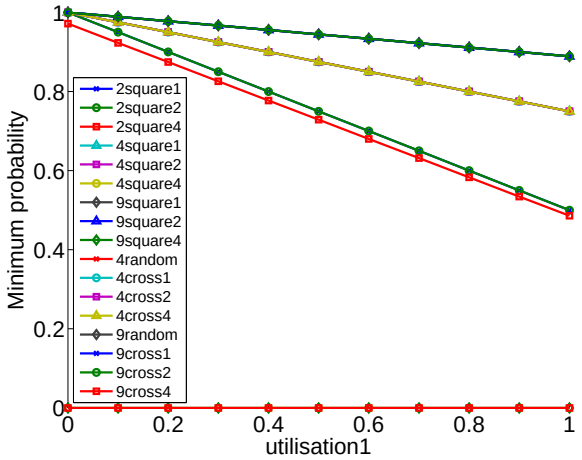
Figure 4.5: Maximum expected throughput over time

Each line of the plot represents a topology. While throughput increases exponentially with increasing channel utilisation, smaller networks generally achieve higher values than larger ones.

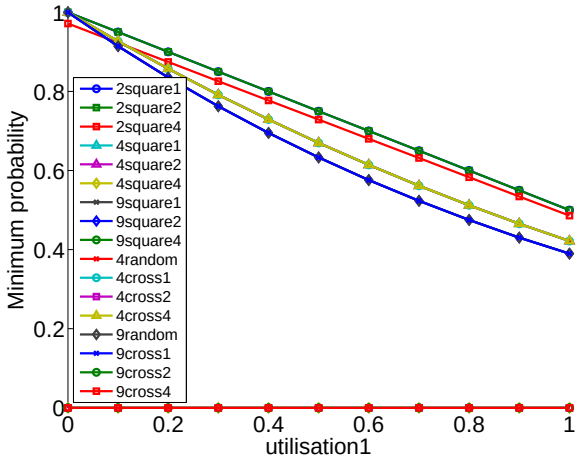
The effective throughput increases exponentially with increasing channel utilisation, indicating that for an efficient application of the flooding protocol used, a reasonably high channel

utilisation for each node should be chosen. Generally, the values for effective throughput correspond to the ones for throughput, and the topologies performing best are the same for both measures.

The second group of experiments investigates the influence of spatial characteristics on completion probability. Figure 4.6 shows the minimum probability to eventually complete and all-complete, respectively, with respect to channel utilisation. The larger the network,



(a) complete



(b) all-complete

Figure 4.6: Minimum probability to eventually complete and all-complete, respectively

the higher the probability to complete, as long as node distances do not get too high. The difference between best and worst topology is much higher when considering the probability for the last station to complete ("complete") than for all stations ("all-complete"). In networks where the completion probability equals zero, this is caused by a too large node distance and too weak received signals resulting from this.

The third group of experiments investigates the influence of spatial characteristics on propagation time. Figure 4.7 shows the expected propagation time until complete and all-complete, respectively. As expected for a gossip protocol, larger networks have lower propa-

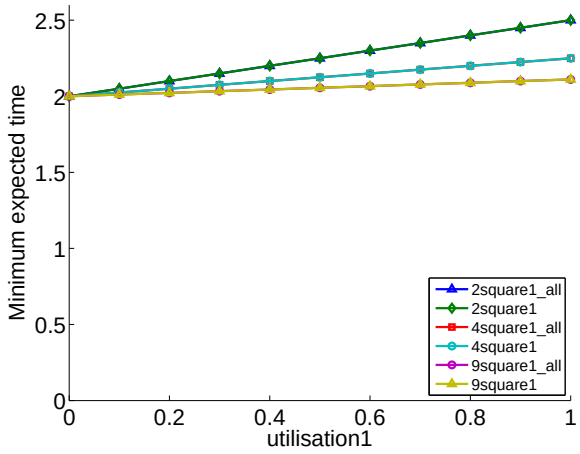


Figure 4.7: Expected propagation time until complete and all-complete, respectively

gation times than smaller ones.

The fourth group of experiments investigates the influence of spatial characteristics on energy consumption. Figure 4.8 shows the expected energy consumption until complete and all-complete, respectively, with respect to channel utilisation. Figure 4.9 shows the minimum energy consumption with respect to a time limit of 100,000 time steps. It can be seen that

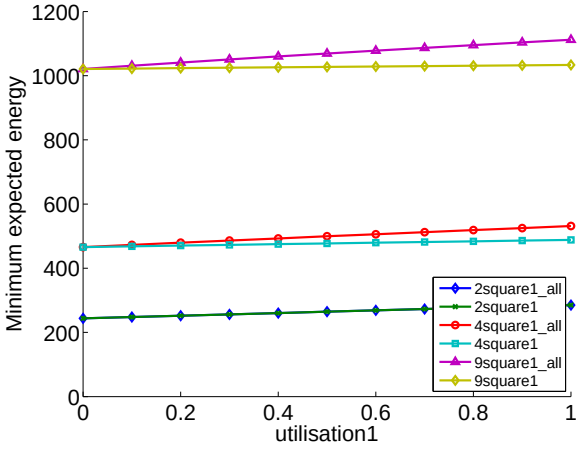


Figure 4.8: Expected energy consumption until complete and all-complete, respectively

larger networks have a higher energy consumption, but the energy consumption per node is very similar for all topologies considered. Also, the sensitivity to channel utilisation is low, except when it is desired that all nodes complete their transmissions, which is usually not

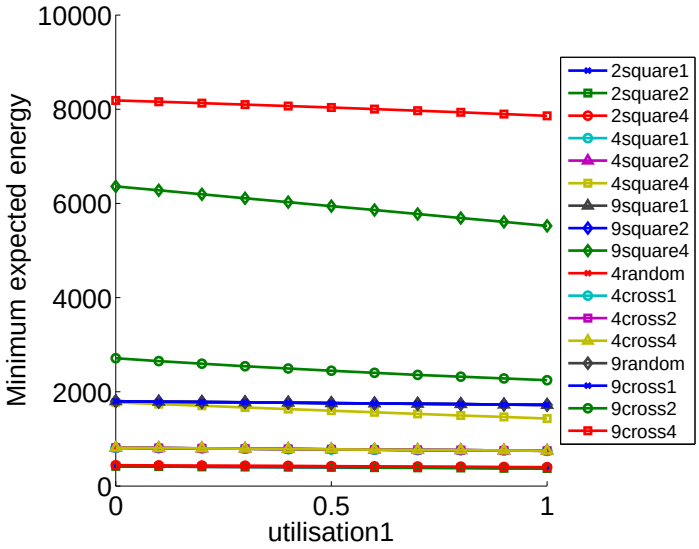


Figure 4.9: Minimum energy consumption until time limit reached

the case; for some networks, the expected energy consumption even decreases with increasing channel utilisation.

4.3 Conclusion

In this chapter, we have presented a methodology for modelling and analysing spatial properties of wireless sensor networks. The semi-formal hybrid approach combining discrete-event simulation and probabilistic model checking has proven to be efficient and effective for the comparative evaluation of wireless networks with respect to performance and energy protocols. The developed spatial models are robust and reliable and the probabilistic abstractions defined can be reused in the subsequent chapters.

In this study we used the Castalia simulator for the evaluation of routing scenarios, which were visualised in CaVi. However, the main arguments against stand-alone simulation are still valid: Castalia uses C programs (not models), the simulation of which does not follow a transparent semantics, and therefore the results are not representative for the protocol modelled.

We felt that the graphical specification approach greatly improved the process of finding suitable networks. However, there are some limitations such as the restriction of spatial network modelling to two dimensions (but not height level), and the relatively large PRISM models generated, which can become too large for the PRISM parser.

Chapter 5

Formal analysis of the IEEE 802.15.4 contention resolution protocol

In this chapter, we present the first formal analysis of the recent networking standards IEEE 802.15.4 and ZigBee. Using probabilistic model checking and the analytical framework presented in Chapter 3, we analyse performance and energy properties of the IEEE 802.15.4 contention resolution protocol.

After studying the impact of novel wireless channel and radio models on spatial properties of a simple, generic gossip protocol in Chapter 4, this chapter considers the modelling and analysis of a variety of quality-of-service properties for a complex, standardised real-world wireless sensor network protocol. While the previous chapter describes semi-formal, graphical modelling using CaVi, this chapter focusses on the detailed formal modelling required to represent wireless sensor network protocols in PRISM, demonstrates how those models can be combined with novel interference models obtained using CaVi, and evaluates both types of models with respect to result accuracy and execution performance.

Some of the work presented in this chapter first appeared in [39] and has now been extended in several ways. First, in order to reason about energy properties, we have modelled the "battery life extension", a variant of CSMA-CA for the slotted mode designed for energy reduction; also, we have modelled and analysed energy consumption for a sample device.

Second, we have demonstrated how existing PRISM models can be extended with the novel wireless channel models introduced in Chapter 3, and we have compared the collision model (used so far) with the collision-free model and the additive-interference model with respect to result accuracy and execution performance.

Low-rate wireless personal area networks (LR-WPANs) are a central communication infrastructure for pervasive computing. The recently published standard ZigBee [2] defines the upper network layers, while the lower layers are described in the IEEE standard 802.15.4 [89].

Crucial for the efficiency of a wireless network protocol is its contention resolution mechanism. When more than one node attempts to transmit a frame at the same time, a *collision* occurs, and subsequently all frames get corrupted. The standard mechanism for contention resolution in computer networks is called *carrier-sense multiple access (CSMA)*. CSMA algorithms attempt to break the symmetries of failing transmissions being restarted at almost the same time by using randomised binary exponential backoff procedures. While wired devices can listen during their own transmissions and employ CSMA with collision detection (CSMA/CD), nodes in wireless networks usually cannot listen to their own transmissions, and consequently colliding transmissions can only be detected after they have been completed. Thus, wireless devices use CSMA with collision avoidance (CSMA/CA or CSMA-CA).

The contention resolution algorithm in IEEE 802.15.4 (CSMA-CA) is a variant of those used in IEEE 802.3 Ethernet (CSMA/CD) and IEEE 802.11 Wireless LAN (CSMA/CA). It contains a more complex logical structure than the other two, but involves much smaller numerical values, and is therefore more feasible for formal verification. Since wireless sensor networks are increasingly often used in safety-critical applications, formal analysis is essential. This is the first comprehensive study of the CSMA-CA contention resolution protocol of the IEEE 802.15.4 networking standard.

In this case study, we apply a range of performance measures to different scenarios in order to evaluate how the operation of low-rate wireless personal area networks is affected by different settings of protocol attributes and how both model abstractions and interference models affect accuracy and execution performance of probabilistic model checking. This work follows previous case studies of IEEE 802.3 [29, 66] and IEEE 802.11 [64]. Our modelling approach is based on an integral-time semantics for probabilistic timed automata [62].

This chapter is divided into four sections. In the next section, we give an informal description of the IEEE 802.15.4 networking standard and the CSMA-CA contention resolution protocol. Section 5.2 contains network configuration, modelling assumptions, and probabilistic timed automata representation of our models. In Section 5.3, we present experiments and verification results. Section 5.4 concludes this chapter.

5.1 Contention resolution in IEEE 802.15.4

This section briefly introduces the IEEE 802.15.4 networking standard, defines its contention resolution protocol, and lists relevant numerical attributes.

5.1.1 The networking standard IEEE 802.15.4

The international standard IEEE 802.15.4 [89] defines *low-rate wireless personal area networks (LR-WPANs)* as structures of "low data rate wireless connectivity with fixed, portable, and moving devices with no battery or very limited battery consumption requirements typically operating in the personal operating space of 10 metres". Devices conforming to this specification can operate on 27 channels in three frequency bands with bandwidths of 20, 40, and 250 kbit/s.

Numerical attributes

In order to specify timing constraints of the contention resolution protocol, size parameters of the superframe structure, and length restrictions for different frame types, the standard uses a number of numerical attributes. Table 5.1 contains all parameters and constants that are used in our models. All values refer to the physical layer, taking into account an additional six *octets*[2] needed to transmit a frame that has been received from the media access control layer.

As transmissions at different bandwidths use different modulation techniques, the transmission of one octet requires different numbers of *symbols*[3]: for the channels 0 to 10 (at 20

[1] The first value is used for the channels 0 to 10 and the second for the channels 11 to 26.

[2] An octet is a grouping of eight bits.

[3] A symbol is the smallest unit of data that can be transmitted on a particular channel. The transmission time for one symbol is one *symbol period*.

Attribute	Value
CCA duration	8 symbol periods
PHY acknowledgement frame length	11 octets
PHY beacon frame length	23–100 octets
PHY data frame length	15–133 octets
`aBaseSlotDuration`	60 symbol periods
`aMaxBE`	5
`aMaxFrameRetries`	3
`aMaxSIFSFrameSize`	18 octets
`aMinCAPLength`	440 symbol periods
`aMinLIFSPeriod`	40 symbol periods
`aMinSIFSPeriod`	12 symbol periods
`aTurnaroundTime`	12 symbol periods
`aUnitBackoffPeriod`	20 symbol periods
`macAckWaitDuration`	120 or 54 symbol periods[4]
`macBattLifeExtPeriods`	8 or 6 symbol periods[4]
`macBeaconOrder`	0–15 (default 15)
`macMaxCSMABackoffs`	0–5 (default 4)
`macMinBE`	0–3 (default 3)
`macSuperframeOrder`	0–15 (default 15)

Table 5.1: Numerical attributes in IEEE 802.15.4

and 40 kbit/s), one octet corresponds to 8 symbols, while for the channels 11 to 26 (at 250 kbit/s), it corresponds to 2 symbols.

Superframe structure

In order to synchronise devices and to assign *guaranteed time slots (GTSs)* for low-latency applications and applications requiring a specific data bandwidth, a *superframe* structure can be used, as shown in Figure 5.1. Each superframe consists of 16 equally sized slots and is bounded by network *beacons*, which are transmitted by a designated *coordinator* device at the beginning of the first slot of each superframe. The superframe is divided into an active and an inactive portion. The former consists of a *contention access period (CAP)* and a *contention free period (CFP)* of guaranteed time slots. The CAP ends at a superframe slot boundary and has a minimum length of `aMinCAPLength`, although an exception of the latter is allowed for a temporary increase of the beacon frame to perform *GTS maintenance*[4].

If a superframe structure is used, the network is called *beacon-enabled*, otherwise *non-beacon-enabled*. In beacon-enabled networks, all communication takes place via the coor-

[4]A beacon frame performs GTS maintenance by accommodating a list of up to seven descriptors of currently maintained GTSs.

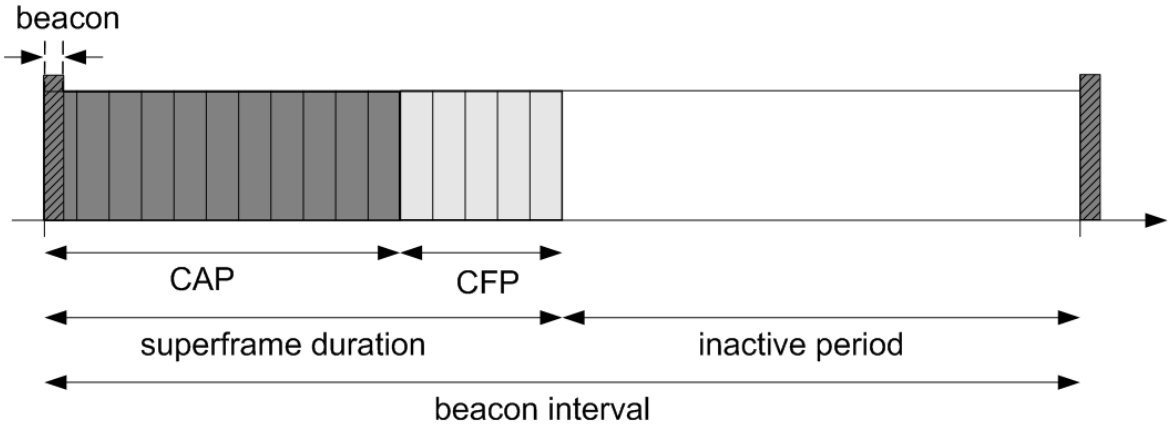


Figure 5.1: Superframe structure

dinator, while nodes in non-beacon-enabled networks can also communicate *directly* in a peer-to-peer mode.

5.1.2 The contention resolution protocol CSMA-CA

The CSMA-CA protocol is used for transmissions of data frames and MAC command frames within the CAP, unless the frame can be transmitted quickly[5] following the acknowledgement of a data request command. It is not used for the transmission of beacon frames, acknowledgement frames, or data frames within the CFP.

Depending on the type of network, the protocol operates in either *slotted* or *unslotted* mode. In beacon-enabled networks, slotted CSMA-CA is used for transmissions between the coordinator and a device. In non-beacon-enabled networks or if no beacons can be located in a beacon-enabled network, unslotted CSMA-CA is used. Peer-to-peer transmissions always use unslotted CSMA-CA. In this section, we consider both beacon-enabled networks (using slotted mode) and non-beacon-enabled networks (using unslotted mode).

The contention resolution protocol consists of the following steps:

Initialisation If a device wishes to transmit a frame using CSMA-CA, it first initialises the local variables `BE` := `macMinBE` for the *backoff exponent* and `NB` := 0 for the number of successive backoffs before the current transmission.

Backoff Before a node attempts to send a frame, it has to wait for a random integer number

[5]That is, within `aTurnaroundTime` and `aTurnaroundTime` + `aUnitBackoffPeriod` symbol, on a backoff slot boundary, and with sufficient time remaining in the CAP for the message, the IFS, and the acknowledgement.

of between 0 and $2^{\texttt{BE}} - 1$ complete *backoff periods* of length `aUnitBackoffPeriod`. This process is called *backoff*. If slotted CSMA-CA is used, transmissions are synchronised with the beacon, and therefore the backoff starts at the beginning of the next backoff period; if unslotted CSMA-CA is used, the backoff starts immediately. The first backoff period of each superframe starts with the transmission of the beacon. If the backoff has not been completed at the end of the CAP, it is suspended until the start of the next superframe, where it resumes.

Battery Life Extension When slotted CSMA-CA is used, a so-called *battery life extension* feature is available. This is a mechanism aimed at reducing coordinator receiver operation time during the CAP: when `macBattLifeExt` is set to 1, the backoff exponent is initialised with the minimum of 2 and `macMinBE`. The backoff countdown only occurs during the first six backoff periods after the end of the beacon's IFS period.

Clear channel assessment After completing its backoff, the node performs a *clear channel assessment (CCA)*. If, after eight symbol periods, the channel is assessed to be busy, both `BE` and `NB` are incremented by one, up to a maximum of `aMaxBE` for `BE` and `macMaxCSMABackoffs` + 1 for `NB`. If `NB` exceeds `macMaxCSMABackoffs`, the protocol terminates with a *channel access failure*; if not, it returns to the backoff step. If the channel is assessed free, the frame can be transmitted. In slotted CSMA-CA, two CCAs, each starting at the beginning of a backoff period, have to be performed.

Starting the Transmission In slotted CSMA-CA, a transmission can only start at a backoff period boundary and only if all steps (two CCAs, frame transmission, and acknowledgement) can be completed at least one IFS period before the end of the CAP. If the battery life extension is set to 1, the frame transmission has to start in one of the first six backoff periods after the end of the beacon's IFS period.

Acknowledgement If the originator has not requested an acknowledgement, the transmission is assumed to have been successful. If an acknowledgement has been requested, the sender needs `aTurnaroundTime` to switch from sending to receiving mode and vice versa. The recipient starts the transmission of the acknowledgement `aTurnaroundTime` after the reception of the last symbol of the data or MAC command frame if unslotted

CSMA-CA is used; it starts at a backoff period boundary between `aTurnaroundTime` and `aTurnaroundTime` + `aUnitBackoffPeriod` after the reception of the last symbol of the data or MAC command frame if slotted CSMA-CA is used. If the originator receives an acknowledgement from the recipient within a time of `macAckWaitDuration`, the data transfer has been successful. If no acknowledge is received within that time, the frame will be retransmitted up to a maximum of `aMaxFrameRetries` times, after which the protocol terminates and a *communications failure* is issued.

Interframe Space In order to give the MAC layer enough time to process data received by the PHY layer, and to support fair use of the channel [88], after each node has to wait for an amount of time called *interframe space (IFS)*[6] period after successfully completing a transmission.

5.2 Modelling

In this section, we present basic network configuration, modelling assumptions, and probabilistic timed automata models, together with extensions to support realistic interference for our analysis of the CSMA-CA contention resolution protocol.

5.2.1 Network configuration

For all scenarios, we consider a personal area network consisting of a fixed configuration of sending and receiving devices. Each sending node s_i intends to send, using CSMA-CA, a single data frame to its corresponding receiving node r_i. Both nodes start sending at the same time. As the behaviour of the destination nodes is deterministic, we incorporate into that of the sending nodes, removing the destination nodes from the model. This abstraction has been verified in the probabilistic timed automata model for contention resolution in the IEEE 802.11 protocol [64].

Other network activity, such as data transmissions from a coordinator to a node, including indirect transmissions of pending messages by the coordinator, has not been modelled. Communication activity within the CFP has been modelled indirectly as follows: dynamic allocations of GTSs in the CFP lead to varying lengths of the CAP in different superframes.

[6]This IFS is either a short IFS (SIFS) of length `aMinSIFSPeriod` if the frame length is up to `aMaxSIFSFrameSize` or a long IFS (LIFS) of length `aMinLIFSPeriod` otherwise.

In order to reflect this, the size of the CAP is determined nondeterministically for each superframe. Temporary decreases of the CAP length below `aMinCAPLength` due to GTS maintenance are not modelled.

5.2.2 Modelling assumptions

We have implemented all features of the network protocol in as much detail as possible, considering their contributions to both accuracy of the results and complexity of model construction and verification.

Ideal channel

For all models in this case study, we assume a perfect medium and ideal channel conditions, that is, no messages get lost. For beacon-enabled networks, we assume that only one personal area network (PAN) is present in the personal operating space, no PAN conflicts occur, and all nodes in the PAN are and remain synchronised; in particular, there are no synchronisation problems related to PAN ID, association and disassociation, or security.

Vulnerable period

Before starting a transmission, nodes have to perform a clear channel analysis and to switch from receiving to sending. Concurrent transmissions that start during this period, which is commonly referred to as the *vulnerable period*[7], can lead to collisions. As air propagation times of $16\mu s$ to $50\mu s$ for one symbol are negligible, we adapted Heindl and German's formula for the vulnerable period [46] to

$$\begin{aligned} VULN &\stackrel{\text{def}}{=} CCA + \texttt{aTurnaroundTime} \\ &= \texttt{aUnitBackoffPeriod} \end{aligned}$$

for unslotted CSMA-CA and

$$VULN \stackrel{\text{def}}{=} 2 \times \texttt{aUnitBackoffPeriod}$$

[7]The term "vulnerable period" refers to the fact that this is the only point in time at which the protocol is unable to protect transmissions from collisions. All transmissions are preceded by a vulnerable period, thus there is no way to avoid this vulnerability.

for slotted CSMA-CA, where CCA is the duration of a clear channel assessment.

Nondeterministic Frame Length

We assume the lengths of data frames to be nondeterministic. In unslotted CSMA-CA, different transmissions of the same message (for instance, when a lost or corrupted message is retransmitted) are allowed to have different lengths. In slotted CSMA-CA, the length of a data frame is used in order to determine whether its transmission can be finished within the CAP and is therefore the same for possible retransmissions.

Interframe Spaces

Since each node sends only one message, explicit modelling of interframe spaces is not generally necessary. In our model for slotted CSMA-CA, however, the length of interframe spaces is calculated in order to determine whether a transmission in the CAP is allowed to start.

5.2.3 Probabilistic Timed Automata models

In this study, we developed high-level generic models for slotted and unslotted operation mode of the CSMA-CA protocol. The models were developed directly in the PRISM modelling language. Each model is defined as a probabilistic timed automaton, which is given as a parallel composition of smaller modules. Various high-level features of timed automata have been used: urgent events, urgent locations, and integer variables. The model for unslotted CSMA-CA (see Figure 5.2) consists of three modules: the channel, taken from [64], and two nodes. Beacon synchronisation, which is part of slotted CSMA-CA, is realised using an extra coordinator module and modified node modules.

Generic PRISM models

Although PTAs already are a concise representation formalism, the PRISM code of the models includes additional optimisation: first, assigning arbitrary non-zero values to integer variables in a PTA would require additional states and transitions, as variables are modelled as clocks and assigning arbitrary values to them requires step-wise incrementation of these clocks by a fixed value. In PRISM, initialising integer variables to an arbitrary value can be done without introducing additional states or transitions, and this has been done for `BE`

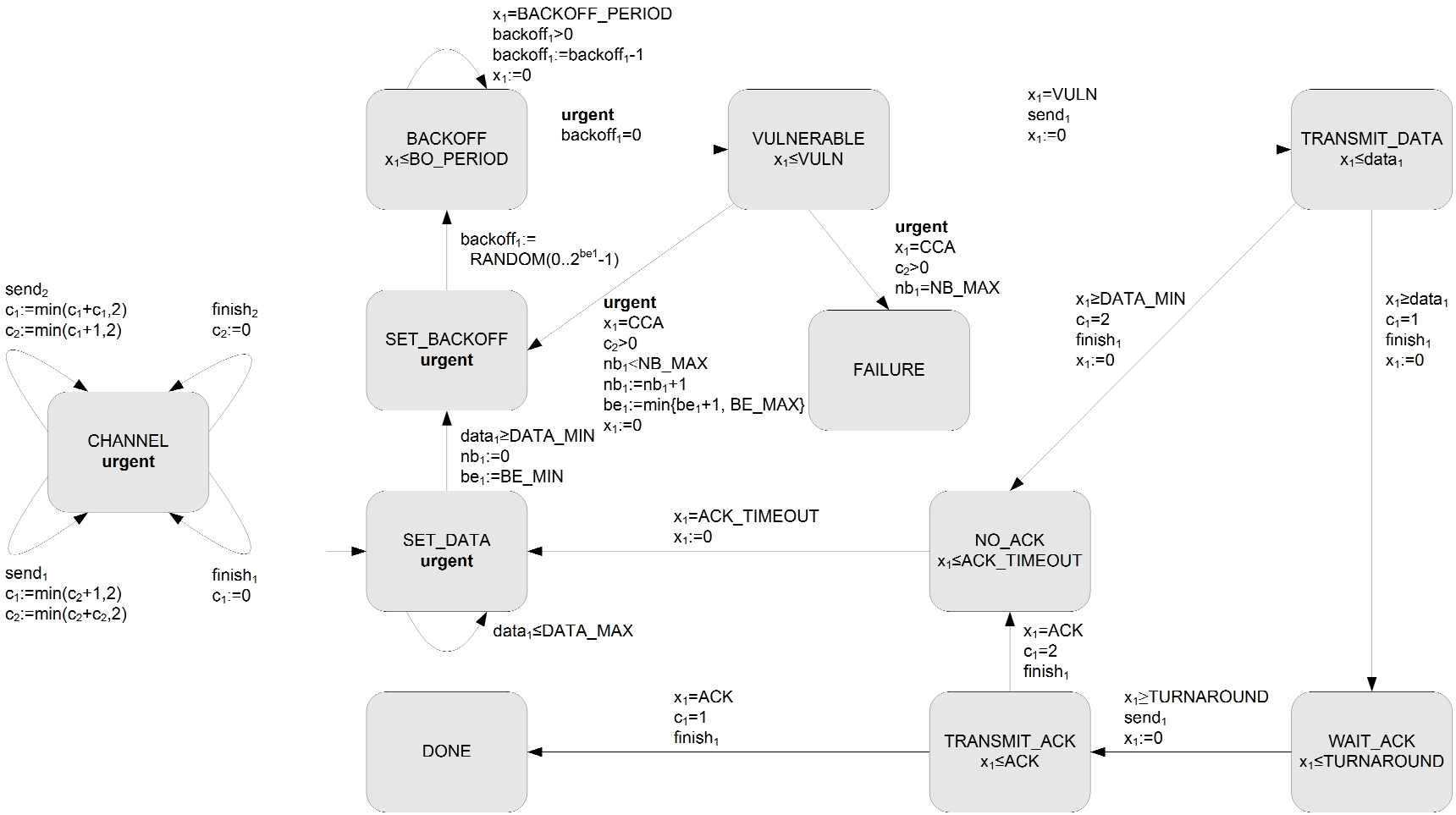


Figure 5.2: Probabilistic timed automata models for channel and node in unslotted CSMA-CA [39]

and NB. Second, the invariant conditions on transitions have been simplified after expanding high-level features such as integer variables and urgent locations.

Contrary to previous case studies, our models are generic with respect to many aspects of network configuration, transmission types, and timing parameters. This allows a much wider range of scenarios to be investigated. For both unslotted and slotted mode, the channel characteristics (frequency band and modulation technique), as well as the minimum and maximum values for the backoff procedure macMinBE, aMaxBE, macMaxCSMABackoffs can be modified. In addition to that, beacon synchronisation in the slotted mode can be controlled by modifying the parameters macBeaconOrder and macSuperframeOrder. Finally, acknowledgements, failures due to too many collisions, and acknowledgement failures can all be separately enabled or disabled.

Beacon synchronisation

For the first time, we have applied probabilistic model checking to a model of a contention resolution protocol that includes beacon synchronisation. The beacon synchronisation process synchronises the timing of all devices in the PAN and defines the lengths of CAP, CFP, and inactive period in the following superframe. Beacon synchronisation is essential for large PANs, and although our model is relatively small, it is generic enough to be adapted to larger scenarios focussing on this particular feature.

Timescale abstraction

When we applied timescale abstraction to the probabilistic timed automata models, we encountered a number of crucial modelling issues.

The abstraction granularity (the new unit of time) should be a common divisor of all constants appearing in clock constraints. Otherwise, some constants would have to be rounded up or down. If this affects the same constant in invariant conditions of a sequence of transitions, the imprecisions of these roundings can sum up to more than 1, which may add spurious behaviour to the abstracted model. These implicit delays over sequences of transitions have to be modelled by further approximating the respective constants downwards and upwards.

For example, when we scale down the PTA for slotted CSMA-CA and 20 kbit/s frequency

band, using a granularity of `aUnitBackoffPeriod`, all constants are first divided by 20. Consequently, the turnaround time after WAIT_ACK is scaled down from 12–32 symbol periods to 0–2, the acknowledgement time `ACK` from 88 to 4–5, and the acknowledgement timeout time `ACK_TIMEOUT` (that is, the maximum time for turnaround plus acknowledgement) from 120 to 6. We can see that the acknowledgement timeout delay of 6 conflicts with the possible time of $2 + 5 = 7$ for turnaround plus acknowledgement. This problem can be resolved by setting `ACK_TIMEOUT` to 7, since, according to [5], upper bounds are approximated upwards. This only causes an imprecision of 1 time unit per execution of this transition, which is acceptable since this is the only imprecision caused by timescale abstraction of this model, and it only affects transmissions in slotted CSMA-CA where an acknowledgement is being requested and transmitted successfully. For all other constants, however, the approximations are unambiguous, for example, `ACK`, which is scaled from 88 to 4–5, has to be rounded up to 5 as it is used as an upper bound. In order to model the concurrent delay constraints `ACK` and `ACK_TIMEOUT` exactly, two clocks would be necessary.

When the granularity for timescale abstraction is not a common divisor of CCA and $VULN$, a similar problem can occur. Then, the period from the beginning to the end of a CCA (which has a duration of 8 symbol periods), where transmissions can still take place without necessarily causing a collision, cannot be distinguished from the following period where the CCA has been completed but no transmissions have yet been started, where a collision would take place.

The highest granularity for an exact timescale abstraction of our models is 4 symbol periods.

Correctness The integral-time semantics preserves probabilistic and expected reachability properties of closed, diagonal-free PTA. It is easy to see that all clock constraints in our automata are closed. Despite that, the locations Backoff and Wait_Transmit in the model for slotted CSMA-CA and the locations Def_CAP_Length and CAP in the model for the coordinator contain diagonal comparisons of clocks. We can see that this does not affect the correctness of our approach:

1. In the PTA for slotted CSMA-CA, the clocks `x0` and `cap` are external to the module and can be regarded as constants. When the clock `data`i is compared to `x`i, $i = 1, 2$, it

is used as a constant.

2. In the PTA for the coordinator, the clock `cap` is compared twice to the clock $\mathtt{x}_0$. In the transition from location DEF_CAP_LENGTH (encoded as $\mathtt{s0} = 1$), the diagonal comparison is used as part of the nondeterministic choice of `x0`. In the transition from location CAP (encoded as $\mathtt{s0} = 2$), `cap` is used as a constant.

3. In the locations where `cap` and `data`$i, i = 1, 2$, are assigned values, no time can pass and all outgoing transitions are deterministic.

Hence, probabilistic and expected reachability properties of our models are still evaluated correctly in the integral-time semantics.

5.2.4 Realistic interference models

A known weakness of many formal models for wireless communication is their use of oversimplified interference models. In the collision model, for example, simultaneous transmissions from two stations in a network always result in a collision. We conjecture that models based on additive interference are more realistic but also more expensive to analyse than the simple interference models commonly used in formal analysis. It remains unclear, however, to what extent the quantitative behaviour of different interference models varies. To further investigate this aspect, we consider the three interference models introduced in Chapter 3: two common, simple variants of interference models – the collision-free model and the collision model – as well as the additive-interference model based on the wireless communication models from [102].

In order to analyse properties of the interference models, we first created specific PRISM models for each interference model. For the additive-interference model, we subsequently used CaVi to compute probabilistic abstractions for the reception probabilities. Finally, further adjustments of the PRISM models were carried out. In the following paragraphs, we describe these preparations in detail.

In order to realise different interference models, we modified each of the PRISM models for unslotted mode, slotted mode, and slotted mode with battery-life extension in order to support collision-free, collision, and additive-interference channels. In addition to the abstractions discussed in the previous chapters, we use a new variable `p_nocoll` to abstract

the collision probabilities in the `channel` module. For the additive-interference model, this abstraction is computed in CaVi, based on channel and radio parameters of the CaVi model; for the collision-free model and the collision model, fixed values of 1 and 0, respectively, are assumed. The actions `send1` and `send2` have been modified accordingly, such that sending has been assigned a probability of `p_nocoll` representing a successful transmission and a probability of `1-p_nocoll` representing a message loss.

In order to assess the different operation modes of the protocol for the additive-interference model, we needed to define a sample network that describes all relevant aspects of wireless communication, in particular number and strength of noise sources. Using CaVi's graphical specification mechanism and its computation and instant visualisation of simple properties (such as reception probability), we evaluated different network topologies and values for radio and channel parameters. As a result of this, we defined the sample network as a square grid of nine nodes with a distance of 1.5 metres between them and the other network parameters as in Chapter 4. For the different operation modes of the protocol, we used CaVi to compute probabilistic abstractions for reception probabilities in the additive-interference model and parameterised the existing PRISM model accordingly.

We further modified the PRISM models in order to resolve a number of modelling issues. One main issue was to make `p_send1` and `p_send2` parameterisable. As `station1` and `station2` are not mutually opposing noise sources, they have been considered as part of a larger network, from which they especially incorporate noise. To achieve realistic values for signal-to-noise-ratio, the values for the number and strength `linRxSignal_`i`_`j of noise sources had to be adapted with respect to device specification, topology, and device policy. To ensure that grid size and distance correlate to the number and strengths of nodes in a meaningful way, the sending probability has been set to `p_send`$i \stackrel{\text{def}}{=} \frac{\texttt{utilisation}i}{\texttt{NETWORK\_SIZE}}$. For example, a utilisation of 0.1 for a network of four nodes leads to a sending probability of 0.025 for each node.

The resulting PRISM models are given in Appendix B. This includes the implementations of the different interference models (Appendix sections B.1.4, B.1.3, and B.1.5), where the additive-interference model uses the wireless channel and radio models introduced in Chapter 3.

5.3 Experiments and results

In this section, we present our verification methodology, experiments, and results. Using probabilistic model checking, we investigated five aspects of the protocol: the performance and accuracy impact of model abstractions, the performance and accuracy impact of interference models, the performance impact of beacon synchronisation, the performance impact of the backoff procedure, and the energy impact of beacon synchronisation and backoff procedure. We consider channels of 20 kbit/s bandwidth for the performance analyses and 250 kbit/s bandwidth for the energy analyses. The bandwidth uniquely determines all timing parameters of the model. For our experiments, we used version 3.3 of the probabilistic model checker PRISM, in particular its symbolic verification engine, "MTBDD", on an Intel Pentium M CPU (model number 760, single core, 2.0 GHz clock rate) with 2 GB of memory.

5.3.1 Specification of rewards

In order to evaluate the expected time for a correct transmission, the expected number of collisions, and the expected energy consumption, PRISM supports *reachability rewards* [69, 54]. In the PRISM property specification language [59], which extends the probabilistic temporal logic PCTL with support for reward-based measures, each formula describes the expected value of a reward function that is defined over states and transitions. In PRISM, both types of reward functions are described using so-called *reward structures*.

The reward structures for this chapter's models are listed in Appendix section B.4.

Expected number of collisions The *number of collisions* describes the number of send attempts by a station while another station is already sending. It has been implemented in the reward structure `collisions`.

Expected time passed The *time passed* describes the number of transitions with the action *time*, that is, the number of transitions. It has been implemented in the reward structure `time`.

Expected energy consumption The energy-related experiments are based on the IEEE 802.15.4/ZigBee device specified in Section 2.4. The *energy consumption* describes the

State	energy consumption
BACKOFF	$320\mu s \times$ AM $\times$ number of timeslots
VULNERABLE	$128\mu s \times$ RX_wait ($+192\mu s \times$ RX$\rightarrow$ TX if CCA successful) once
TRANSMIT_DATA	$320\mu s \times$ TX $\times$ number of timeslots
WAIT_ACK	$320\mu s \times$ AM ($+192\mu s \times$ RX$\rightarrow$ TX) once
TRANSMIT_ACK	$352\mu s$ once at transition "finish" with $c1 = 1$
NO_ACK	$864\mu s \times$ RX_wait once

Table 5.2: Calculations for energy reward function

amount of energy consumed during one time slot. In the PRISM model, this reward function is based on the following calculation: at the device's bandwidth of 250 kbit/s, 1 octet corresponds to 2 symbols. Therefore, the transmission of 1 symbol takes 16 microseconds. With a time scale abstraction of 20, each time step represents 320 microseconds.

In state BACKOFF, nothing is being done and finally a switching operation is required; in state VULNERABLE, switching from RX to TX is required; in the states TRANSMIT_DATA and TRANSMIT_ACK, a TX operation is required; in state WAIT_ACK, an RX operation is required; and in state NO_ACK, a switching operation is required and afterwards nothing is being done. During transitions from RX to TX, the power is the same as within TX.

The resulting parameterisation of the transition reward function is given in Table 5.2, with the rewards in μWs. It has been implemented in the reward structure `energy`.

5.3.2 Specification of performance properties

In order to compare different models, we used a set of probabilistic reachability and expected reachability properties that were expressed in the PRISM property specification language. Although the models are PTAs, they are in fact treated as MDPs, since we use an integral semantics. As they are nondeterministic, probabilistic properties typically refer to minimum and maximum probabilities over all possible policies, rather than to one single probability as in the deterministic case.

In order to evaluate the probabilities for a transmission to be finished correctly and for a transmission to contain at least k collisions, we define the probabilistic reachability properties PR1 and PR2 as follows:

PR1 Minimum probability of both nodes successfully completing their transmissions.

$$\mathcal{P}_{\min=?}[\text{true}\,\mathcal{U}\, done_1 \wedge done_2].$$

PR2 Maximum probability of at least k collisions.

$$\mathcal{P}_{\max=?}[\text{true}\,\mathcal{U}\, col \geq k].$$

Based on the reward functions specified, we defined the expected reachability properties ER1, ER2, and ER3 as follows:

ER1 Maximum expected number of collisions until both nodes have successfully completed their transmissions.

$$\mathcal{R}_{\max(\text{collisions})=?}[\text{true}\,\mathcal{U}\, done_1 \wedge done_2].$$

ER2 Maximum expected time until both nodes have successfully completed their transmissions.

$$\mathcal{R}_{\max(\text{time})=?}[\text{true}\,\mathcal{U}\, done_1 \wedge done_2].$$

ER3 Minimum expected energy consumed until both nodes have successfully completed their transmissions.

$$\mathcal{R}_{\min(\text{energy})=?}[\text{true}\,\mathcal{U}\, done_1 \wedge done_2].$$

5.3.3 Analysing the impact of model abstractions

Owing to the state explosion problem, most of our properties could only be verified in simplified versions of the models. In order to overcome state space explosion, we used the following abstractions, which were performed manually.

Timescale abstraction Although the optimal granularity for timescale abstraction of our models is 4 symbol periods, we have used a granularity of 20 symbol periods except where stated otherwise. This substantially reduces the size of the model without sacrificing much precision and decreases verification times and memory requirements.

Fixed beacon frame length In the slotted-mode model, the length of beacon frames is chosen nondeterministically for each beacon interval. This not only increases the state space immensely, but can also prevent transmissions when both beacon and data frame are large

but superframe and CAP are small. This situation can be exploited by *pathological policies*[8] that permanently block transmissions by choosing the respective values for beacon and data frame length.

Although a sufficiently large superframe could be ensured by assigning higher values to `macBeaconOrder` and `macSuperframeOrder`, this would only worsen the state space explosion. Instead, we fixed the length of the beacon frame to the minimum value permitted by the specification, and thereby resolved both permanent prevention of transmissions and further state space explosion. Considering our small scenarios of two network nodes, this is a reasonable assumption which only slightly reduces the generality of our results.

Fixed data frame length In unslotted CSMA-CA, the length of a data frame is chosen nondeterministically within each beacon interval, while in slotted CSMA-CA, it is chosen nondeterministically before the first transmission and then maintained during possible retransmissions. As in our scenarios each node only sends one message, this nondeterminism can be replaced by data frames of fixed lengths which constitute separate models that can then be analysed separately. This abstraction reduces the state space while preserving our properties.

Table 5.3 shows model sizes (in terms of *minterms (mt)* and nodes of the underlying binary decision diagram (BDD)[9]) and verification results for unslotted and slotted mode models with different abstractions for the data frame length, using the collision interference model. In order to obtain comparable results for all models, we assumed data transmissions without acknowledgement. For the expected reachability properties, we set the maximum number of successive backoffs `macMaxCSMABackoffs` to infinity. We observed that many verification tasks only became feasible when using fixed data frame lengths and timescale abstraction. Timescale abstraction renders results less precise and should be used with care. Further experiments have shown that using data frames of equal, fixed length instead of nondeterministic length for each retransmission is necessary in order to obtain stable results for property PR1 and PR2, while the properties ER1 and ER2 can already be evaluated sufficiently precisely for nondeterministic length. For property ER3, using data frames of

[8] We use the term "pathological policy" to describe policies for resolving nondeterminism in a given model that are valid within that model but impossible (having a probability of zero) in practice.

[9] Parker [78] defines a minterm as "a valuation of its [the BDD's] Boolean variables which results in a non-zero value".

Model	data frm length	network size	time unit	PR1 nodes	PR1 mt	PR1 result	ER1 nodes	ER1 mt	ER1 result
unslotted	fixed[10]	2	4	22k	120k	1.0	22k	120k	0.125
unslotted	nondet	2	4	180k	960m	1.0	180k	960m	0.125
unslotted	fixed[10]	2	20	6.8k	13k	1.0	6.9k	13k	0.125
unslotted	nondet	2	20	56k	19m	1.0	56k	19m	0.125
slotted	fixed[10]	2	20	29k	47k	1.0	29k	47k	0.125
slotted	nondet	2	20	1m	130m	1.0	1m	130m	0.125

Model	data frm length	network size	time unit	ER2 nodes	ER2 mt	ER2 result	ER3 nodes	ER3 mt	ER3 result
unslotted	fixed[10]	2	4	93k	210k	112.8 ms	39k	200k	1371 Ws
unslotted	nondet	2	4	280k	1.6bn	-[11]	240k	1.6bn	-[11]
unslotted	fixed[10]	2	20	10k	17k	123.1 ms	11k	17k	1425 Ws
unslotted	nondet	2	20	83k	26m	123.1 ms	79k	25m	190 Ws
slotted	fixed[10]	2	20	45k	67k	166.0 ms	51k	74k	1495 Ws
slotted	nondet	2	20	1.6m	180m	166.0 ms	1.2m	180m	-[11]

Table 5.3: Performance and accuracy of different model abstractions (scenario: 20 kbit/s channel, `macBeaconOrder=1`, `macSuperframeOrder=1`, minimum beacon frame length/maximum CAP length, no permanent transmission failures)

fixed length is the only feasible option.

5.3.4 Analysing the impact of interference models

We have analysed models for unslotted mode, slotted mode, and slotted mode with battery-life extension.

Table 5.4 shows model sizes for different properties of unslotted and slotted mode under collision-free, collision, and additive interference model. Using the same assumptions and notation as in Table 5.3, we only considered data frames of fixed length and only used timescale abstraction with a granularity of 20. As a quantitative comparison of single properties over different interference models is not very meaningful, we do not list verification results. We observed that the model sizes correspond to the complexity of the interference model, with the collision-free model yielding the smallest and the additive-interference model yielding the largest ones. Nevertheless, for all unslotted and slotted mode models, the analysis of the properties considered is feasible.

Figure 5.3 shows the expected throughput for three different interference models: no collision, collision, and additive interference. Each of these models has been applied to each operation mode of the protocol: unslotted mode, slotted mode, and slotted mode with bat-

[10]The values given for *nodes* and *minterms* are for data frames of maximal length.
[11]This property could not be verified within 2 GB of memory.

Model	size	PR1		ER1		ER2		ER3	
		nodes	mt	nodes	mt	nodes	mt	nodes	mt
unslotted, cf	2	4.0k	1.4k	4.2k	1.5k	6.6k	2.3k	6.9k	2.3k
unslotted, c	2	6.8k	13k	6.9k	13k	10k	17k	11k	17k
unslotted, ai	2/9[12]	7.3k	25k	7.4k	25k	11k	33k	11k	33k
slotted, cf	2	10k	6.3k	11.0	6.4	16k	11k	24k	14k
slotted, c	2	29k	47k	29k	47k	45k	67k	51k	74k
slotted, ai	2/9[12]	31k	92k	31k	92k	47k	131k	88k	200k

Table 5.4: Model size for different interference models (scenario: 20 kbit/s channel, `macBeaconOrder=1`, `macSuperframeOrder=1`, minimum beacon frame length/maximum data frame length/maximum CAP length, no permanent transmission failures, timescale abstraction granularity 20)

tery life extension. Note that, within the 2 GB of memory allocated, not all values for the additive-interference model of the slotted mode could be verified. The results show that the probability of successful transmission equals 1 for the collision-free model, which is a unrealistic but solely explainable by the model's inherent assumption of collision-freeness. For the collision model, it equals 0 for `macMinBE` $= 0$[13] and increases from over 0.6 for `macMinBE` $= 1$ to almost 1 for `macMinBE` $= 3$. The value of 0 for `macMinBE` $= 0$ is unrealistic, as simultaneous transmissions from two nodes do not necessarily cause a collision; this depends on additional channel and radio parameters, for example network topology and signal strength. The collision model exhibits a low sensitivity to frame length. For the additive-interference model, the probabilities for unslotted and slotted mode range from about 0.1 to almost 1, except for slotted mode with battery-life extension, where they range from about 0.65 to 1; these unexpectedly high probabilities let us conjecture that the scenario considered is very suitable for using the battery-life extension. The additive-interference model exhibits a higher sensitivity to frame length, with probabilities decreasing strictly monotonically with increasing frame size and being distributed more evenly than with the other interference models.

Figure 5.4 shows the expected energy consumption for each operation mode and each interference model. The results show a slightly higher energy consumption for slotted mode (with and without battery life extension) compared to unslotted mode. Throughout all interference models, the variation of the energy consumption and its sensitivity to frame

[12]Two nodes from a network of nine nodes are sending as part of the contention resolution protocol, the others only represent inherent noise.

[13]When the minimum backoff exponent `macMinBE` equals zero, this corresponds to an initial backoff of length zero. As in our experiment, there are two stations attempting to send simultaneously, this always results in a collision.

length increase in the order from unslotted mode over slotted mode to slotted mode with battery life extension. However, in both simple interfrence models, the ratio between energy consumption and frame size is almost constant, while in the additive interference model it is slightly increasing with higher frame lengths.

Overall the detailed results show that, for the scenarios considered, the behaviour of additive-interference models is more plausible that that of simpler interference models. We believe that additive-interference models yield more accurate results than simple interference models, because the underlying wireless channel and radio models are considered state-of-the-art and have been empirically validated. As CaVi interfaces with the simulator Castalia, it would generally be possible to assess CaVi models of wireless sensor networks by means of simulation. However, such a simulation-based analysis of network protocols is not part of this thesis.

5.3.5 Analysing beacon synchronisation

In order to study the impact of beacon synchronisation, we evaluated our set of properties on models of unslotted and slotted mode using a timescale granularity of 20 symbol periods and data frames of different lengths. For the slotted mode models, the superframe parameters `macBeaconOrder` and `macSuperframeOrder` were set to 1. As expected reachability properties are evaluated to infinity if there exist policies where the respective state is not reached (here, $DONE$ is not reached when a transmission fails), we do not consider permanent transmission failures. For this, we set the maximum number of successive backoffs `macMaxCSMABackoffs` and the maximum number of retransmissions of data frames `aMaxFrameRetries` to infinity.

Figure 5.5 shows the verification results. The probability of successful transmission (Figure 5.5(a)) is always higher in the slotted mode than in the unslotted mode. In the unslotted mode, large data frames have a lower success probability than small data frames. In the slotted mode, however, an anomaly of the success probabilities can be observed. This is because, for large frames, beacon interval and superframe duration are relatively small; shortly after a transmission has started, the remaining superframe duration is not large enough anymore to accomodate a second large data frame, and consequently that transmission will not be allowed to commence (transmissions may only start if they can be finished within the CAP

– that is also the reason for some pathological policies which we refer to in the following paragraphs).

The expected number of collisions (Figure 5.5(b)) is always larger in the slotted mode than in the unslotted mode. Also, it is generally higher for larger frames than for smaller ones. The expected transmission time (Figure 5.5(c)) is always larger in the slotted mode than in the unslotted mode. Also, the larger the data frame, the longer the expected transmission time. For these slotted mode scenarios, pathological policies can permanently prevent the transmission of data frames larger than 30 backoff periods, as the combined size of beacon interval and superframe duration is only 96 backoff periods.

These experiments confirmed our intuition that the slotted mode increases the reliability of transmissions while slightly increasing the transmission time and the number of collisions. The higher number of collisions in slotted mode has two main reasons. First, in slotted mode, beacon synchronisation limits the possible times at which sending attempts may be made, thus it is more likely that two stations attempting to send choose nearly the same time for that. Second, beacon synchronisation causes an overhead in transmission time, thus the number of transmissions per time unit increases. However, the probability of successful transmission is still higher for slotted mode, as beacon synchronisation makes permanent transmission failures less likely. The plot for the probability of successful transmission in the slotted mode shows an interesting anomaly: while for small data frames an average backoff is long enough to avoid most collisions, and for large ones a node resuming from backoff does not have enough remaining time in the CAP to start a new transmission, a scenario with data frames whose length is half of the length of the usable (non-beacon) period of the CAP accounts for the worst case.

For large sizes of the data frames and a small size of the superframe, the expected transmission time evaluates to ∞, due to the existence of pathological policies where transmissions are not completed successfully. However, for larger superframe sizes, it evaluates to a finite value.

5.3.6 Analysing the backoff procedure

In another experiment, we have studied the impact of the backoff parameter `macMinBE`, which determines the minimum value of the backoff exponent `BE`, for data frames of different

lengths with the same models as in the previous table. Figure 5.6 shows the properties PR1, ER1 and ER3 for different data frame lengths in unslotted mode and in slotted mode, with and without the battery life extension feature enabled. As expected, a high value of `macMinBE` (the default is 3) decreases both collision probability and expected transmission time. However, longer backoff times result in a slightly higher energy consumption.

5.3.7 Analysing energy characteristics

Energy consumption is a key issues for wireless sensor networks, and the analysis of energy properties allows further insight into the behaviour of networks and protocols beyond the analysis of timing and collisions. In these experiments, we have compared energy properties for different power management policies and for different values of the backoff parameter `macMinBE`.

Figure 5.4 shows verification results for the property ER3 in unslotted mode, slotted mode, and slotted mode with the battery life extension feature enabled, for collision-free, collision and additive interference models. The results show that the probability of successful transmission is lowest for unslotted mode, higher for slotted mode, and highest for slotted mode with battery life extension. The energy consumption per time unit is only marginally higher for slotted mode than for unslotted mode, but visibly higher for slotted mode with battery life extension. The energy consumption per successful transmission, however, is lowest when slotted mode with battery life extension is used, as this ensures the shortest backoff times and the lowest numbers of retransmissions.

As the additive-interference model consists of a larger network that takes into account noise, its performance results cannot directly be compared with those for the two simple interference models. Nevertheless, the same relationships with respect to successful-transmission probability and energy consumption between the protocol's operation modes and values for `macMinBE` can be identified for all interference models. However, both modelling and analysis of the additive-interference models are significantly more time-consuming than for the simple interference models. Hence, for application scenarios where exact quantitative results are not required, we recommend using the collision model, while for other application scenarios we recommend the additive-interference model.

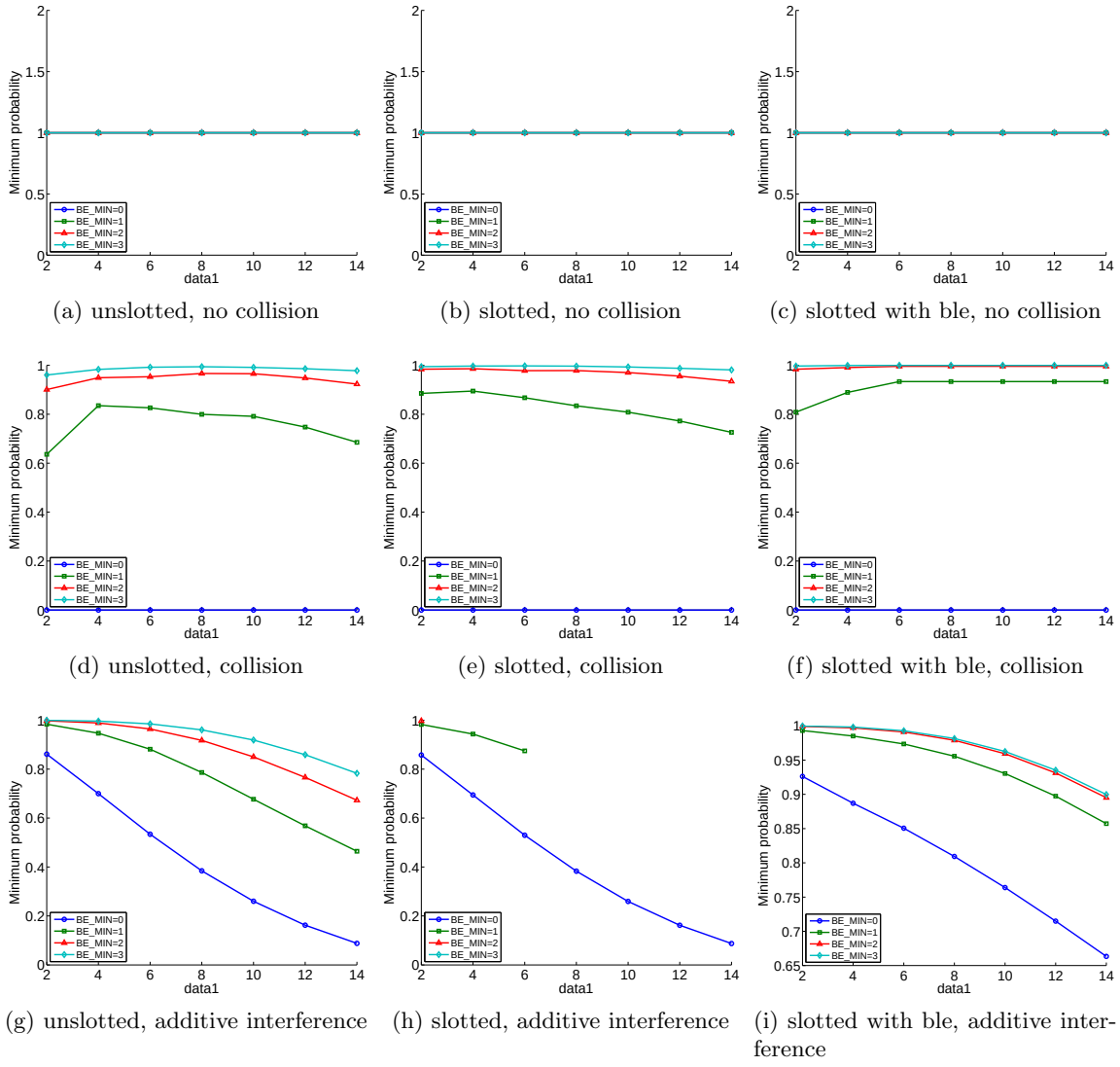
5.4 Conclusion

We have presented the first application of probabilistic model checking to the IEEE 802.15.4 networking standard. In a comprehensive case study, we developed high-level generic models for the CSMA-CA contention resolution protocol, evaluated performance properties, and compared different abstraction techniques, thereby providing a better understanding of both the protocol and modelling issues. For the first time, probabilistic model checking was used to evaluate different interference models on a real-world protocol. Contrary to test and simulation, our formal approach provides provably correct results that cover the full behaviour of the models.

In comparison to previous applications of probabilistic model checking to contention resolution procotols [29, 64, 66], our models are more realistic. Amongst other details, they cover all operation modes and parameters of the protocol and the energy consumption for all power states and transitions of a real wireless sensor device. We have shown that previous modelling techniques for timescale abstraction are inadequate here and produce pathological policies and consequently misleading results.

While probabilistic model checking of wireless sensor networks is limited by the state space explosion problem, we have presented efficient modelling techniques to deal with this. In particular, timescale abstraction with a suitable abstraction granularity and the use of suitable interference models provides an effective trade-off between result accuracy and execution performance. For instance, where the additive-interference model describes the quantitative behaviour of the protocol most realistically, the collision model delivers reliable qualitative information about the protocol with significantly lower demands on memory and execution time. Other state-space reduction methods such as symmetry reduction, partial-order reduction, symbolic representations, and induction are likely to further improve the scalability of our approach to larger scenarios.

This case study can be continued in many ways. Recent work [100] resulted in a detailed performance analysis of security key exchange in the ZigBee protocol for a range of scenarios. New experiments could focus on larger networks with more complex behaviours of the nodes, such as transmitting more than one message per node and allowing nodes to send and to receive.

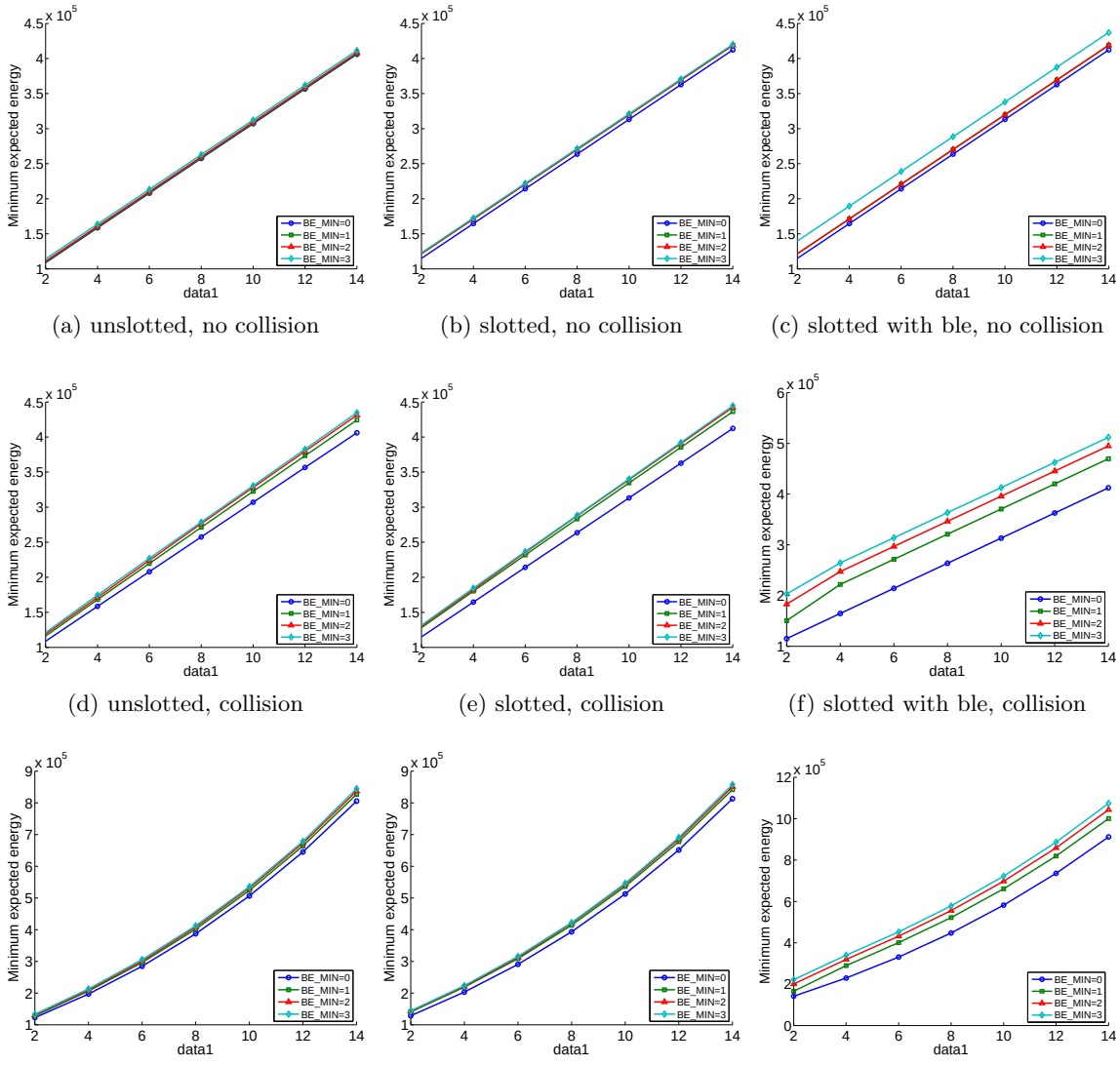


(a) unslotted, no collision (b) slotted, no collision (c) slotted with ble, no collision

(d) unslotted, collision (e) slotted, collision (f) slotted with ble, collision

(g) unslotted, additive interference (h) slotted, additive interference (i) slotted with ble, additive interference

Figure 5.3: Probability of successful transmission for different interference models (scenario: `250 kbit/s channel, noise bandwidth 1 MHz, macBeaconOrder=1, macSuperframeOrder=1, minimum beacon frame length/maximum CAP length, timescale abstraction: 20 symbol periods`)



(a) unslotted, no collision (b) slotted, no collision (c) slotted with ble, no collision

(d) unslotted, collision (e) slotted, collision (f) slotted with ble, collision

(g) unslotted, additive interference (h) slotted, additive interference (i) slotted with ble, additive interference

Figure 5.4: Energy consumption for different interference models (scenario: `250 kbit/s channel, macBeaconOrder=1, macSuperframeOrder=1, minimum beacon frame length/maximum CAP length, no permanent transmission failures, timescale abstraction: 20 symbol periods`)

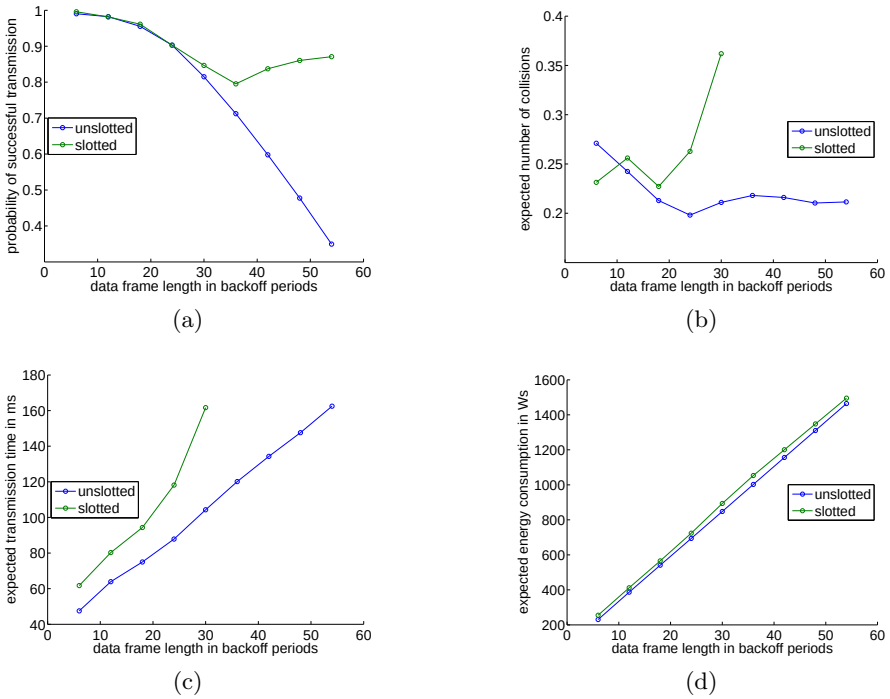
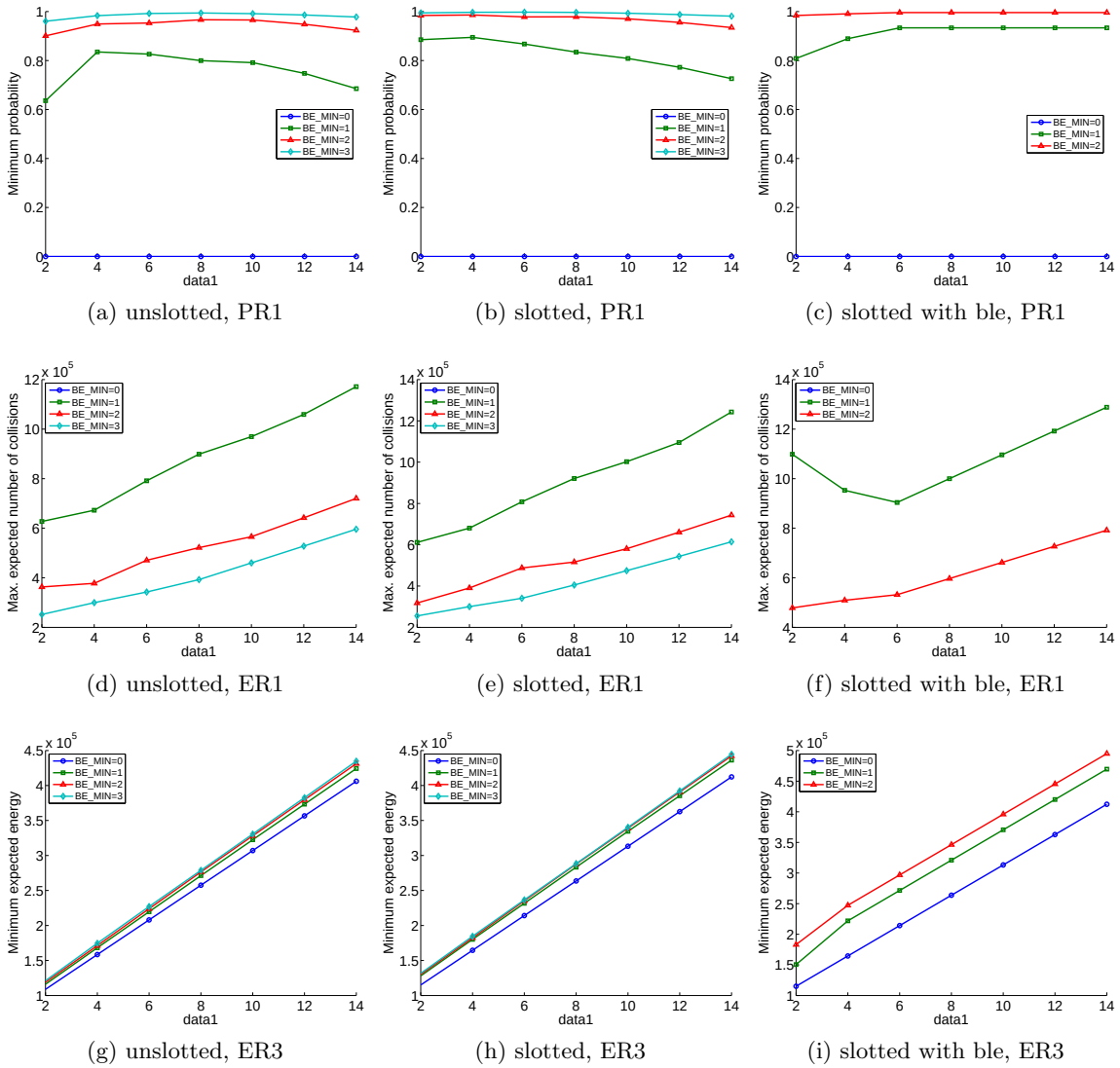


Figure 5.5: Performance for data frames of different lengths (scenario: `20 kbit/s channel, macBeaconOrder=1, macSuperframeOrder=1, minimum beacon frame length/maximum CAP length, no permanent transmission failures, timescale abstraction: 20 symbol periods`)



(a) unslotted, PR1 (b) slotted, PR1 (c) slotted with ble, PR1

(d) unslotted, ER1 (e) slotted, ER1 (f) slotted with ble, ER1

(g) unslotted, ER3 (h) slotted, ER3 (i) slotted with ble, ER3

Figure 5.6: Performance for different values of `macMinBE` (scenario: `250 kbit/s channel, macBeaconOrder=1, macSuperframeOrder=1, minimum beacon frame length/maximum CAP length, no permanent transmission failures for expected reward properties, timescale abstraction: 20 symbol periods`)

Chapter 6

Formal analysis of dynamic slot allocation protocols for low-rate wireless networks

In this chapter, we present the first formal analysis of dynamic slot allocation protocols for *low-rate wireless networks*. Based on probabilistic model checking and the analytical framework presented in Chapter 3, we describe a methodology for modelling and analysis of such protocols.

Slot allocation protocols are a crucial bandwidth management mechanism for ad-hoc networks. A key aspect of protocol design is the optimisation of critical performance criteria. Successful development of a protocol requires realistic modelling and analysis techniques as well as efficient tool implementations.

In this work, we present an alternative approach for conducting comparative studies between designs. Using probabilistic model checking, the precise operational behaviour of the system is under direct control of the protocol designer, as opposed to ambiguous semantics of simulators. Both probabilistic and nondeterministic behaviour, which are common in wireless applications, are modelled adequately, adding performance measures ranging from worst-case to best-case. As this approach is purely model-based, it is no longer necessary to examine large amounts of data.

Unfortunately, the high expressiveness of this formal approach is also its major draw-

back. Models often result in very large state spaces, which makes model checking infeasible. Although there are established techniques for improving efficiency, such as symbolic data structures [72], the analysis of detailed protocols and large networks is still not viable. As shown by Fehnker et al. [34], appropriate abstractions of a protocol are able to extend the scope of formal methods without comprimising the quality of the analyses. Abstractions refer to versions of the model where all details unrelated to the performance property considered have been removed. Therefore, a central part of this work is to find appropriate abstractions for protocol, environment, and traffic such that realistic performance profiles for different protocol designs and node policies can be evaluated. Currently only very few useful abstractions for performance properties of wireless protocols exist.

This work is motivated by the problem of optimising protocol design for dynamic slot allocation in wireless sensor networks. As mentioned in Section 2.2.1, medium access control can be realised using either random, scheduled, or hybrid access schemes. While random access protocols allow a highly flexible allocation of slots to senders, their performance is unsatisfactory under high traffic and in dense network topologies, as these situations cause a high number of collisions. Scheduled access protocols can deliver a high throughput in static scenarios, but perform badly in dynamic scenarios, where their schedules are either fixed or require frequent updating, both wasting bandwidth. In order to guarantee a certain Quality of Service, as for applications that require a specific minimum bandwidth or maximum latency, employing scheduled access is the only solution.

Hybrid access protocols aim to deliver a high performance under flexible traffic conditions, the highest overall performance, and a guaranteed Quality of Service. According to [82], the best hybrid access control protocols, ADAPT and Z-MAC, are schedule-based but apply random access whenever the owner of a slot does not indicate its intention to use it. For this purpose, each slot is divided into different parts for priority, contention, and transmission (ADAPT); or, scheduled and random access are always used conjunctively but with smaller backoff values for the owner of the slot (Z-MAC). Indeed, these protocols perform always at least as well as their random equivalents, and better than scheduled protocols if the bandwidth requirements are sufficiently dynamic. However, the initial slot allocation schedule of Rana's protocol does not take bandwidth requirements into account, resulting in inefficient use of frames (some parts not used) and slots (sending by non-owners takes longer).

Rana et al. propose a bandwidth-aware dynamic slot allocation protocol that is claimed to improve the performance of previously existing ones. However, an analysis of this protocol is beyond the scope of this work, as a presentation of its intricate algorithms would be extensive and still not likely to produce important insights into the methodological issues that are the focus of this work.

In this chapter, we present a methodology based on probabilistic model checking for modelling and analysis in several development phases of dynamic slot allocation protocols. To illustrate this, we study a simplified dynamic slot allocation protocol for low-rate wireless networks, which is sufficiently simple to allow a comprehensive analysis, yet complicated enough to exhibit interesting, non-trivial results. This is the first application of formal methods to dynamic slot allocation protocols, and thereby the first time such protocols have been analysed using probabilistic model checking.

We develop formal abstractions for all the critical parts of the protocol, which makes the model generic and scalable and enables a comparison of the performance of different slot allocation policies. Using the probabilistic model checker PRISM, we describe how one can systematically find policies and design parameter values for given scenarios that are optimal with respect to performance properties such as throughput, message loss, and delay. As energy is a critical aspect of wireless network protocols, energy-related properties such as energy consumption, battery lifetime, and network lifetime are also studied. We also show how the relationship between locally optimal policies, that is, policies that are optimal with respect to the node running it, and globally optimal policies, that is, policies that are optimal with respect to the whole network, can be studied.

This chapter is divided into four sections. In the next section, we define the protocol studied. In Section 6.2, we describe the modelling approach, and present stochastic models for environment, network configuration, and protocol. In Section 6.3, we present the analysis approach, describe experiments performed, and discuss results. Section 6.4 concludes this chapter.

6.1 A simple slot allocation protocol

The basis of this study is a simple protocol for dynamic slot allocation, which can be considered in the context of a gossiping-based routing application. It is not based on an actual protocol but inspired by the ones presented in [82].

In a wireless network, network nodes, called *forwarders*, receive messages from users in the environment and retransmit them through the network. Time is divided into frames of equal size, and each frame is divided into time slots of equal size. In order to avoid collisions, which may be caused by simultaneous submissions from different nodes, communication follows a scheduled access scheme.

Nodes usually use adaptive policies to decide, depending on the context, when to request and when to acknowledge other nodes' requests for further time slots. Adaptivity of a node's policy takes into account information about the nodes' resources, its policy, and the environment. Policies consider, for instance, the number of slots allocated to this node, this node's incoming traffic, or the network's overall traffic. Its possible actions include requesting the allocation of additional time slots, giving up allocated time slots upon other nodes' requests, and not giving up currently not needed time slots despite other nodes' requests.

The protocol consists of the following steps:

Initialisation At the outset, each node is statically allocated one slot (the first slot of each frame belongs to the first node, the second slot to the second node, and so on), as indicated in Figure 6.1. These slots cannot be reallocated. The remaining slots are initially distributed evenly amongst all nodes but remain available for dynamic reallocation.

Receiving messages from the environment Consider a node running this protocol. In each time slot, depending on the network traffic, the node may receive a single message that it has to retransmit, which it subsequently stores in its buffer or drops.

Choosing an action The owner of the current time slot has a choice between three actions: forwarding a message, requesting a time slot, and acknowledging a request.

Forwarding a message For each time slot, the node owning the current time slot can

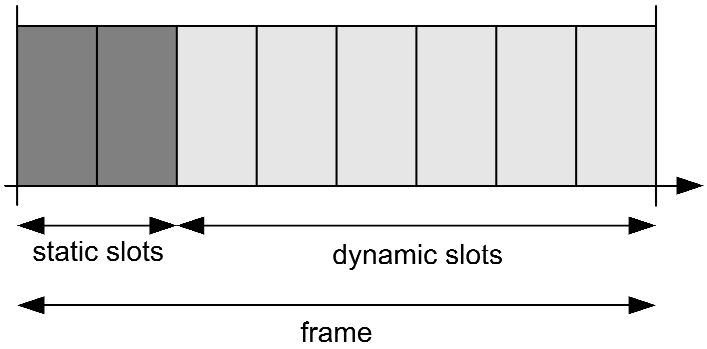


Figure 6.1: Initial slot allocation in a frame

forward a single message that has been stored in its buffer.

Requesting a time slot For each time slot, the node owning the current time slot can request the allocation of an additional time slot that is currently not owned by itself.

Acknowledging a request For each time slot, if a node owns the current time slot and another node has requested the allocation of an additional time slot, it can acknowledge that node's requests, and, with effect from the next time slot, one of its own slots is reallocated to that node.

Time passing After a time slot is over, the next time slot starts, with the same steps available again. The last time slot of a frame is immediately followed by the first time slot of the next frame.

6.2 Modelling

This section describes how dynamic slot allocation protocols can be modelled and analysed using the probabilistic model checker PRISM. Based on the protocol described in section 6.1, we define abstractions for the protocol, the network, and the environment. Each of these entities is modelled using separate PRISM modules, and the full model of the protocol is formed by their parallel composition, representing a discrete-time Markov chain. The models we develop are generic, in the sense that they can be instantiated for different parameters.

For this study, we consider a model of two nodes. The model contains generic representations of many configurable aspects of this class of protocols, and most parameters describing network and protocol can be chosen by the user. Some parameters, such as the number of

nodes, the initial slot allocation, and the position of static slots, are fixed. However, changes to these values can be realised by moderate extensions of the basic model.

The remainder of this section describes the submodels for environment, network, and protocol. The respective PRISM models are given in Appendix C.

6.2.1 The environment

The environment of the protocol comprises the external users of the network and the messages they produce. The only connection between network and environment is from the feeder to the forwarders. In particular, there is no feedback from forwarders to the feeder.

Network traffic is *self-similar* [37], that is, inter-arrival times of messages follow a time-dependent distribution. As our model is based on DTMCs, we model inter-arrival times as a geometric distribution, which is memoryless. The quantitative behaviour of the environment, that is, the traffic characteristics, is determined by the probability distribution of outgoing messages. In this work, we abstract traffic using two parameters: *channel utilisation* and *reception probability*. The channel utilisation u of a node is defined as the average ratio of the actual to the maximum possible amount of traffic sent to this node. The reception probability is defined differently depending on whether a packet has been received in the previous time slot: If that is the case, the reception probability put_1 (which we call *reception-continuation probability*), $0 \leq put_1 < 1$ represents the average length of blocks of traffic. Otherwise, that is, if no packet has been received in the previous time slot, put_0, $0 < put_0 \leq 1$, is defined as the probability that a node does not receive a message when it has received traffic in the previous slot (which we call *reception-initiation probability*). In the model, the reception probabilities of node j are represented as `P_PUT0_j` and `P_PUT1_j`.

The relationship between channel utilisation and reception probability can be described mathematically as follows. The definition of put_1 implies that blocks of consecutive time slots where a specific forwarder receives messages have an expected length of $\frac{1}{1-put_1}$. Analogously, the reception probability put_0 implies that blocks of time slots where this forwarder does not receive messages have an expected length of $\frac{1}{put_0}$. The channel utilisation for a single forwarder is the ratio between the average length of a message block and the sum of the

average lengths of a non-message block and a message block. It is given by

$$u \stackrel{\text{def}}{=} \frac{\frac{1}{1-put_1}}{\frac{1}{put_0} + \frac{1}{1-put_1}}.$$

This can be simplified to the following function of put_0 and put_1:

$$u = \frac{put_0}{put_0 + (1 - put_1)}.$$

From this, the following equation for the reception probability put_0, which is used in the PRISM model, can be derived:

$$put_0 = (1 - put_1) \cdot \frac{u}{1 - u}.$$

In this study, we assume that the stochastic characteristics of each user's traffic is independent from the number of users. Accordingly, the behaviour of all users is represented in a single module `feeder` (see Appendix section C.1.1). The probability parameters `PUTi_j`, declared in Appendix section C.1.2, describe the traffic from users to individual nodes j of the network; they give the probability for a message sent in the current slot. The model distinguishes two cases: i equals 1 if in the previous slot a message has been sent to node i, and it equals 0 otherwise. Note that in the model the formulae above contain `UTILISATION1` and `UTILISATION2` instead of u. This is because channel utilisation refers to each node's fair share of the overall bandwidth, while the probability parameters have the same meaning locally and globally.

A state-transition graph for the semantics of the environment module `feeder` is shown in Figure 6.2. In each time slot, four types of interactions between the environment on one hand and its feeding users and network nodes on the other hand take place, represented by different values of the variable `mode`. Each behaviour corresponds to a number of actions, which are represented by action labels written in square brackets. That is, in the initial state (`mode` equals 1), users can send messages to node 1 (action `put1`). Subsequently, when `mode` equals 2, users can send messages to node 2 (action `put2`). After this, when `mode` equals 3, depending on the current allocation of slots and the policies of the nodes, it is chosen (action `choose1` or `choose2`) which activity the slot owner (given by variable `ownc`)

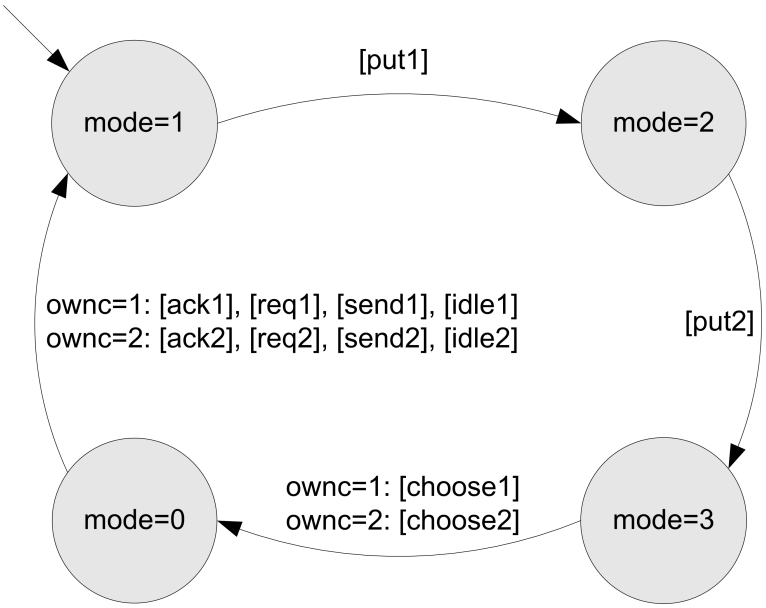


Figure 6.2: Labelled transition system of the environment

```
const int index1 = 1;
const int index2 = 2;
```

Figure 6.3: Declaration of generic forwarder indices

performs next: send, request, acknowledge or nothing. Then, when `mode` equals 0, the slot owner (given by variable `ownc`) performs the chosen action: it can forward a message (action `send1` or `send2`), request additional time slots (action `req1` or `req2`), acknowledges other nodes' requests (action `ack1` or `ack2`), or stay idle (action `idle1` or `idle2`). Finally, `mode` equals 1 again. The motivation for designing the feeder module based on these four simple modes is to allow intuitive understanding and simple synchronisation with the model's other modules.

6.2.2 The network

In this example, the network consists of two nodes. As transitions in the model need to refer to each node in a generic way, we declare generic indices for them (see Figure 6.3). By doing this, PRISM modules for forwarders that refer to individual nodes need to be defined only once and modules for further forwarders can be defined by simple syntactical replication of them.

The behaviour of each node of the network is controlled by its buffer size and its policy.

Policy parameters, declared in Appendix section C.2.1, can be divided into stochastic and discrete parameters. The buffer sizes for node 1 and node 2 are determined by the values of `BS1` and `BS2`, respectively. There are three pairs of discrete parameters that describe when requests can be made and when other nodes' requests may be acknowledged:

- the *acknowledging thresholds* `ACK1` and `ACK2` describe the maximum buffer occupation for node 1 and node 2, respectively, to acknowledge another node's request;
- the *requesting thresholds* `REQ1` and `REQ2` describe the minimum buffer occupation for node 1 and node 2, respectively, to make a request;
- the *buffer occupation difference thresholds* `BDIFF1` and `BDIFF2` describe the minimum difference in buffer occupations of node 1 over node 2 and node 2 over node 1, respectively, to make a request.

The parameters `REQ1` and `REQ2` represent the eagerness or aggressiveness of a policy with respect to requesting dynamic time slots. Low values stand for eager policies and high values for lazy ones. The parameters `ACK1` and `ACK2` represent the hesitancy or passivity of a policy with respect to granting acknowledgements. Low values stand for hesitant policies and high values for non-hesitant ones.

There are two pairs of stochastic parameters that describe the probabilistic choice between the actions `ack`, `req`, and `send` when more than one of them is available:

- the *sending inclination* parameters `P_SEND1` and `P_SEND2` describe the probability for node 1 and node 2, respectively, to forward a message (`send`) when it is able to acknowledge another node's request (`ack`) or to make a request (`req`);
- the *requesting inclination* parameters `P_REQ1` and `P_REQ2` describe the probability for node 1 and node 2, respectively, to make a request (`req`) when it is unable to forward a message (`send`) but is able to acknowledge another node's request (`ack`).

When two or three actions are enabled at the same time, the probabilistic choice between them combines sending inclination, requesting inclination, and their complements and uses their products as conditional probabilities (see Appendix section C.2.3). The sum of the probabilities of all enabled commands is always one, in order to give a well-formed DTMC.

In each slot, when mode equals 3, a choice between the actions available to the slot owner is made, which is reflected in the variables `choice1` and `choice2`; when mode equals 0, the action is executed based on sending and requesting inclination.

As this work is based on a general notion of adaptive policies, this includes most static and random policies as well. In static policies, all time slots are subject to a fixed allocation to nodes. They are represented by

$$\texttt{ACK} = -1, \texttt{REQ} = \texttt{BS} + 1.$$

The parameter definition $\texttt{ACK} = -1$ for a node says that other nodes' requests may be acknowledged precisely when this node's buffer occupation does not exceed -1, that is, other nodes' requests are never acknowledged. The parameter definition $\texttt{REQ} = \texttt{BS} + 1$ for a node says that this node may make a request precisely if its buffer occupation equals or exceeds $\texttt{BS}+1$, that this is, this node never makes requests. In random policies, all time slots but one per node are allocated dynamically, and they are always reallocated to requesting nodes as soon as possible. They are represented by

$$\texttt{ACK} = \texttt{BS}, \texttt{REQ} = 0, \texttt{P\_SEND} \in (0,1), \texttt{P\_REQ} \in (0,1).$$

The parameter definitions $\texttt{ACK} = \texttt{BS}$ and $\texttt{REQ} = 0$ for a node say that other nodes' requests may be acknowledged whenever the buffer occupation does not exceed `BS` (that is, requests may always be acknowledged) and that requests may be made whenever its buffer occupation equals or exceeds 0 (that is, requests may always be made). The parameter definitions for `P_SEND` and `P_REQ` exclude the deterministic cases of never/always sending/receiving. The various combinations of policy parameters allow the representation of a wide range of adaptive policies.

In order to support a succinct, high-level representation of policies, the availability of the different actions is represented using abstractions defined in Appendix section C.2.2, for example, `can_ack1` for "node 1 can acknowledge another node's request for dynamic time slots". The boolean formulae in module `forwarder1` describe whether a node is allowed to acknowledge another node's request (`can_ack1`), to receive an acknowledgement from another node (`can_ack2`), to make a request (`can_req1`), or to forward a message (`can_send1`). The

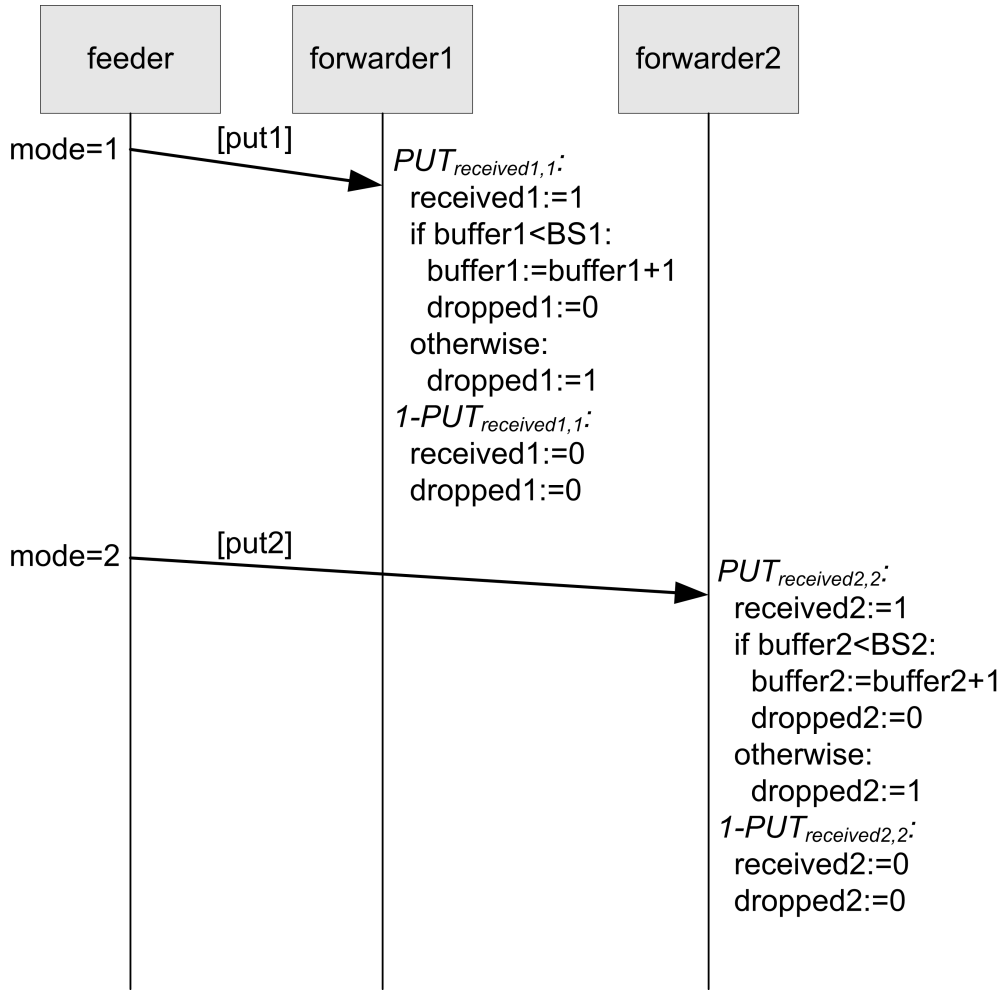


Figure 6.4: Semantics of the actions `put1` and `put2`

variables `r1` and `r2` equal 1 when node 1 or node 2 have requested ownership of a slot owned by nodes other than itself, respectively; otherwise, they equal 0.

The behaviour of network nodes conforming to the protocol is implemented in module `forwarder1` (see Appendix section C.2.3). New messages received are accepted at any time (`put1`); however, following [4], we assume that messages are lost by a node if the node's buffer is full when they are received (see Figure 6.4). The actions `ack1`, `req1`, and `send1` are available whenever the corresponding primitives of `can_ack1`, `can_req1`, and `can_send1`, respectively, evaluate to true (see Figures 6.5, 6.6, and 6.7). The variable `r1` indicates whether node 1 has requested the allocation of an additional slot. The probabilistic choice between these actions is governed by the policy parameters `P_SEND1` and `P_REQ1`. To avoid livelocks and deadlocks when none of the previously introduced communication primitives is enabled, the action `idle1` (see Figure 6.8) can always be chosen to let time pass when this is the case.

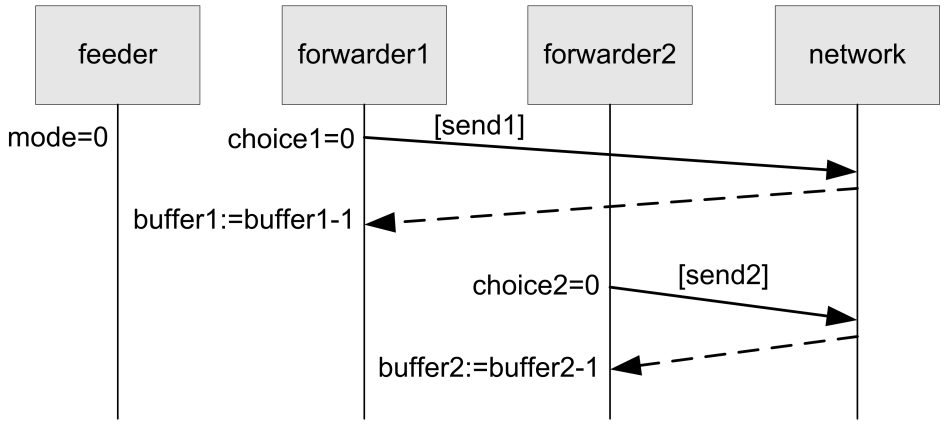


Figure 6.5: Semantics of the actions `send1` and `send2`

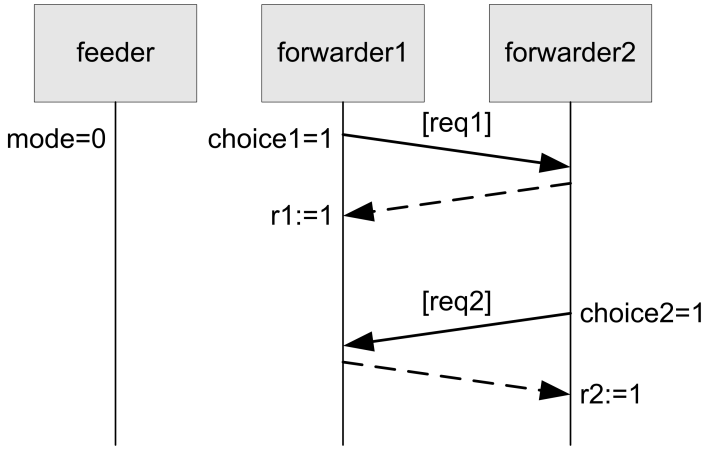


Figure 6.6: Semantics of the actions `req1` and `req2`

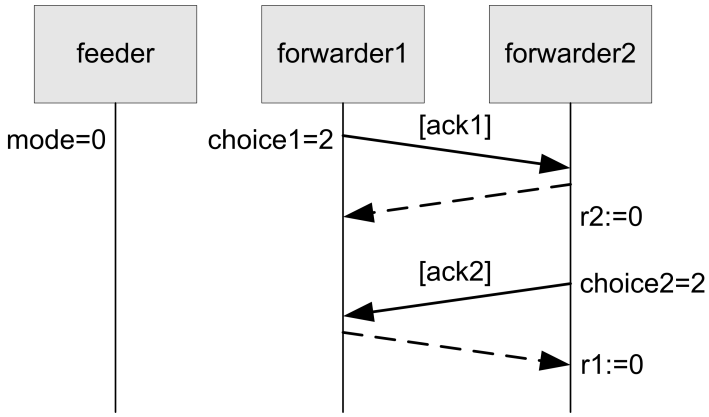


Figure 6.7: Semantics of the actions `ack1` and `ack2`

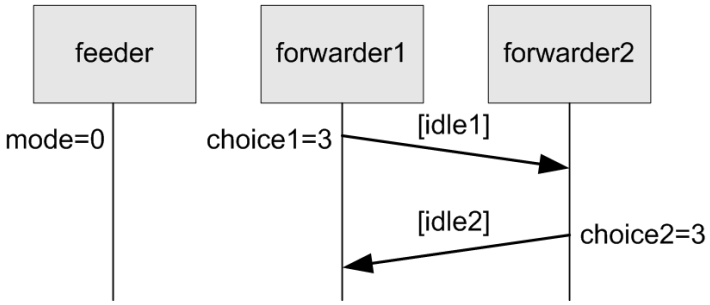


Figure 6.8: Semantics of the actions `idle1` and `idle2`

The choice amongst available actions is realised by the action `choose1` (see Figure 6.9).

The module `forwarder2` (see Appendix section C.2.4) describes the second node, and a similar structure can be used to describe further nodes running the protocol. As module `forwarder2` is defined by renaming of module `forwarder1` and nodes are only referenced by the generic indices defined above, its structure is completely analogous to module `forwarder1`. Also, the semantics of the defined abstractions is equivalent for both nodes 1 and 2.

6.2.3 The protocol

In addition to the rules for the behaviour of nodes, which have been described in section 6.2.2, the protocol comprises rules for dynamically assigning slots to nodes. To allow different scenarios, there is a network parameter `FS` for frame size as declared in Appendix section C.3.1.

In order to complete the specification of the protocol, we had to carefully model the semantics for all actions a node can perform. To support a succinct, high-level representation of the acknowleding and time-passing behaviour, complex boolean conditions are represented using abstractions defined in Appendix section C.3.2, for example, `last_slot` for "this is the last slot of the frame" and `next2` for "the next time slot may be allocated to node 2".

The protocol has been implemented in the PRISM module `slotCounter` (see Appendix section C.3.3). The semantics of the nodes' actions is as follows:

- `send1`, `send2`: node 1 or node 2, respectively, forwards a message. The next slot is owned by node 2 if this node still has an unused slot remaining, else by node 1 if this node has an unused slot remaining, and otherwise by no one. If this slot is the final

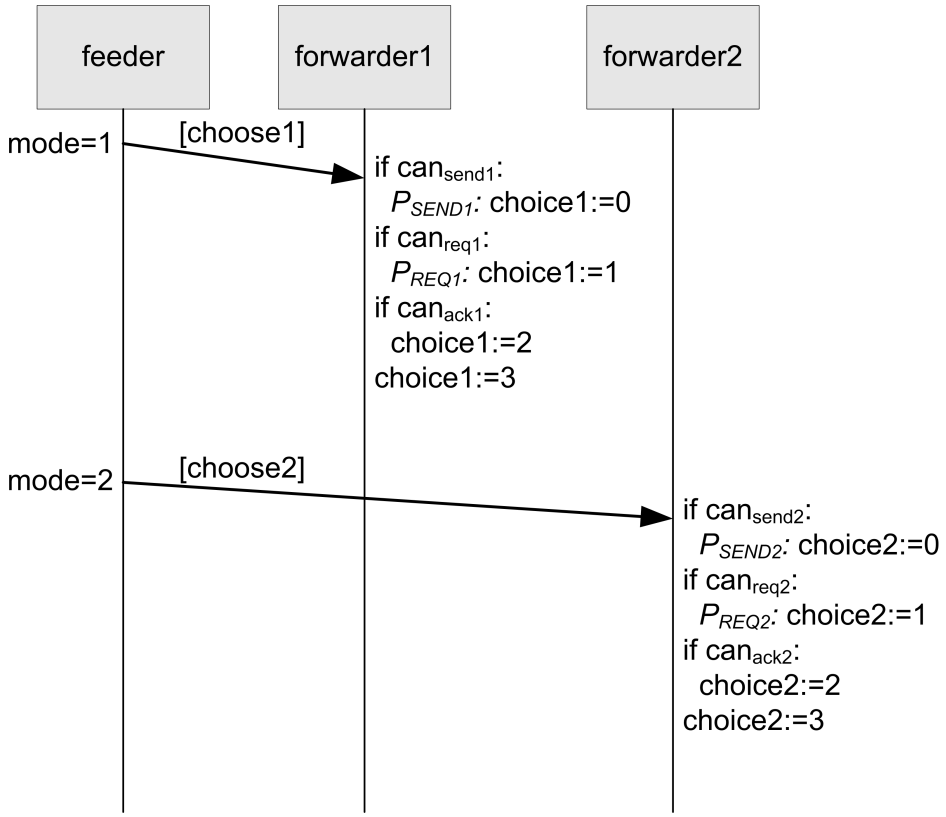


Figure 6.9: Semantics of the actions `choose1` and `choose2`

slot of the frame, the model resets its variables for the new frame.

- `req1`, `req2`: node 1 or node 2, respectively, request dynamic slots from another node. The semantics for allocating the next slot and initialising a new frame is equal to the one for the actions `send1` and `send2`.

- `ack1`, `ack2`: node 1 or node 2, respectively, attempts to acknowledge another node's request for dynamic slots. We consider the case of `ack1`, as both cases are fully symmetric to each other (see Figure 6.10).

  If the number of unallocated slots is sufficient for fulfilling node 2's request, that is, at least as large as the number of messages currently stored in node 2's buffer, this is done. Otherwise, only one of node 1's dynamic slots is reallocated to node 2.

  The semantics for allocating the next slot and initialising a new frame is equal to the one for the actions `send1` and `send2`.

- `idle`: time passes. The semantics for allocating the next slot and initialising a new frame is equal to the one for the actions `ack1` and `ack2`.

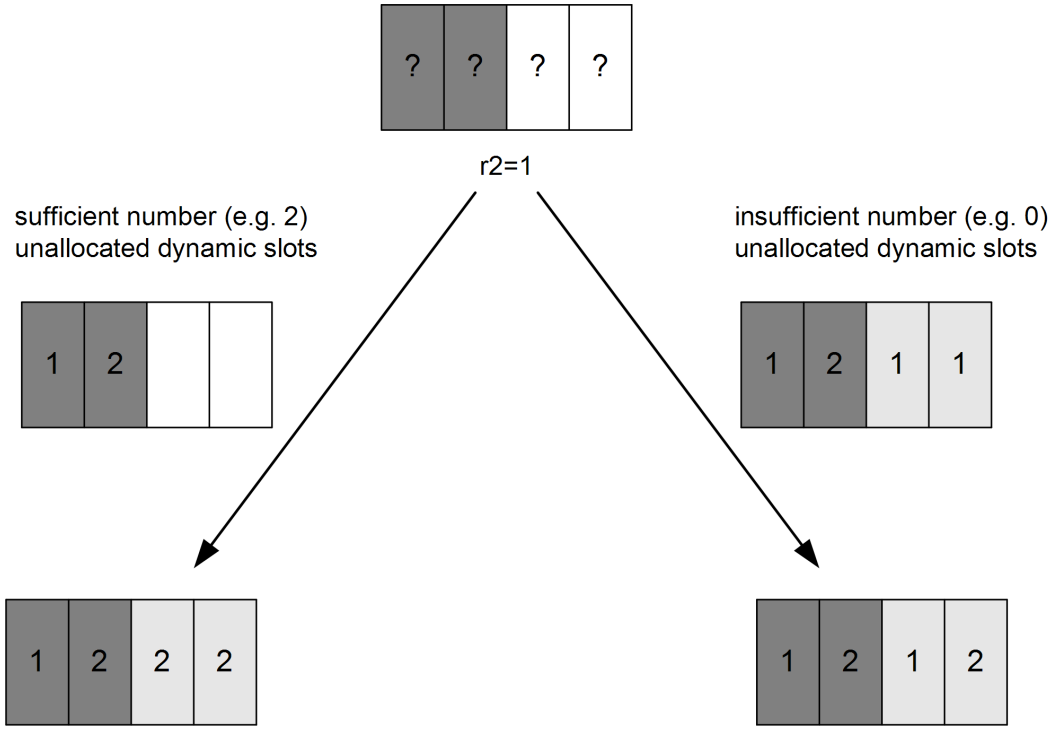
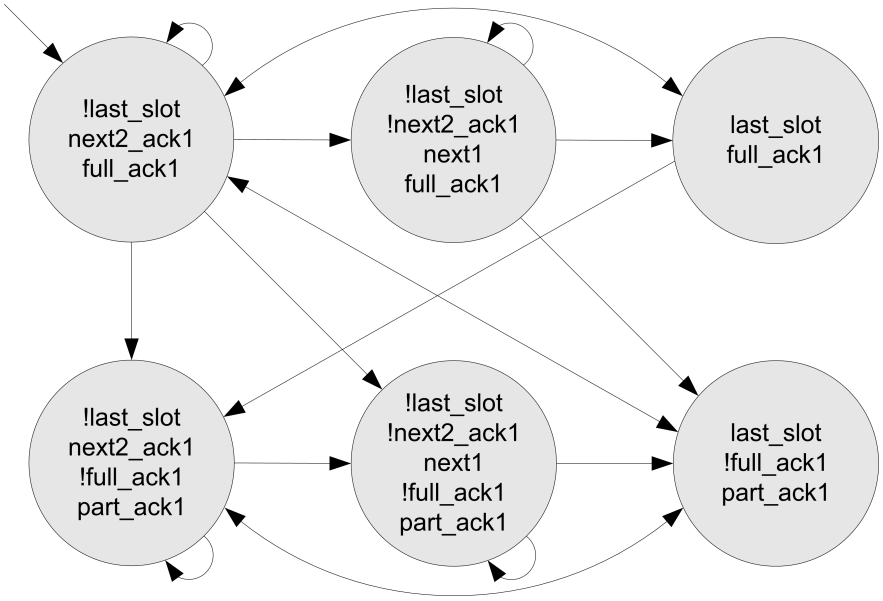


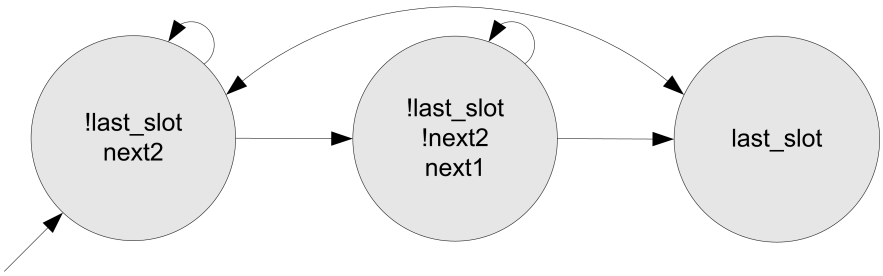
Figure 6.10: Slot allocation in a frame of size 4

In particular, acknowledgements of ownership requests become effective immediately (contrary to becoming effective in the next frame) and new allocations are valid until changed. It is possible, however, that a policy prescribes not to acknowledge another node's request even if there is a sufficient number of unallocated slots; the availability of slots is only checked when executing `ack1` or `ack2`, but such a policy never executes any of those actions.

Figure 6.11 shows a state-transition graph for the semantics of the protocol as given in the `slotCounter` module, using the abstractions over states defined in Appendix section C.3.2. Each circle and each arc refer to classes of states resulting from applying these abstractions. Subfigure 6.11a shows the model for action `ack1` and Subfigure 6.11b shows the model for the actions `idle1`, `req1`, and `send1`. Considering action `send1`, for example, there are three possible transitions on the abstract states shown: if the current slot is not the last slot of the frame (`!last_slot`), the next slot can be used by node 2 (`next2`) if possible (as slots are allocated via round robin amongst all nodes that have not used up their allocated number of slots and the current slot is used by node 1), by node 1 (`next1`) if not (`!next2`); otherwise, that is, if the current slot is the last slot of the frame (`last_slot`), the next state is described by `!last_slot` and `next2` (as the first state is always allocated to node 1 and at least one



(a) Semantics of `ack1`



(b) Semantics of `idle1`, `req1`, and `send1`

Figure 6.11: Labelled transition systems of the protocol

slot has been allocated to node 2).

6.2.4 Realistic wireless channel and radio models

In order to model the considered slot allocation protocols as realistically as sensible, we take into account the wireless channel and radio models with interference that have been introduced in Chapter 3 and used in Chapters 4 and 5. In contrast to existing models for wireless communication, they allow a significantly more accurate representation of spatial and interference characteristics.

As described in the previous chapters, we obtain models in three steps: graphical specification in CaVi, generation of PRISM code in CaVi, and manual enhancement of the PRISM models.

We have extended these models for the slot allocation protocol in order to investigate two questions: First, which properties does this protocol exhibit when considering a larger model with a realistic representation of channel and radio characteristics, including asynchronous and multi-hop communication? Second, how realistic and scalable are the novel wireless communication models?

In order to address these questions, we have computed probabilistic abstractions for reception probabilities using CaVi and subsequently enriched the existing PRISM models with those abstractions. For the modelling of wireless channel and radio, we have assumed the node in the network to be placed within a square grid of three times three nodes with a distance of 1.5 metres and the other network parameters as in Chapter 4.

For the enhanced model, all actions that involve an interaction with the channel (that is, requesting, sending, and acknowledging) have been adapted to the lossy-channel paradigm. As large networks make it more likely for messages to be lost before reaching their destination, guaranteeing a certain Quality of Service with respect to the number of delivered messages requires a higher messages input and thus a higher network load. Performance properties such as effective throughput can give valuable insights into this.

The PRISM model based on the wireless-channel and radio models from Chapter 3 is given in Appendix C. In addition to the abstractions discussed in the previous chapters, we use a new variable `p_nocoll` to abstract the reception probability between a sending and a receiving node. This abstraction is computed in CaVi, based on channel and radio parameters of the

CaVi model. All actions involving an interaction with the wireless channel have been modified accordingly, such that sending has been assigned a probability of `p_nocoll` representing a successful transmission and a probability of `1-p_nocoll` representing a message loss. The resulting PRISM code is given in Appendix section C.2.5.

6.3 Experiments and results

In this section, we present the analysis methodology used, describe concrete experiments, and report on results. The results can be distinguished into three groups: first, quantitative ones such as computed performance measures, second, qualitative ones such as optimal policy parameters with respect to a given performance measure, and third, nonnumerical trends such as interesting behaviours.

All experiments have been conducted using version 3.3 of PRISM, in particular its hybrid verification engine, on an Intel Core 2 Duo CPU (model number P9600, two cores, 2.66 GHz clock rate) with 2 GB of memory.

Performance characteristics are described using probability and expected reward measures. Numerical solution is performed using PRISM's default settings but with the *JOR* solution method, as other solution methods do not always converge within the set maximum number of iterations.

In order to find optimal policies[1] for different environments, even the policy space of this relatively small protocol is too large to be explored randomly. Therefore, a systematic approach based on iterative interpretation and improvement of experiments is used to find policies that are optimal for given scenarios. As our model is generic, it can be analysed for different sets of values, thus yielding optimal policies for different scenarios of fixed parameter values. However, the aim of this work is not primarily to find a provably optimal policy or to explore the whole policy space, but to demonstrate how a user can conduct experiments in order to find optimal policies for given scenarios with the desired level of confidence but a significantly reduced effort required.

Unless indicated otherwise, all experiments have been performed with the default param-

[1] In this chapter, "optimal policies" refers to finding protocols parameters that are optimal with respect to minimisation or maximisation of a given performance measure. We do not study optimal policies of MDPs as described by [81].

eters given in Table 6.1.

Parameter	Value
channel utilisation	0.5
reception-continuation probability	0.7
frame size	4
buffer size	2
P_SEND1, P_SEND2	0.9
P_REQ1, P_REQ2	1.0
ACK1, ACK2	0
REQ1, REQ2	0
BDIFF1, BDIFF2	-100

Table 6.1: Default parameter values for experiments

As the models used in this study are based on discrete time, time bounds correlate to numbers of transitions. All properties describing long-run behaviour are evaluated for a time limit of TIME_MAX $= 100,000$ steps, that is, until this maximum time has been reached.

6.3.1 Specification of performance properties

Quantitative performance measures of DTMC models are described using reward formulae of the PRISM property specification language [59], which extends the probabilistic temporal logic PCTL with support for reward-based measures. The properties evaluated in this work are expressed either as a single formula (for instance, throughput) or as an arithmetic expression over two formulae (for instance, delay – defined as an expression over buffer occupation and throughput). Each formula describes the expected value of a reward function that is defined over states and transitions. In PRISM, both types of reward functions are described using so-called *reward structures*. The remainder of this subsection describes all performance properties used in this work.

The reward structures for this chapter's models are listed in Appendix section C.4.

Simple reward structures and formulae

Note that, as each time slot is divided into four modes (for each of the different values of mode), the expected average slot length corresponds to four, and thus a reward of four is given in order to achieve an expected reward of one per time slot.

Expected message throughput The main performance property in the context of routing or gossiping is *message throughput*, which describes the number of messages forwarded. It has been implemented in the reward structure `throughput`. The *expected message throughput* describes the expected number of messages forwarded in the long run and is specified by the following formula:

$$\mathcal{R}_{\text{throughput}=?}[\text{C}^{\leq TIME\_MAX}].$$

In order to juxtapose locally and globally optimal policies, the message throughput for a single node (here node 1) is described by the reward structure `throughput1`.

Expected message loss The *message loss* describes the number of messages lost and has been implemented in the reward structure `loss`. The *expected message loss* describes the expected number of messages lost in the long run and is specified by the following formula:

$$\mathcal{R}_{\text{loss}=?}[\text{C}^{\leq TIME\_MAX}].$$

Expected average buffer occupation The *buffer occupation* describes the number of messages in the nodes' buffers and has been implemented in the reward structure `buffer`. The intuitive idea behind this definition is to measure the buffer occupation at the beginning of each slot, before any feeder or forwarder action occurs. The *expected average buffer occupation* describes the expected average number of messages in the nodes' buffers in the long run and is specified by the following formula:

$$\mathcal{R}_{\text{buffer}=?}[\text{C}^{\leq TIME\_MAX}].$$

Expected frequency of idle periods The *number of idle periods* describes the number of idle periods passed, that is, time slots in which neither requesting, nor acknowledging, nor forwarding is possible, and has been implemented in the reward structure `idle`. The *expected frequency of idle periods* describes the expected average number of idle periods in the long run and is specified by the following formula:

$$\mathcal{R}_{\text{idle}=?}[\text{C}^{\leq TIME\_MAX}].$$

Expected number of acknowledgements The *number of acknowledgements* describes the number of acknowledgements sent, and at the same time the number of reallocations, and has been implemented in the reward structure `acknowledged`. The *expected number of acknowledgements* describes the expected number of acknowledgements sent in the long run and is specified by the following formula:

$$\mathcal{R}_{\text{acknowledged=?}}[\text{C}^{\leq TIME\_MAX}].$$

Expected number of slots reallocated The *number of slots reallocated* has been implemented in the reward structure `reallocated`. The intuitive idea behind this definition is to measure the number of slots that are newly allocated to a node, either from another node or from being unallocated. The *expected number of slots reallocated* describes the expected number of slots reallocated in the long run and is specified by the following formula:

$$\mathcal{R}_{\text{reallocated=?}}[\text{C}^{\leq TIME\_MAX}].$$

Expected traffic The *traffic* describes the number of messages sent by the environment and has been implemented in the reward structure `traffic`. The *expected traffic* describes the expected average number of messages per time slot sent in the long run and is specified by the following formula:

$$\mathcal{R}_{\text{traffic=?}}[\text{C}^{\leq TIME\_MAX}].$$

Expected energy consumption The energy experiments are based on the IEEE 802.15.4/ZigBee device specified in Section 2.4.

As the slot allocation protocol is an abstract schema that does not define a frequency for its operation, its energy behaviour is described time-independently by giving reward values for a time duration of $1\,ms$. The reward values are based on the fact that, for acknowledging and requesting, both a receive operation (RX) and a transmit operation (TX) are required, while for sending only a TX operation is needed, for staying idle no operation is required, and during *choose* no time is passing. The resulting parameterisation of the transition reward function is given in Table 6.2, with the rewards in μWs: each `put` or `send` action corresponds to a transmission to one node; each `ack` or `req` action corresponds to a transmission from one

node to another node that is receiving; each `idle` action corresponds to both nodes being in the so-called "active mode"; each `choose` action corresponds to an instant computation thus no energy consumption.

Action	**energy consumption**
`put`	$320\mu s \times TX$
`send`	$320\mu s \times TX$
`ack`	$320\mu s \times (RX + TX)$
`req`	$320\mu s \times (RX + TX)$
`idle`	$320\mu s \times AM$
`choose`	$0\mu s$

Table 6.2: Calculations for energy reward function

The *energy consumption* describes the amount of energy in nWh consumed during one time slot and has been implemented in the reward structure `energy`. The *expected energy consumption* describes the expected amount of energy consumed per time slot in the long run and is specified by the following formula:

```
R{"energy"}=? [ S ]
```

Derived reward structures and formulae

In order to describe multidimensional Quality of Service properties, basic performance properties can be combined: for example, throughput with loss, throughput with delay, or throughput with loss and delay. As constrained optimisation problems cannot be solved within PRISM, such properties are implemented using arithmetic expressions over two reward functions.

Expected delay The *delay* describes the average number of time slots from reception until forwarding of a message. The *expected delay* is defined as the ratio of expected buffer occupation to expected message throughput.

Expected effective throughput The *effective throughput* describes the number of new messages forwarded, that is, messages lost are not counted.

Let t be the expected throughput and l be the expected number of lost messages. As we assume that lost messages are resent, the expected number of messages to be sent in each time slot is $t + l$. Thus, forwarding one message requires $\frac{t+l}{t}$ attempts, or – in other words

– the expected fraction of new messages is $\frac{t}{t+l}$. Hence, the *expected effective throughput* e is defined as the ratio of expected throughput t to the expected number of transmission attempts per message, given by

$$e \stackrel{\text{def}}{=} \frac{t^2}{t+l}.$$

6.3.2 Scalability of the analyses

An important aspect of formal methods is their scalability. To reason about this, space and time requirements have been evaluated with respect to frame and buffer size.

In Table 6.3 and Table 6.4, the expected throughput is computed for different frame sizes and for different buffer sizes, respectively. The tables show the number of states of the probabilistic model, its required space, the number of iterations performed by the underlying numerical method, and the time required for model construction and verification.

Frame size	**States**	**Space**	**Iterations**	**Verification time**
4	3665	0.3 MB	4753	0.63 s
6	9072	0.9 MB	10485	3.3 s
8	18109	1.6 MB	18411	11 s
10	31857	2.8 MB	28401	30 s
12	51390	4.2 MB	40334	66 s
14	77733	5.9 MB	54107	134 s
16	111960	7.9 MB	69663	244 s
18	154992	11.2 MB	86975	489 s
20	207867	13.9 MB	104712	1115 s

Table 6.3: Impact of frame size on accuracy, model size, and verification time (scenario: `BS`=2, `ACK`=0, `REQ`=0, `BDIFF`=-100), `P_SEND`=0.9, `P_REQ`=1)

Buffer size	**States**	**Space**	**Iterations**	**Verification time**
2	3665	0.3 MB	4753	0.63 s
4	10860	0.6 MB	4710	1.7 s
6	21912	1.0 MB	4620	3.3 s
8	36804	1.5 MB	4601	5.2 s
10	55536	2.1 MB	4590	7.6 s
12	78108	2.8 MB	4579	13 s
14	104520	3.7 MB	4574	16 s
16	134772	4.4 MB	4570	22 s
18	168864	5.3 MB	4568	28 s

Table 6.4: Impact of buffer size on accuracy, model size, and verification time (scenario: `FS`=4, `ACK`=0, `REQ`=0, `BDIFF`=-100), `P_SEND`=0.9, `P_REQ`=1)

The results show that the space and time requirements are exponential with respect to

the frame size, but the increase is less sensitive to buffer size than to frame size. This is because an increase of the frame size affects the ranges of many variables of the PRISM model, thus increasing the state space, while an increase in buffer size only affects values in the PRISM model, but within the existing state space. Overall, the analysis of medium-sized networks is feasible.

6.3.3 Optimising stochastic policy parameters

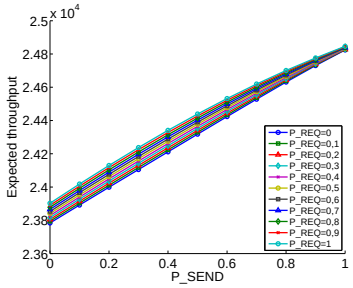
Four key performance properties – expected message throughput, expected message loss, expected effective throughput, and expected delay – have been analysed for different values of buffer size and frame size, respectively. In several experiments, optimal values for policy parameters under given scenarios (that is, for some parameters fixed) are determined, where optimality usually refers to maximum expected throughput.

As defined earlier, dynamic policies describe how a node, depending on its context, chooses whether to make a request, send an acknowledgement for another node's request, or forward a message. In this work, the context of a node comprises all nodes' buffer status, and choices can be made stochastically.

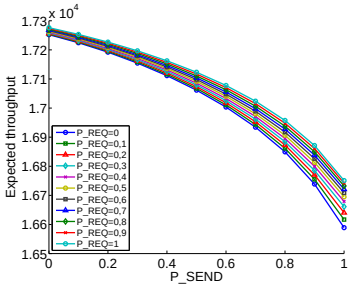
Policies are specified using two stochastic parameters (`P_SEND` and `P_REQ`) and three discrete parameters (`ACK`, `REQ`, and `BDIFF`). Each of the stochastic policy parameters influences the other stochastic one, and each of the discrete ones influence the other discrete ones.

The first experiment determines the relationship between stochastic policy parameters and traffic characteristics. For four combinations of channel utilisation and reception-probability, Figure 6.12 shows the expected throughput with respect to different values of `P_SEND` and `P_REQ`. The optimal values for the stochastic policy parameters highly depend on the traffic characteristics. For low channel utilisation of 0.25 and low reception-continuation probability of 0.1, the optimal values for sending and requesting inclination are 1.0 and 1.0, respectively. For a high reception-continuation probability of 0.9, the optimal values for sending and requesting inclination are 0.0 and 1.0, respectively. For high channel utilisation of 0.75 and low reception-continuation probability of 0.1, the optimal values for sending and requesting inclination are 0.9 and 1.0. For high reception-continuation probability of 0.9, the optimal values for sending and requesting inclination are 1.0 and 1.0.

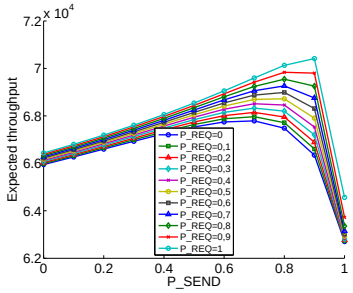
Generally, when the channel utilisation is low and the reception-continuation probability



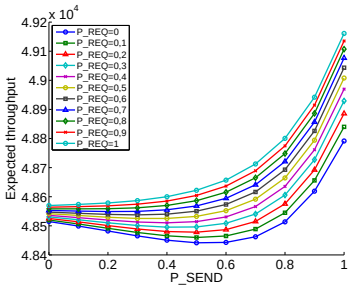
(a) channel util. 0.25, reception-cont. prob. 0.1



(b) channel util. 0.25, reception-cont. prob. 0.9



(c) channel util. 0.75, reception-cont. prob. 0.1



(d) channel util. 0.75, reception-cont. prob. 0.9

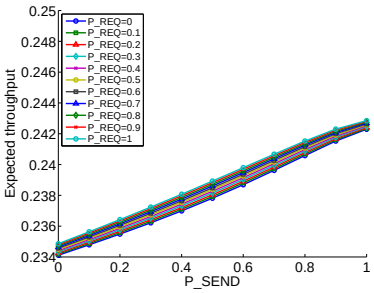
Figure 6.12: Optimal stochastic policy parameters for different traffic characteristics (scenario: `FS`=4, `BS`=2, `ACK`=0, `REQ`=0, `BDIFF`=-100)

is high, good policies favour requesting and acknowledging over sending. This is because the scenario is the one when it is most likely for dynamic slots to be available for reallocation. Under opposite traffic characteristics, optimal policies favour sending.

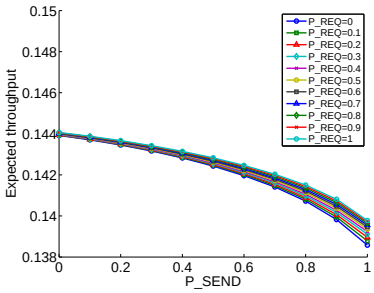
Figure 6.13 also shows the expected throughput, but for an additive interference model and a larger network. For a channel utilisation of 0.25, the expected throughput is almost identical to that for the standard model, but for a channel utilisation of 0.75, it is zero. It is likely that the latter is due to a low signal-to-noise ratio caused by the high channel utilisation.

Figure 6.14 shows the expected energy consumption with respect to different values of `P_SEND` and `P_REQ`. The results show that the minimum expected energy consumption decreases with higher sending probabilities. This is because higher sending probabilities lead to a faster completion of the protocol.

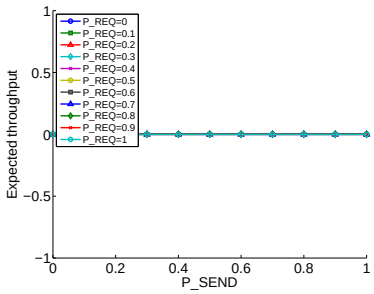
Figure 6.15 shows the expected energy consumption, but for an additive interference model and a larger network. For a channel utilisation of 0.25, the values are almost identical to those for the standard model, but for a channel utilisation of 0.75, they are constantly low.



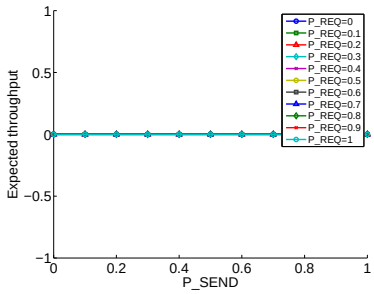
(a) channel util. 0.25, reception-cont. prob. 0.1



(b) channel util. 0.25, reception-cont. prob. 0.9

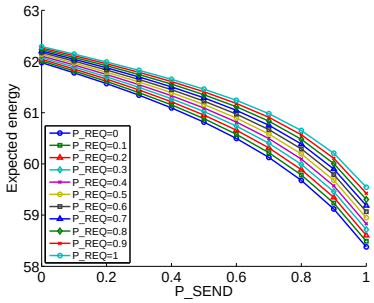


(c) channel util. 0.75, reception-cont. prob. 0.1

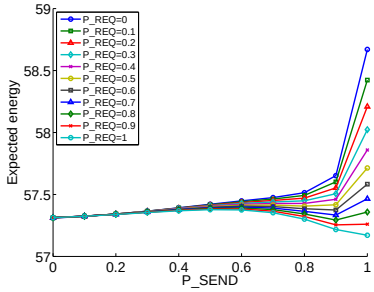


(d) channel util. 0.75, reception-cont. prob. 0.9

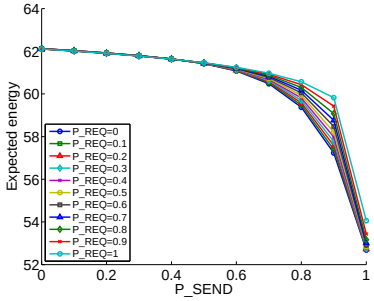
Figure 6.13: Optimal stochastic policy parameters for realistic wireless model and different traffic characteristics (scenario: FS=4, BS=2, ACK=0, REQ=0, BDIFF=-100)



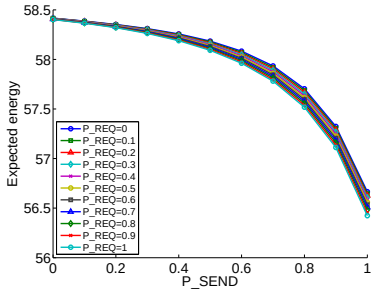
(a) channel util. 0.25, reception-cont. prob. 0.1



(b) channel util. 0.25, reception-cont. prob. 0.9

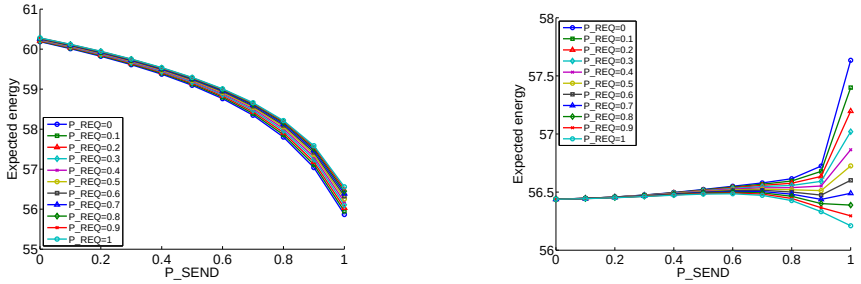


(c) channel util. 0.75, reception-cont. prob. 0.1



(d) channel util. 0.75, reception-cont. prob. 0.9

Figure 6.14: Energy-optimal stochastic policy parameters for different traffic characteristics (scenario: FS=4, BS=2, ACK=0, REQ=0, BDIFF=-100)



(a) channel util. 0.25, reception-cont. prob. 0.1

(b) channel util. 0.25, reception-cont. prob. 0.9

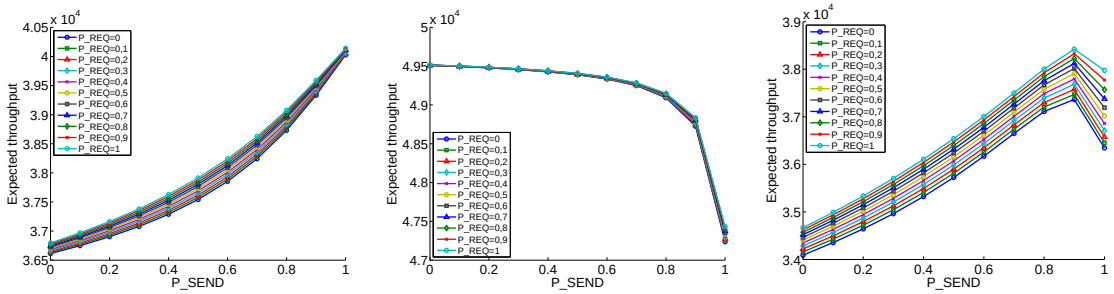
(c) channel util. 0.75, reception-cont. prob. 0.1

(d) channel util. 0.75, reception-cont. prob. 0.9

Figure 6.15: Energy-optimal stochastic policy parameters for realistic wireless model and different traffic characteristics (scenario: `FS`=4, `BS`=2, `ACK`=0, `REQ`=0, `BDIFF`=-100)

The latter correlates with the throughput of zero, and it is likely that a low signal-to-noise ratio caused by the high channel utilisation has led to sending and receiving not taking place and energy consumption resulting merely from listening.

The second experiment determines the relationship between stochastic policy parameters, frame size, and buffer size. For three combinations of frame size and buffer size, Figure 6.16 shows the expected throughput with respect to different values of `P_SEND` and `P_REQ`. The optimal value for sending inclination varies with both frame and buffer size while the optimal



(a) frame size 4, buffer size 2

(b) frame size 4, buffer size 8

(c) frame size 8, buffer size 2

Figure 6.16: Optimal stochastic policy parameters for different frame and buffer sizes, part 1 (scenario: `UTILISATION`=0.5, `PUT1`=0.7, `ACK`=0, `REQ`=0, `BDIFF`=-100)

value for requesting inclination is always 1.0.

The third experiment determines the relationship between stochastic policy parameters, frame size, and buffer size. For the four optimal value combinations of stochastic policy parameters that have been ascertained in the first experiment, Figure 6.17 shows expected values for throughput, message loss, effective throughput, and delay. The optimal combinations of sending and requesting inclination vary with different frame and buffer sizes. For high frame sizes, a sending inclination of 0.9 combined with a requesting inclination of 1.0 results in the highest throughput, highest effective throughput, lowest message loss, and lowest delay. With increasing buffer size, a sending inclination of 0.0 combined with a requesting inclination of 1.0 has analogous results.

6.3.4 Optimising discrete policy parameters

The second group of parameters that constitutes a node's adaptive policy are the discrete policy parameters. The values of $\texttt{ACK}i$, $\texttt{REQ}i$, and $\texttt{BDIFF}i$ describe how each node i chooses, depending on buffer occupation, whether it may request slots and whether it may acknowledge such requests. Again, optimisation is performed for given scenarios, that is, for some parameters fixed.

The first experiment determines optimal discrete policy parameters with respect to different traffic characteristics. Figure 6.18 shows the expected throughput. The optimal values for the discrete policy parameters are partially influenced by traffic characteristics. For low reception probabilities, a buffer occupation difference threshold of 3 is optimal. That is, optimal policies are static, as requesting is discouraged. For high reception probabilities, a buffer occupation threshold of -2 is optimal. Here, the optimal acknowledging and requesting thresholds are both 0. That is, under optimal policies, requesting is always allowed, but nodes only acknowledge requests if their own buffer is empty.

Figure 6.19 also shows the expected throughput, but for an additive interference model and a larger network. Both expected throughput and optimal discrete policy parameter values differ widely for these scenarios. This is probably because spatial characteristics are considered in the model used.

Figure 6.20 shows the expected energy consumption with respect to different traffic characteristics. The results indicate that all scenarios have $\texttt{ACK} = 2, \texttt{REQ} = 2$ with $\texttt{BDIFF} = 2$ or

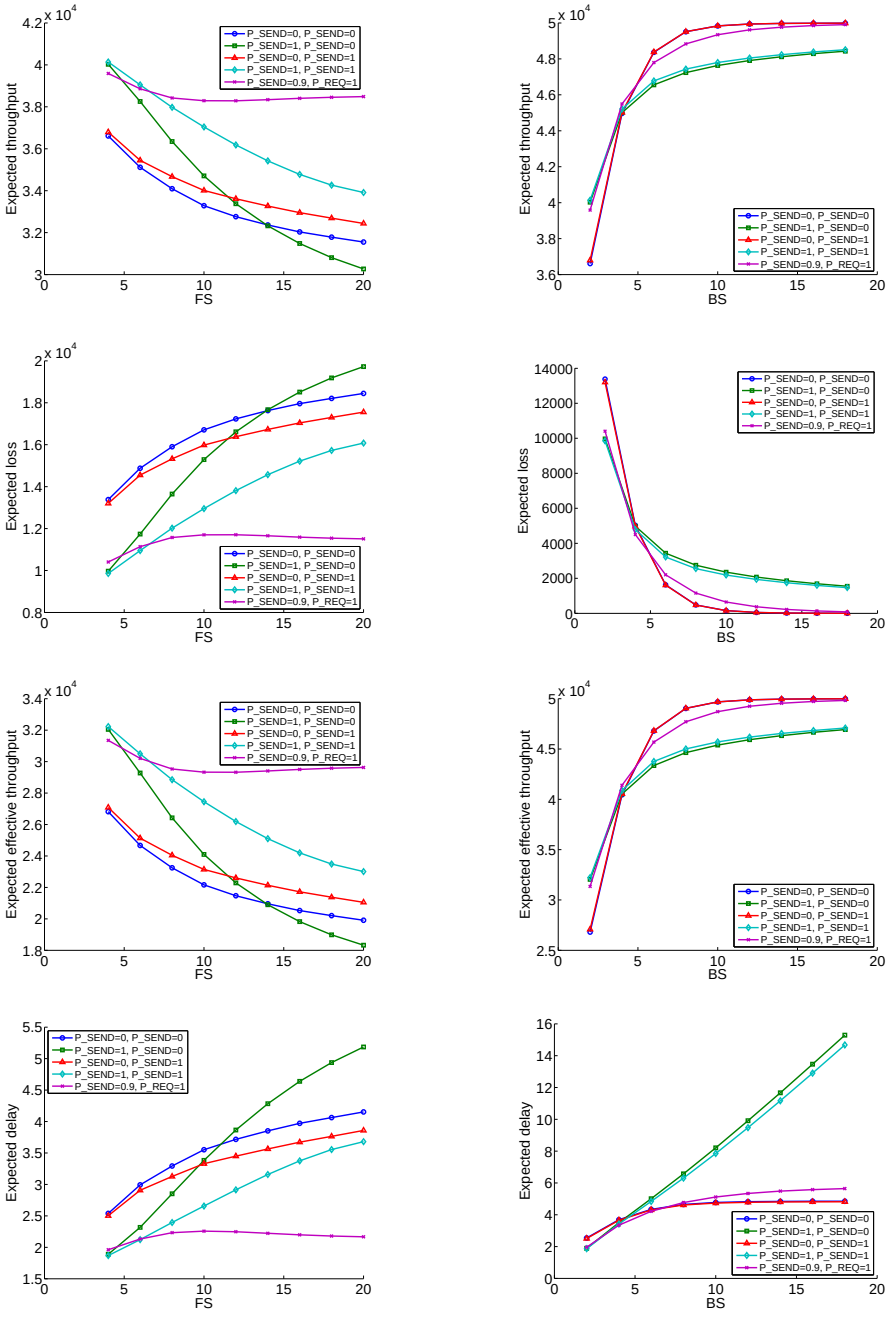
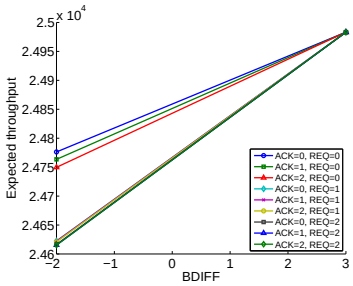
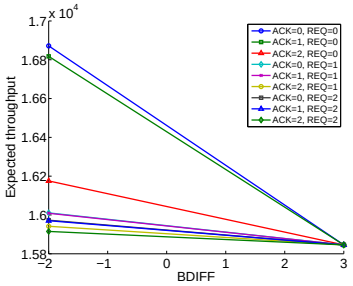


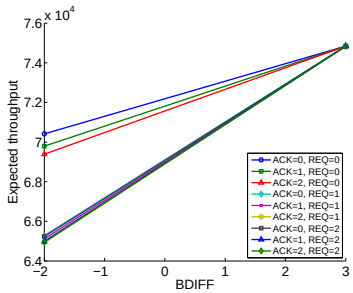
Figure 6.17: Optimal stochastic policy parameters for different frame and buffer sizes, part 2 (scenario: `UTILISATION`=0.5, `PUT1`=0.7, `FS`=4/`BS`=2, `ACK`=0, `REQ`=0, `BDIFF`=-100)



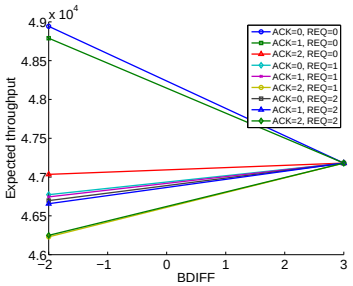
(a) channel util. 0.25, reception-cont. prob. 0.1



(b) channel util. 0.25, reception-cont. prob. 0.9

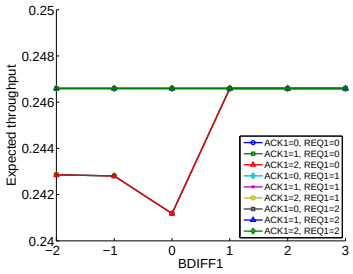


(c) channel util. 0.75, reception-cont. prob. 0.1

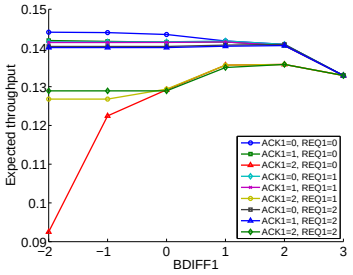


(d) channel util. 0.75, reception-cont. prob. 0.9

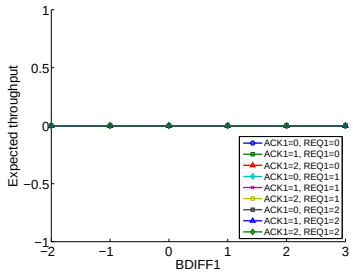
Figure 6.18: Optimal discrete policy parameters for different traffic characteristics (scenario: `FS`=4, `BS`=2, `P_SEND`/`P_REQ` optimal values from previous experiments)



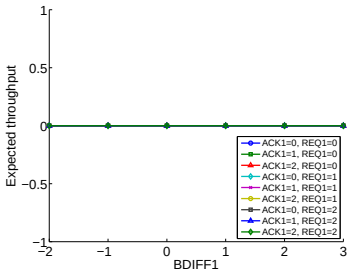
(a) channel util. 0.25, reception-cont. prob. 0.1



(b) channel util. 0.25, reception-cont. prob. 0.9

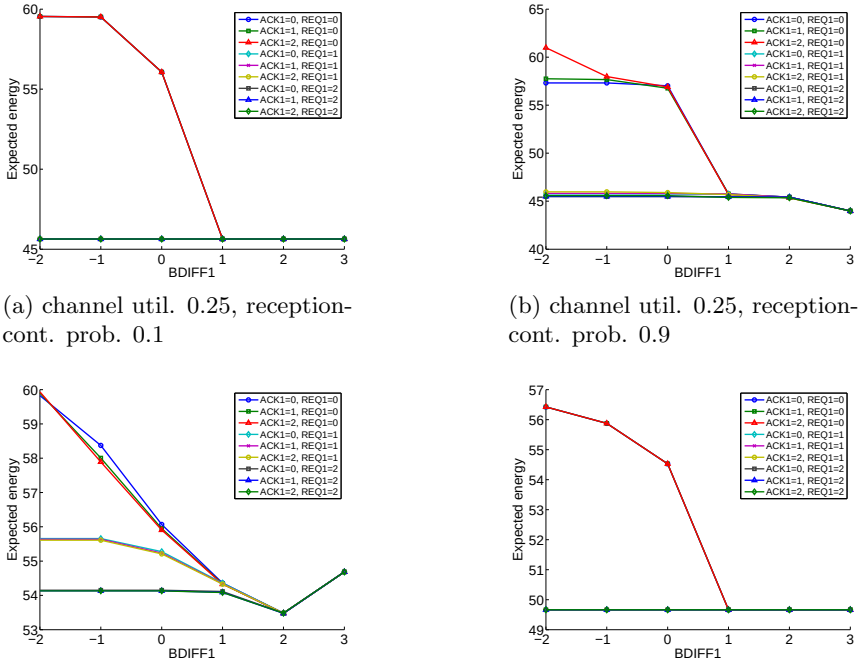


(c) channel util. 0.75, reception-cont. prob. 0.1



(d) channel util. 0.75, reception-cont. prob. 0.9

Figure 6.19: Optimal discrete policy parameters for realistic wireless model and different traffic characteristics (scenario: `FS`=4, `BS`=2, `P_SEND`/`P_REQ` optimal values from previous experiments)



(a) channel util. 0.25, reception-cont. prob. 0.1

(b) channel util. 0.25, reception-cont. prob. 0.9

(c) channel util. 0.75, reception-cont. prob. 0.1

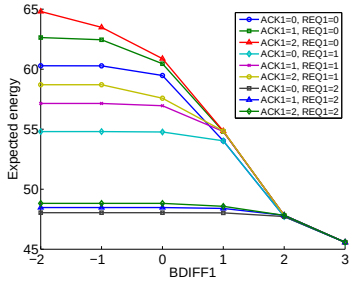
(d) channel util. 0.75, reception-cont. prob. 0.9

Figure 6.20: Energy-optimal discrete policy parameters and different traffic characteristics (scenario: `FS`=4, `BS`=2, `P_SEND`/`P_REQ` optimal values from previous experiments)

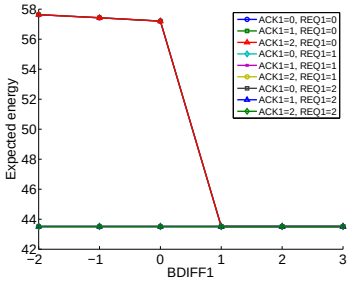
`BDIFF` = 3 as an optimal policy, with significant differences in the resulting energy consumptions.

Figure 6.21 shows the expected energy consumption with respect to different traffic characteristics, but for an additive interference model and a larger network. The results are in the same range as those for the standard model, with the same optimal discrete policy parameters.

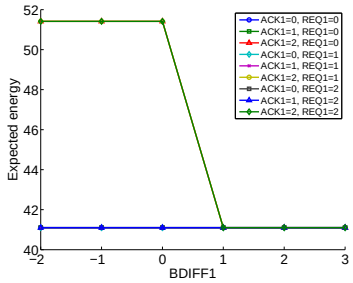
The second experiment determines optimal discrete policy parameters with respect to different frame and buffer sizes. The values for the acknowledging threshold are the optimal values from the previous experiments. Figure 6.22 shows expected values for throughput, message loss, effective throughput, and delay. For low frame sizes, an acknowledging threshold of 0 and a buffer occupation difference threshold of -100 are optimal, while for large frame sizes an acknowledging threshold of -1 is optimal. That is, for small frame sizes, under optimal policies, requesting is always allowed but nodes only acknowledge requests if their own buffer is empty; for large frame sizes, the simplest optimal policies are static. However, for all frame sizes there are optimal policies that are dynamic. For time-critical applications,



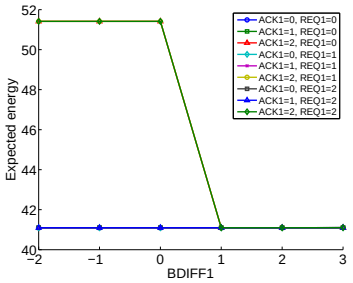
(a) channel util. 0.25, reception-cont. prob. 0.1



(b) channel util. 0.25, reception-cont. prob. 0.9



(c) channel util. 0.75, reception-cont. prob. 0.1



(d) channel util. 0.75, reception-cont. prob. 0.9

Figure 6.21: Energy-optimal discrete policy parameters for realistic wireless model and different traffic characteristics (scenario: `FS`=4, `BS`=2, `P_SEND`/`P_REQ` optimal values from previous experiments)

these policies may still have advantages over static ones.

For different buffer sizes, the optimal policies are always dynamic such that requesting is always allowed but nodes only acknowledge requests if their own buffer is empty.

A significant correlation between buffer size and expected delay can be observed. The reason for this is that, with smaller buffer sizes more messages are dropped, but those messages that are stored in the buffer are forwarded comparably faster. In practice, however, lost messages usually need to be resent. This can be represented more appropriately using a new performance measure *effective delay*, which can be defined analogously to effective throughput.

6.3.5 Optimal policies

The experiment conducted in this chapter approximate optimal policy parameters with respect to different scenarios (for example, traffic characteristics) by searching the parameter space.

In order to reduce the large space of parameter values, a number of combinations of chan-

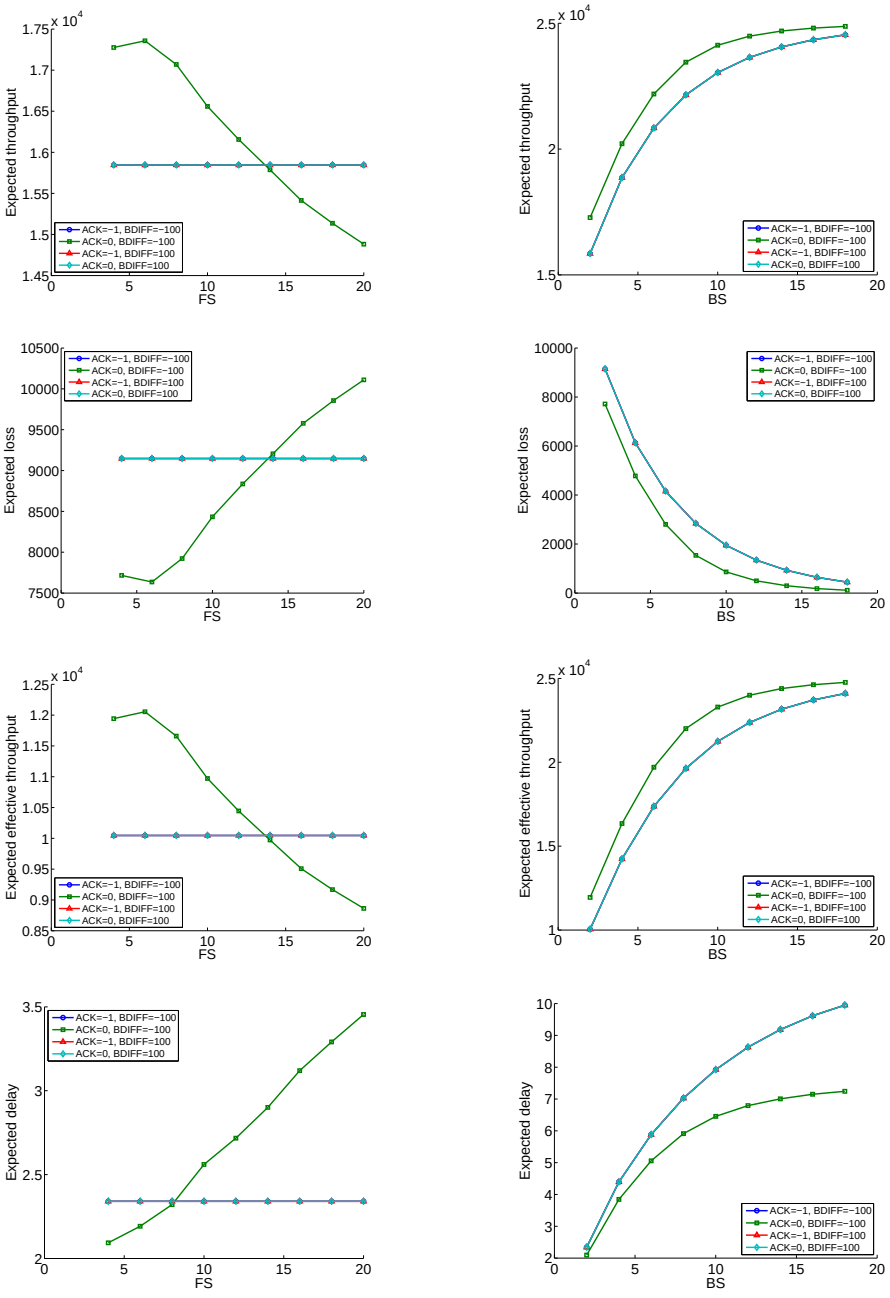
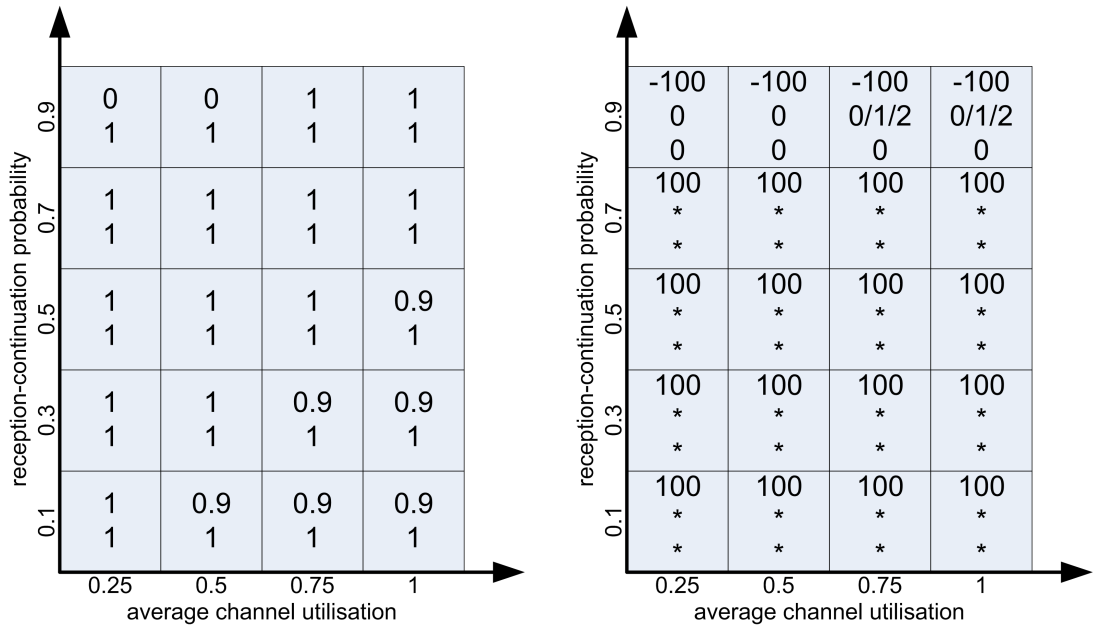


Figure 6.22: Optimal discrete policy parameters for different frame and buffer sizes (scenario: `UTILISATION`=0.25, PUT1=0.9, `FS`=4/`BS`=2, `P_SEND`=0, `P_REQ`=1, `REQ`=0



(a) P_SEND, P_REQ

(b) BDIFF, ACK, REQ

Figure 6.23: Optimal policies for different traffic characteristics (scenario: `FS`=4, `BS`=2, `ACK`=0,`REQ`=0, `BDIFF`=-100/`P_SEND`,`P_REQ` optimal values from previous experiments)

nel utilisation and reception-continuation probability have been identified such that they can be presented in partition graphs. Figure 6.23 shows the optimal stochastic policy parameters (in Figure 6.23a) and the optimal discrete policy parameters (in Figure 6.23b) for all partitions. Most policies favour sending over requesting and requesting over acknowledging. However, for low channel utilisation (up to 0.5 of the capacity) and a high reception-continuation probability of 0.9, requesting is favoured over sending; also, for low to medium reception probabilities (up to 0.5) and comparably high channel utilisation (from 0.5 of its capacity), the bias towards sending is slightly lower at 0.9.

The two cases where policies are less biased towards sending can be explained as follows: In the first case, a low channel utilisation allows nodes to accept other nodes' requests to give up time slots without a large performance risk. By doing this, it also becomes more likely for their own requests to be acknowledged, thus sending such requests becomes more effective, too. In the second case, while the channel utilisation is high, the lower reception-continuation probability makes it rather unlikely that frames are received in large blocks. Interestingly, the higher the reception-continuation probability (the more likely it is that frames are received in blocks and the longer such blocks are), the higher the required channel utilisation for a

sending bias of 0.9 to be effective. Thus, also in this case, a slight bias towards requesting time slots turns out to be beneficial.

For many of the considered classes of traffic characteristics, the optimal policies are trivial. That is, in these cases, the parameters related to buffer occupation (`BDIFF`, `ACK` and `REQ`) are such that they have no effect, and second, sending is always (with a probability of 1.0) favoured over receiving and receiving is always favoured over acknowledging. In the first case, the buffer sizes considered are too small for these parameters to be effective when the reception-continuation probability is 0.7 or less, otherwise they are effective. In the second case, the exchanges of control messages for requesting and acknowledging happen too often and consume too much time when the likelihood of successful re-allocation of time slots is too low and the costs of re-allocation outweighs the benefit, especially as the number of slots allocated is sometimes suboptimal (due to the protocol).

Trivial policies occur much less often under demanding traffic conditions, such as high channel utilisation combined with low reception-continuation probability and vice versa. Thus, it can be concluded that the protocol does impose adaptive policies on nodes when needed but remains silent otherwise. Hence, from this general point of view, the protocol is efficient.

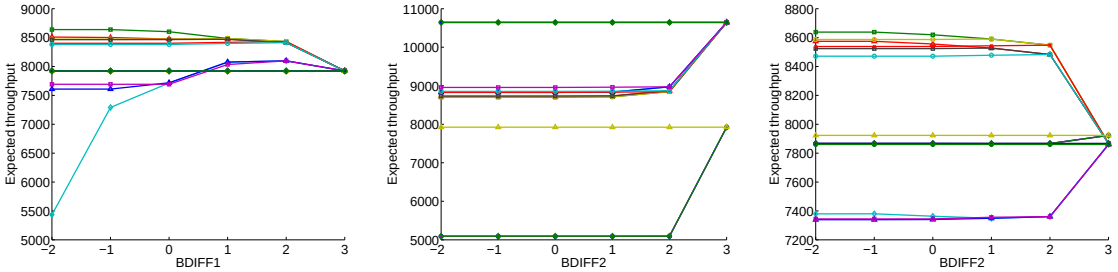
The interpretation of the obtained optimal policy parameters presented in this Chapter has been confirmed using PRISM's discrete-event simulator.

6.3.6 Local and global optimality

Policies that are locally optimal, that is, optimal for the performance of a single node, are not necessarily globally optimal, that is, optimal for the performance of a whole network.

In the protocol considered in this work, all nodes of the network collaborate on the common goal of forwarding received messages. That is, the objective of the protocol is to optimise global performance. As single nodes usually do not have a global view of the network, protocol design and implementation is responsible for modifying the existing performance measures such that local and global optimality coincide.

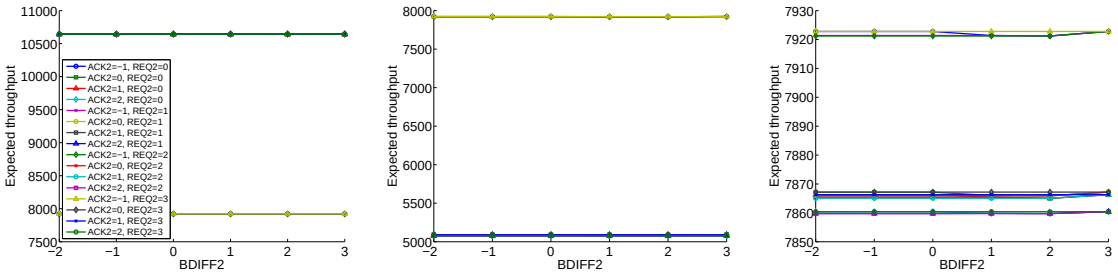
Situations where locally optimal node policies cannot be realised simultaneously are called *competition scenarios*. In order to establish policies for which local and global optimality coincides, combinations of the five policy parameters are qualitatively abstracted as "roles".



(a) equal policies, throughput per node

(b) globally optimal policy for node 1, throughput of node 1

(c) globally optimal policy for node 1, throughput per node



(d) locally optimal policy for node 1, throughput of node 1

(e) locally optimal policy for node 1, throughput of node 2

(f) locally optimal policy for node 1, throughput per node

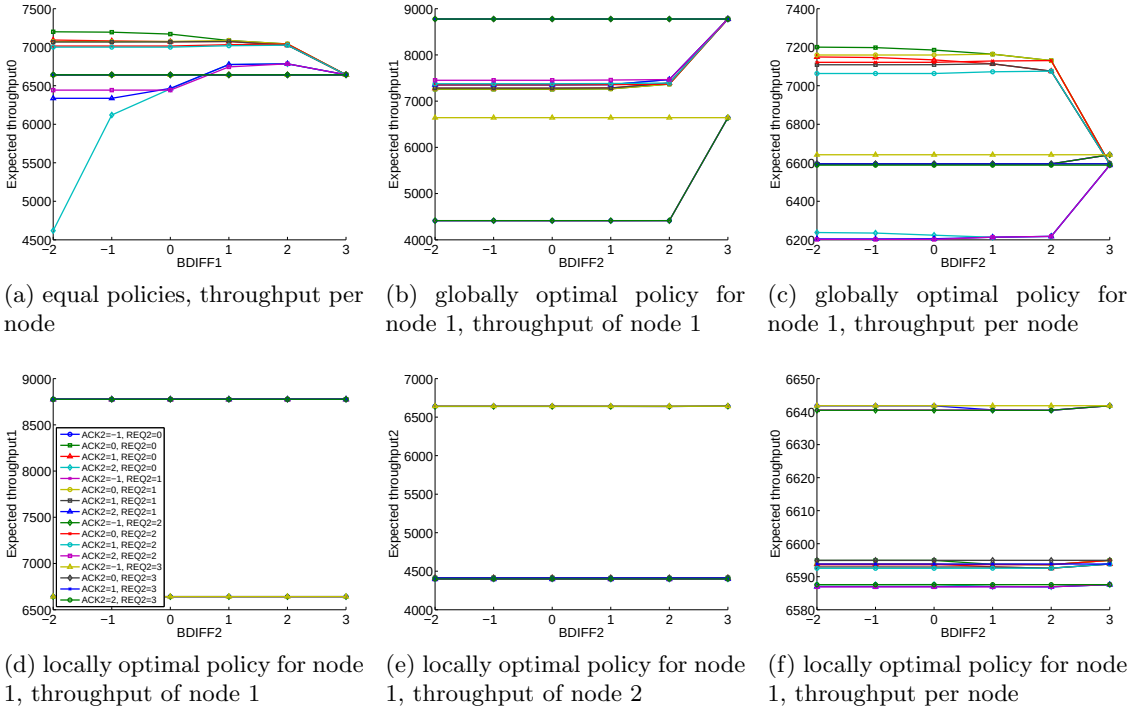
Figure 6.24: Locally and globally optimal discrete policy parameters for different traffic characteristics (scenario: `UTILISATION`=0.25, PUT1=0.9, `FS`=4, `BS`=2, `P_SEND`/`P_REQ` optimal values from previous experiments, time limit 100,000)

On one end of the spectrum, node policies can realise cooperative roles, on the other end they can realise non-cooperative, don't-care or competitive roles; obviously, most policies are located between these extremes.

In order to compare different competition scenarios, this experiment has been conducted to determine the optimal discrete policy parameters. Figure 6.24 shows the expected throughput. When both nodes use the same policy, the maximum expected throughput over all nodes is 8,638 and the minimum throughput is 5,434 (as shown in Subfigure 6.24a).

When node 1 uses the globally optimal policy, which is the policy with the maximum expected throughput over all nodes (`ACK` $= 0$, `REQ` $= 0$, `BDIFF` $= -2$), the maximum expected throughput is 10,643 and the minimum expected throughput is 5,092, depending on the policy of node 2 (as shown in Subfigure 6.24b). For node 2, the values are almost the same with a maximum of 10,643 and a minimum of 5,078. As shown in Subfigure 6.24c, the maximum expected throughput over all nodes is 8,638 and the minimum 7,339.

Subfigure 6.24d shows that the locally optimal policy for node 1, which is the most aggressive and least hesitant policy (`ACK` $= -1$, `REQ` $= 0$, `BDIFF` $= -2$), achieves a maximum



(a) equal policies, throughput per node

(b) globally optimal policy for node 1, throughput of node 1

(c) globally optimal policy for node 1, throughput per node

(d) locally optimal policy for node 1, throughput of node 1

(e) locally optimal policy for node 1, throughput of node 2

(f) locally optimal policy for node 1, throughput per node

Figure 6.25: Locally and globally optimal discrete policy parameters for realistic wireless model and different traffic characteristics (scenario: `UTILISATION`=0.25, PUT1=0.9, `FS`=4, `BS`=2, `P_SEND`/`P_REQ` optimal values from previous experiments, time limit 100,000)

expected throughput of 10,643 and a minimum expected throughput of 7,922 for this node. For node 2, the maximum expected throughput is 7,923 and the minimum is 5,076 (as shown in Subfigure 6.24e). As shown in Subfigure 6.24f, the maximum expected throughput over all nodes is 7,923 and the minimum is 7,860. Fully static policies always have an expected throughput of 7,923.

Thus, a node using a locally optimal policy instead of a globally optimal one increases its minimum expected throughput by 56 percent while its maximum expected throughput remains the same. With respect to the whole network, the minimum expected throughput achieved by locally optimal policies is seven percent higher than that for globally optimal policies, while the maximum throughput is eight percent lower. Hence, globally optimal policies can achieve a higher network performance than locally optimal policies, but they are less robust, meaning that they are significantly adversely affected by locally optimal policies of other nodes, where locally optimal policies are not.

Figure 6.25 shows the expected throughput for the same competition scenarios, but for an additive interference model and a larger network. The results show visibly lower values for

expected energy consumption, but the best and worst policies are the same for all scenarios. That is, as expected, general relationships between aggressive and hesitant local policies remain the same despite changes in channel and network model.

6.4 Conclusion

In this chapter, we have presented a comprehensive methodology based on probabilistic model checking for design, modelling, and analysis of dynamic slot allocation protocols. The general approach used in this work – based on generic models, formal abstractions, and step-wise state-space exploration – is applicable to a wide range of network protocols.

For a simple sample protocol developed as part of this study, a modular, generic model of dynamic slot allocation has been developed and high-level abstractions for traffic characteristics, node policies, and protocol semantics have been defined. The model has been shown to be very scalable, and the abstractions can easily be adapted to different, more complex node policies and protocol semantics. For example, the integration of better traffic models, based on time-dependent probability distributions, is a topic for future work. By iterative evaluation of performance properties and adjustment of parameters, policies have been systematically improved towards optimality without wasting efforts, and the state space has been explored efficiently.

This study has confirmed the suitability of probabilistic model checking for systems of high complexity, exceeding the capabilities of non-formal methods such as simulation: First, with respect to reliability, as model checking considers all possible executions of a model and delivers provably correct results by method, which is important for analysing corner-case scenarios and finding spurious behaviour; second, with respect to analysis breadth, where simple properties can be model-checked instantly while the execution time of a simulation does not change for small properties and is therefore higher; and third, with respect to analysis depth, as model-checking can deal with a very expressive temporal logic-based specification language and considers model paths of arbitrary length.

Hence, by combining the exhaustive formal method of probabilistic model checking with high-level models defined in the formal language of reactive modules, exact and repeatable quantitative and qualitative results have been obtained. Nevertheless, this work meets the

limits of today's probabilistic model checking tools. With increasing structural and numerical complexity of systems analysed, it becomes increasingly tedious to perform comprehensive analyses, and this consumes a significant part of users' productivity. Future work should improve tool support for managing modelling and analysis, including metamodelling, model structures, parameter dependabilities, experiment groups, and coverage analysis.

Experiments have confirmed that dynamic slot allocation protocols are an effective means of increasing performance and flexibility in scenarios of high traffic and under Quality of Service requirements. Interesting and sometimes counterintuitive effects of node policies, which are similar to anomalies that occur in real protocols, have been revealed, especially the high sensitivity of optimal parameter values to traffic characteristics, frame size, and buffer size. During this work, we have identified several aspects of protocol semantics and node policies where further performance improvements are possible. As protocol design is only a peripheral objective of this work, we do not discuss this further.

Node policies have also been compared with respect to local and global optimality. It has been shown that policies that are optimal for specific nodes do not generally achieve a combined performance that is optimal for the whole network. The theory of *networking games* provides results and methods [3, 4] for reducing the global (cooperative) problem to the local (non-cooperative) problem by adding rewards. Future work should explore the applicability of this game-theoretic approach for the specification of policies and probabilistic model checking for their computation and validation.

The scope of network and protocol detail can be further improved by applying abstraction and symmetry reduction techniques, as suggested in [55].

Chapter 7

Conclusion

In this thesis, we have successfully demonstrated how formal methods and models from the simulation community can be combined for the analysis of wireless network protocols. In studies of four network protocols, we have used probabilistic model checking, enhanced with probabilistic abstractions computed with our tool CaVi, to analyse Quality-of-Service and energy properties.

We conclude by summarising the main contributions of this thesis and reviewing directions for future research.

7.1 Thesis summary

Chapter 3 presented the first hybrid modelling and analysis approach that combines probabilistic model checking and simulation. It has been demonstrated how realistic wireless channel and radio models from the simulation domain can be combined with probabilistic model checking to obtain a powerful hybrid analysis framework that is supported by a clearly accessible tool for graphical specification and analysis.

Chapter 4 presented the application of the hybrid modelling and analysis approach to spatial properties of wireless sensor networks. Identified as the major factors influencing the integrity of wireless communication, distance, density, and spatial relationships have been analysed.

Chapter 5 presented the first application of probabilistic model checking to the standards IEEE 802.15.4 and ZigBee. As part of this work, we compared simple interference models

from the literature with novel, more realistic ones presented in Chapter 3. By modelling the full extent of the protocol, several abstractions and optimisations have been developed and various qualitative and quantitative performance properties have been analysed. Our hybrid analysis framework has proven to be both effective and efficient.

Chapter 6 presented the first application of probabilistic model checking to dynamic slot allocation protocols for wireless sensor networks. It has been demonstrated how optimal parameter values can systematically be determined from a large parameter space. Also, it has been shown how results from our hybrid analysis framework can be used to support network design decisions.

The presented modelling and analysis approach benefits from a combination of probabilistic model checking with its exhaustive state-space exploration and realistic wireless communication models with their detailled representation of topology, channel and radio. For the analysis of protocols for wireless sensor networks, we have demonstrated clear advantages of an approach based on generic models and combinations of the presented analysis formalisms. However, weaknesses such as the limitation to relatively small configurations and the disregarding of aspects such as mobility and three-dimensionality of node positions remain to be addressed in future work.

The graphical specification tool CaVi was used to provide visual performance indicators, which rely on internal runs of either Castalia or PRISM and support the rapid design of sensible network topologies and parameterisations as well as the rapid assessment of network behaviour. CaVi was used in two ways: First, to interactively find suitable topologies and parameters for channel and radio in order to specify the desired networks. Second, to compute probabilistic abstractions for reception probabilities of given networks in form of PRISM formulae. All these abstractions are based on wireless models from the simulation community, and have been empirically validated; however, a formal proof of their correctness would concern physical behaviour, which is beyond the scope of this thesis if provable at all; a validation using field experiments should be performed as part of future work.

7.2 Future work

The work in this thesis can be extended in various ways.

By combining probabilistic model checking with models from the simulation community, significant insights, in addition to those from pure PRISM analysis, can be gained. These results are transferable to other studies and real applications. Furthermore, a number of new problems has been highlighted, opening new directions of research.

The following topics would provide useful extensions to the main contributions made in this study:

- Validating the theoretical models and tool chains in systematic field experiments, against various real devices and under various environmental conditions.
- Establishing an improved process for using the tool chain in different development stages, for instance design, implementation, finding characteristic (faulty, critical, redundant) nodes and connections.
- Studying techniques for interleaving model checking and optimisation, for instance to guide it towards finding local extrema in parameter spaces.
- Showing the correctness of code generation from CaVi, that is, showing that CaVi models have the same behaviour as PRISM and Castalia models generated from them.
- Improving the expressiveness of CaVi, for instance by supporting new concepts such as mobility.
- Applying further state-space reduction techniques for probabilistic model checking such as symmetry reduction, where possible, in order to cope with larger models.
- Optimisation of both generated code and parsing engines of connected tools in order to make the analysis of large models feasible.
- Developing patterns for individual protocols and, moreover, formal languages to describe these patterns.
- Improving the integration of probabilistic model checking and simulation in order to improve instant analysis, debugging, and reporting of results.

Bibliography

[1] Norman Abramson. THE ALOHA SYSTEM – another alternative for computer communications. In *Proceedings of the 1970 Fall Joint Computer Conference*, volume 37, pages 281–285. AFIPS Press, 1970.

[2] ZigBee Alliance. Zigbee specification 1.0, 2005.

[3] Eitan Altman, Konstantin Avratchenkov, Nicolas Bonneau, Mrouane Debbah, Rachid El-Azouzi, and Daniel Sadoc Menasché. Constrained stochastic games in wireless networks. In *Proceedings of the 50th Global Communications Conference (GLOBECOM 2007)*, pages 315–320, 2007.

[4] Eitan Altman, Rachid El Azouzi, and Tania Jiménez. Slotted aloha as a game with partial information. *Computer Networks*, 45(6):701–713, 2004.

[5] R. Alur, A. Itai, R. P. Kurshan, and M. Yannakakis. Timing verification by successive approximation. *Information and Computation*, 118(1):142–157, 1995.

[6] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP 1990)*, volume 443, pages 322–335. Springer, 1990.

[7] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[8] Rajeev Alur and Thomas A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[9] Suzana Andova, Holger Hermanns, and Joost-Pieter Katoen. Discrete-time rewards model-checked. In *Proceedings of the First International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003)*, volume 2791 of *Lecture Notes in Computer Science*, pages 88–104. Springer, 2003.

[10] Søoren Asmussen and Peter W. Glynn. *Stochastic simulation: algorithms and analysis.* Springer, 2007.

[11] Rena Bakhshi, Lucia Cloth, Wan Fokkink, and Boudewijn Haverkort. Mean-field analysis for the evaluation of gossip protocols. In *Proceedings of the 6th International Conference on the Quantitative Evaluation of Systems (QEST 2009)*, pages 247–256. IEEE Computer Society Press, 2009.

[12] Rena Bakhshi and Ansgar Fehnker. On the impact of modelling choices for distributed information spread. In *Proceedings of the 6th International Conference on Quantitative Evaluation of Systems (QEST 2009).*, 2009.

[13] Rena Bakhshi, Daniela Gavidia, Wan Fokkink, and Maarten van Steen. An analytical model of information dissemination for a gossip-based protocol. *Computer Networks*, 53(13):2288–2303, 2009.

[14] Jerry Banks, II John S. Carson, Barry L. Nelson, and David M. Nicol. *Discrete-event system simulation*. Pearson, fifth edition, 2010.

[15] Athanassios Boulis. Castalia: A simulator for wireless sensor networks and body area networks, version 2.3. `http://castalia.npc.nicta.com.au/`, 2009.

[16] Athanassis Boulis, Ansgar Fehnker, Matthias Fruth, and Annabelle McIver. Cavi: Simulation and model checking for wireless sensor networks. In *Proceedings of the 5th International Conference on the Quantitative Evaluation of Systems (QEST 2008)*, pages 37–38, 2008.

[17] Marcel Busse, Thomas Haenselmann, and Wolfgang Effelsberg. Energy-efficient forwarding schemes for wireless sensor networks. In *Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2006)*, pages 125–133. IEEE Computer Society Press, 2006.

[18] Rachel Cardell-Oliver. Why flooding is unreliable. Technical report UWA-CSSE-04-001, The University of Western Australia, 2004.

[19] Castalia. `http://castalia.npc.nicta.com.au`.

[20] David Cavin, Yoav Sasson, and André Schiper. On the accuracy of MANET simulators. In *Proceedings of the 2nd ACM International Workshop on Principles of Mobile Computing (POMC 2002)*, pages 38–43, 2002.

[21] Dan Chalmers, Matthew Chalmers, Jon Crowcroft, Marta Kwiatkowska, Robin Milner, Eammon O'Neill, Tom Rodden, Vladimiro Sassone, and Morris Sloman. Ubiquitous computing: Experience, design and science, 2006. Draft of 23rd February 2006.

[22] APMC: Approximate Probabilistic Model Checker. `http://apmc.berbiqui.org/`.

[23] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceedings of the 1981 Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1982.

[24] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with Kronos. In *Proceedings of the 16th IEEE Real-Time Systems Symposium (RTSS 1995)*, pages 66–75. IEEE Computer Society Press, 1995.

[25] Akim Demaille, Thomas Hérault, and Sylvain Peyronnet. Probabilistic verification of sensor networks. In *Proceedings of the 14th IEEE International Conference on Computer Sciences, Research, Innovation and Vision for the Future (RIVF 2006)*, pages 45–54. IEEE Computer Society Press, 2006.

[26] Akim Demaille, Sylvain Peyronnet, and Benoît Sigoure. Modeling of sensor networks using XRM. In Tiziana Margaria, Anna Philippou, and Bernhard Steffen, editors, *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, pages 271–276, Paphos, Cyprus, November 2006. IEEE Computer Society.

[27] Cyrus Derman. *Finite State Markovian Decision Processes*. Academic Press, 1970.

[28] David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.

[29] Marie Duflot, Laurent Fribourg, Thomas Herault, Richard Lassaigne, Frédéric Magniette, Stéphane Messika, Sylvain Peyronnet, and Claudine Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. *Electronic Notes in Theoretical Computer Science*, 128(6):195–214, 2005.

[30] Marie Duflot, Marta Kwiatkowska, Gethin Norman, and David Parker. A formal analysis of Bluetooth device discovery. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(6):621–632, 2006.

[31] Ralph El Khoury and Rachid El Azouzi. Modeling the effect of forwarding in a multihop ad hoc networks with weighted fair queueing. In *Proceedings of the 3rd International Conference on Mobile Ad-Hoc and Sensor Networks (MSN 2007)*, volume 4864 of *Lecture Notes in Computer Science*, pages 5–18. Springer, 2007.

[32] Ansgar Fehnker, Matthias Fruth, Michael Ma, and Annabelle McIver. CaVi - a demonstration at the NICTA Techfest 2007, March 2007.

[33] Ansgar Fehnker, Matthias Fruth, and Annabelle McIver. Graphical modelling for simulation and formal analysis of wireless network protocols. In *Proceedings of the Workshop on Methods, Models and Tools for Fault Tolerance (MeMoT 2007) at the 7th International Conference on Integrated Formal Methods (IFM 2007)*, pages 80–87, July 2007. Technical Report CS-TR-1032, University of Newcastle upon Tyne.

[34] Ansgar Fehnker, Matthias Fruth, and Annabelle McIver. Graphical modelling for simulation and formal analysis of wireless network protocols. In *Methods, Models and Tools for Fault Tolerance*, volume 5454 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2009.

[35] Ansgar Fehnker and Peng Gao. Formal verification and simulation for performance analysis for probabilistic broadcast protocols. In *Proceedings of the 5th International Conference on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW 2006)*, volume 4104 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2006.

[36] Ansgar Fehnker, Lodewijk van Hoesel, and Angelika Mader. Modelling and verification of the lmac protocol for wireless sensor networks. In *Proceedings of the 6th International Conference on Integrated Formal Methods (IFM 2007)*, volume 4591 of *Lecture Notes in Computer Science*, pages 253–272. Springer, 2007.

[37] A. J. Field, U. Harder, and P. G. Harrison. Measurement and modelling of self-similar traffic in computer networks. *IEE Proceedings Communications*, 151(4):355–363, 2004.

[38] J. E. Flood. *Telecommunications Switching, Traffic and Networks*. Prentice Hall, 1995.

[39] Matthias Fruth. Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol. In Tiziana Margaria, Anna Philippou, and Bernhard Steffen, editors, *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*

(ISoLA 2006), pages 290–297, Paphos, Cyprus, November 2006. IEEE Computer Society.

[40] Jimmi Grönkvist. Distributed scheduling for mobile ad hoc networks – a novel approach. In *Proceedings of the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004)*, volume 2, pages 964–968, 2004.

[41] Christian Groß, Holger Hermanns, and Reza Pulungan. Does clock precision influence zigbee's energy consumptions? In *Proceedings of the 11th International Conference on Principles of Distributed Systems, (OPODIS 2007)*, volume 4878 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2007.

[42] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[43] Arnd Hartmanns. A Modest checker for probabilistic timed automata. Technical report 49, Sonderforschungsbereich Automatic Verification and Analysis of Complex Systems (AVACS), Saarland University, 2009.

[44] Arnd Hartmanns and Holger Hermanns. A modest approach to checking probabilistic timed automata. In *Proceeding of the 6th International Conference on the Quantitative Evaluation of Systems (QEST 2009)*, pages 187–196. IEEE Computer Society Press, 2009.

[45] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 2001)*, pages 146–159. ACM Press, 2001.

[46] Armin Heindl and Reinhard German. Performance modeling of IEEE 802.11 wireless LANs with stochastic Petri nets. *Performance Evaluation*, 44(1–4):139–164, 2001.

[47] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. A user guide to HyTech. In *Proceedings of the 1st International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer, 1995.

[48] Mong-Fong Horng and Yau-Hwang Kuo. Dynamic slot allocation to control delay in tdma wireless base station. In *Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC 2003)*, pages 1126–1131. IEEE Computer Society, 2003.

[49] Texas Instruments. CC2520 datasheet: 2.4 GHz IEEE 802.15.4/ZigBee RF transceiver, 2007.

[50] John G. Kemeny, J. Laurie Snell, and Anthony W. Knapp. *Denumerable Markov Chains*. Graduate Texts in Mathematics. Springer, second edition, 1976.

[51] Jaesub Kim and Kyu Ho Park. An energy-efficient, transport-controlled mac protocol for wireless sensor networks. *Computer Networks*, 53(11):1879–1902, 2009.

[52] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental evaluation of wireless. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation for Wireless and Mobile Systems (MSWiM 2004)*, pages 78–82, 2004.

[53] Byung-Jae Kwak, Nah-Oak Song, and Leonard E. Miller. Performance analysis of exponential backoff. *IEEE/ACM Transactions on Networking*, 13(2):343–355, 2005.

[54] M. Kwiatkowska, G. Norman, and A. Pacheco. Model checking expected time and expected reward formulae with random time bounds. *Computers and Mathematics with Applications*, 51(2):305–316, 2006.

[55] Marta Kwiatkowska, Gethin Norman, and Dave Parker. Analysis of a gossip protocol in PRISM. *ACM SIGMETRICS Performance Evaluation Review*, 36(3):17–22, 2008.

[56] Marta Kwiatkowska, Gethin Norman, Dave Parker, and Jeremy Sproston. Verification of real-time probabilistic systems. In *Modeling and Verification of Real-Time Systems: Formalisms and Software Tools*, pages 249–288. Wiley, 2008.

[57] Marta Kwiatkowska, Gethin Norman, and David Parker. Probabilistic model checking in practice: Case studies with PRISM. *ACM Performance Evaluation Review*, 32(4):16–21, 2005.

[58] Marta Kwiatkowska, Gethin Norman, and David Parker. Quantitative analysis with the probabilistic model checker PRISM. *Electronic Notes in Theoretical Computer Science*, 153(2):5–31, 2006.

[59] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *Formal Methods for Performance Evaluation, Proceedings of the 7th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM 2007)*, volume 4486 of *Lecture Notes in Computer Science*, pages 220–270. Springer, 2007.

[60] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.

[61] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic games for verification of probabilistic timed automata. In *Proceedings of the 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2009)*, volume 5813 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 2009.

[62] Marta Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29(1):33–78, 2006.

[63] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101–150, 2002.

[64] Marta Kwiatkowska, Gethin Norman, and Jeremy Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In *Proceedings of the 2nd Joint International Workshop on Process Algebra and Probabilistic Methods and Performance Modeling in Verification (PAPM-PROBMIV 2002)*, volume 2399 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2002.

[65] Marta Kwiatkowska, Gethin Norman, and Jeremy Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 14(3):295–318, 2003.

[66] Marta Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. Symbolic model checking for probabilistic timed automata. In *Proceedings of the Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant Systems (FORMATS-FTRTFT 2004)*, volume 3253 of *LNCS*, pages 293–308. Springer, 2004.

[67] YoungMin Kwon and Gul Agha. Scalable modeling and performance evaluation of wireless sensor networks. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2006)*, pages 49–58. IEEE Computer Society, 2006.

[68] GloMoSim: Global Mobile Information Systems Simulation Library. `http://pcl.cs.ucla.edu/projects/glomosim/`.

[69] PRISM manual. `http://www.prismmodelchecker.org/manual/`.

[70] A. K. McIver. Quantitative refinement and model checking for the analysis of probabilistic systems. In *Proceedings of the 14th International Symposium on Formal Methods (FM 2006)*, volume 4085 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2006.

[71] A. K. McIver and A. Fehnker. Formal techniques for the analysis of wireless networks. In *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, pages 263–270. IEEE Computer Society, 2006.

[72] Andrew Miner and David Parker. Symbolic representations and analysis of large probabilistic systems. In *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 296–338. Springer, 2004.

[73] Jelena Mišić, Shairmina Shafi, and Vojislav B. Mišić. Performance of a beacon enabled ieee 802.15.4 cluster with downlink and uplink traffic. *IEEE Transactions on Parallel Distributed Systems*, 17(4):361–376, 2006.

[74] Andrew D. Myers. *Hybrid MAC Protocols For Mobile Ad Hoc Networks*. PhD thesis, University of Texas at Dallas, 2002.

[75] Marcel Neugebauer, Jrn Plönnigs, and Klaus Kabitzsch. A new beacon order adaptation algorithm for IEEE 802.15.4 networks. In *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 302–311, 2005.

[76] The Network Simulator ns 2. `http://www.isi.edu/nsnam/ns/`.

[77] OPNeT. `http://www.opnet.com/`.

[78] David Anthony Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.

[79] Hai N. Pham, Dimosthenis Pediaditakis, and Athanassios Boulis. From simulation to real deployments in WSN and back. In *Proceedings of the 2007 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007)*, pages 1–6. IEEE Computer Society Press, 2007.

[80] PRISM. `http://www.prismmodelchecker.org/`.

[81] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1994.

[82] Yuvraj Krishna Rana, Bao Hua Liu, Alfandika Nyandoro, and Sanjay Jha. Bandwidth aware slot allocation in hybrid MAC. In *Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN 2006)*, pages 89–96. IEEE Computer Society Press, 2006.

[83] Injong Rhee, Ajit Warrier, Mahesh Aia, Jeongki Min, and Mihail L. Sichitiu. Z-MAC: A hybrid MAC for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 16(3):511–524, 2008.

[84] J. J. M. M. Rutten, Marta Kwiatkowska, Gethin Norman, and David Parker. Modelling and verification of probabilistic systems. In Prakash Panangaden and Franck van Breugel, editors, *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.

[85] Mathijs Schuts, Feng Zhu, Faranak Heidarian, and Frits Vaandrager. Modelling clock synchronization in the chess gMAC WSN protocol. In *Proceedings of the 1st Workshop on Quantitative Formal Methods: Theory and Applications (QFM 2009)*, volume 13 of *Electronic Proceedings in Theoretical Computer Science*, pages 41–54, 2009.

[86] Karim Seada, Marco Zuniga, Ahmed Helmy, and Bhaskar Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004*, pages 108–121. ACM Press, 2004.

[87] Oliver Sharma, Jonathan Lewis, Alice Miller, Al Dearle, Dharini Balasubramaniam, Ron Morrison, and Joe Sventek. Towards verifying correctness for wireless sensor network applications using insense and spin. In *Proceedings of the 16th International SPIN Workshop on Model Checking Software (SPIN 2009)*, volume 5578 of *Lecture Notes in Computer Science*, pages 223–240. Springer-Verlag, 2009.

[88] Chandramani Kishore Singh, Anurag Kumar, and P. M. Ameer. Performance evaluation of an IEEE 802.15.4 sensor network with a star topology. *Wireless Networks*, 14(4):543–568, 2008.

[89] IEEE Computer Society. IEEE standard 802.15.4-2003: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), 2003.

[90] Tony Sun, Ling-Jyh Chen, Chih-Chieh Han, Guang Yang, and Mario Gerla. Measuring effective capacity of IEEE 802.15.4 beaconless mode. In *Proceedings of the 2006 IEEE Wireless Communications and Networking Conference (WCNC 2006)*, volume 4, pages 493–498, 2006.

[91] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks*. Pearson, fifth edition, 2010.

[92] Zhenyu Tang and J. J. Garcia-Luna-Aceves. A protocol for topology-dependent transmission scheduling in wireless networks. In *Proceedings of the 1999 IEEE Wireless Communications and Networking Conference (WCNC 1999)*, volume 3, pages 1333–1337, 1999.

[93] Stavros Tripakis. *The formal analysis of timed systems in practice*. PhD thesis, Université Joseph Fourier, 1998.

[94] Stavros Tripakis. Timed diagnostics for reachability properties. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 1999)*, volume 1579 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 1999.

[95] Stavros Tripakis. Verifying progress in timed systems. In *Proceedings of the 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems (ARTS 1999)*, volume 1601 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 1999.

[96] Stavros Tripakis, Sergio Yovine, and Ahmed Bouajjani. Checking timed büchi automata emptiness efficiently. *Formal Methods in System Design*, 26(3):267–292, 2005.

[97] Simon Tschirner, Liang Xuedong, and Wang Yi. Model-based validation of qos properties of biomedical sensor networks. In *Proceedings of the 8th ACM/IEEE International Conference on Embedded Software (EMSOFT 2008)*, pages 69–78. ACM Press, 2008.

[98] Haidi Yue, Henrik Bohnenkamp, and Joost-Pieter Katoen. Analyzing energy consumption in a gossiping mac protocol. In *Proceedings of the 15th International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT 2010)*, volume 5987 of *Lecture Notes in Computer Science*, pages 107–119. Springer, 2010.

[99] Haidi Yue and Joost-Pieter Katoen. Leader election in anonymous radio networks: Model checking energy consumption. In *Proceedings of the 17th International Conference on Analytical and Stochastic Modeling Techniques and Applications (ASMTA 2010)*, volume 6148 of *Lecture Notes in Computer Science*, pages 247–261. Springer, 2010.

[100] Ender Yüksel, Hanne Riis Nielson, Flemming Nielson, Matthias Fruth, and Marta Kwiatkowska. Optimizing key updates in sensor networks. In *Proceedings of the 2011 IEEE Sensors Applications Symposium (SAS 2011)*, pages 82–87. IEEE Computer Society, 2011.

[101] Hafedh Zayani, Kamel Barkaoui, and Rahma Ben Ayed. Probabilistic verification and evaluation of backoff procedure of the WSN ECo-MAC protocol. *CoRR*, abs/1005.2050, 2010.

[102] Marco Zuniga and Bhaskar Krishnamachari. Analyzing the transitional region in low power wireless links. In *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, pages 517–526. IEEE Computer Society Press, 2004.

Appendix A

PRISM model for semi-formal analysis of spatial properties for wireless sensor networks

This appendix contains the source code of the PRISM models for the study on *semi-formal analysis of spatial properties for wireless sensor networks*, which has been presented in Chapter 4.

A.1 Model of additive interference

```
const double NETWORK_SIZE = 4;
const double utilisation1;
const double utilisation2 = utilisation1;
const double utilisation3 = utilisation1;

const double p_send0 = 1.0;
const double p_send1 = utilisation1 / NETWORK_SIZE;
const double p_send2 = utilisation2 / NETWORK_SIZE;
const double p_send3 = utilisation3 / NETWORK_SIZE;

formula all_complete = active0=0 & active1=0 & active2=0 & active3=0;
formula complete = active3=0;
```

```
const double PsendingNode0 = p_send0;
const double PsendingNode1 = p_send1;
const double PsendingNode2 = p_send2;
const double PsendingNode3 = p_send3;

const double linRxSignal_0_1 = 3.162277660168379E-7;
const double linRxSignal_0_2 = 3.162277660168379E-7;
const double linRxSignal_0_3 = 5.991395796778684E-8;
const double linRxSignal_1_0 = 3.162277660168379E-7;
const double linRxSignal_1_2 = 5.991395796778684E-8;
const double linRxSignal_1_3 = 3.162277660168379E-7;
const double linRxSignal_2_0 = 3.162277660168379E-7;
const double linRxSignal_2_1 = 5.991395796778684E-8;
const double linRxSignal_2_3 = 3.162277660168379E-7;
const double linRxSignal_3_0 = 5.991395796778684E-8;
const double linRxSignal_3_1 = 3.162277660168379E-7;
const double linRxSignal_3_2 = 3.162277660168379E-7;

formula snr_0_1 = (linRxSignal_0_1*send0)/(linRxSignal_2_1*send2
    + linRxSignal_3_1*send3 + 1.0E-10);
formula snr_0_2 = (linRxSignal_0_2*send0)/(linRxSignal_1_2*send1
    + linRxSignal_3_2*send3 + 1.0E-10);
formula snr_0_3 = (linRxSignal_0_3*send0)/(linRxSignal_1_3*send1
    + linRxSignal_2_3*send2 + 1.0E-10);
formula snr_1_0 = (linRxSignal_1_0*send1)/(linRxSignal_2_0*send2
    + linRxSignal_3_0*send3 + 1.0E-10);
formula snr_1_2 = (linRxSignal_1_2*send1)/(linRxSignal_0_2*send0
    + linRxSignal_3_2*send3 + 1.0E-10);
formula snr_1_3 = (linRxSignal_1_3*send1)/(linRxSignal_0_3*send0
    + linRxSignal_2_3*send2 + 1.0E-10);
formula snr_2_0 = (linRxSignal_2_0*send2)/(linRxSignal_1_0*send1
    + linRxSignal_3_0*send3 + 1.0E-10);
formula snr_2_1 = (linRxSignal_2_1*send2)/(linRxSignal_0_1*send0
    + linRxSignal_3_1*send3 + 1.0E-10);
formula snr_2_3 = (linRxSignal_2_3*send2)/(linRxSignal_0_3*send0
    + linRxSignal_1_3*send1 + 1.0E-10);
formula snr_3_0 = (linRxSignal_3_0*send3)/(linRxSignal_1_0*send1
    + linRxSignal_2_0*send2 + 1.0E-10);
```

```
formula snr_3_1 = (linRxSignal_3_1*send3)/(linRxSignal_0_1*send0
    + linRxSignal_2_1*send2 + 1.0E-10);
formula snr_3_2 = (linRxSignal_3_2*send3)/(linRxSignal_0_2*send0
    + linRxSignal_1_2*send1 + 1.0E-10);

formula Preceive_0_1 = func(max,0,(snr_0_1>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_0_1)), 200):0);
formula Preceive_0_2 = func(max,0,(snr_0_2>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_0_2)), 200):0);
formula Preceive_0_3 = func(max,0,(snr_0_3>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_0_3)), 200):0);
formula Preceive_1_0 = func(max,0,(snr_1_0>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_1_0)), 200):0);
formula Preceive_1_2 = func(max,0,(snr_1_2>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_1_2)), 200):0);
formula Preceive_1_3 = func(max,0,(snr_1_3>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_1_3)), 200):0);
formula Preceive_2_0 = func(max,0,(snr_2_0>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_2_0)), 200):0);
formula Preceive_2_1 = func(max,0,(snr_2_1>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_2_1)), 200):0);
formula Preceive_2_3 = func(max,0,(snr_2_3>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_2_3)), 200):0);
formula Preceive_3_0 = func(max,0,(snr_3_0>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_3_0)), 200):0);
formula Preceive_3_1 = func(max,0,(snr_3_1>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_3_1)), 200):0);
formula Preceive_3_2 = func(max,0,(snr_3_2>=1.5447406972503184)?
    func(pow,(1-0.5*func(pow,2.71828,-0.5 *4.0 * snr_3_2)), 200):0);

formula recvp0 = func(min,1,Preceive_1_0+Preceive_2_0+Preceive_3_0)>1.0?
    1.0:func(min,1,Preceive_1_0+Preceive_2_0+Preceive_3_0);
formula recvp1 = func(min,1,Preceive_0_1+Preceive_2_1+Preceive_3_1)>1.0?
    1.0:func(min,1,Preceive_0_1+Preceive_2_1+Preceive_3_1);
formula recvp2 = func(min,1,Preceive_0_2+Preceive_1_2+Preceive_3_2)>1.0?
    1.0:func(min,1,Preceive_0_2+Preceive_1_2+Preceive_3_2);
formula recvp3 = func(min,1,Preceive_0_3+Preceive_1_3+Preceive_2_3)>1.0?
    1.0:func(min,1,Preceive_0_3+Preceive_1_3+Preceive_2_3);
```

A.2 Model of the initial node

```
module node0
        active0:[0..2] init 1; // 0...can receive messages; 1...can send and receive messages;
                        // 2...can send and receive, synchronously and asynchronously
        send0:  [0..1] init 0; // 0...does not possess uncorrupted message;
                        // 1...does possess uncorrupted message, which it must forward

// probabilistic choice between sending and not-sending
[tick] send0=0&active0=1 -> PsendingNode0:(send0'=1)&(active0'=1)
                         +(1-PsendingNode0):(send0'=0)&(active0'=0);

// sending
[tick] send0=1&active0=1 -> (send0'=0)&(active0'=0);

// time passing
[tick] active0=0 -> (send0'=0)&(active0'=0);
endmodule
```

A.3 Model of a non-initial node

```
module node1
        active1:[0..2] init 1;
        send1:  [0..1] init 0;

// first, nondeterministic choice between synchronous and asynchronous reception
//   second, probabilistic reception due to noisy channel
//     if uncorrupted message received, probabilistic choice between sending and not sending
//     otherwise, waiting for next uncorrupted message
[tick] send1=0&active1=1 -> recvp1*PsendingNode1:(send1'=1)&(active1'=1)
                         + recvp1*(1-PsendingNode1):(send1'=0)&(active1'=0)
                         + (1 - recvp1):(send1'=0)&(active1'=1);
[tick] send1=0&active1=1 -> recvp1*PsendingNode1:(active1'=2)
                         + recvp1*(1-PsendingNode1):(send1'=0)&(active1'=0)
                         + (1 - recvp1):(send1'=0)&(active1'=1);

// sending
```

```
[tick] send1=1&active1=1 -> (send1'=0)&(active1'=0);

// time passing
[tick] active1=0 -> (send1'=0)&(active1'=0);
[tick] active1=2 ->(active1'=2);

// asynchronous receiving
[]     active1=2 ->(send1'=1)&(active1'=1);
endmodule
```

A.4 Model for the time

```
module time
        time: [0..TIME_MAX] init 0;

[tick] true -> (time'=min(time+1, TIME_MAX));
endmodule
```

A.5 Reward definitions

```
rewards "throughput"
  send3=1 & active3=1: 1;
endrewards

rewards "sent"
  true : send0*active0 + send1*active1 + send2*active2 + send3*active3;
endrewards

rewards "energy"
  // sending
  [tick] send0=1&active0=1 : 61.92;
  [tick] send1=1&active1=1 : 61.92;
  [tick] send2=1&active2=1 : 61.92;
  [tick] send3=1&active3=1 : 61.92;

  // receiving
```

```
  [tick] send0=0 & active0=1 : 44.4;
  [tick] send1=0 & active1=1 : 44.4;
  [tick] send2=0 & active2=1 : 44.4;
  [tick] send3=0 & active3=1 : 44.4;

  // off ("active mode")
  [tick] active0=0 : 3.84;
  [tick] active1=0 : 3.84;
  [tick] active2=0 : 3.84;
  [tick] active3=0 : 3.84;
  [] active0=2 : 3.84;
  [] active1=2 : 3.84;
  [] active2=2 : 3.84;
  [] active3=2 : 3.84;
endrewards

rewards "time"
  active3!=0: 1;
endrewards
```

Appendix B

PRISM model for the IEEE 802.15.4 contention resolution protocol

This appendix contains the source code of the PRISM models for the study on the *IEEE 802.15.4 contention resolution protocol*, which has been presented in Chapter 5.

B.1 Model of the unslotted mode

B.1.1 Parameter declarations

```
const int ACK_TIMEOUT = 6; // macAckWaitDuration
                           // = 120 [symbols] (at 20 and 40 ksymbols/s)
                           // or 54 (at 62.5 ksymbols/s)
const int ACK = 5; // 11 octets = 88 symbols (at 20 and 40 ksymbols/s)
                   // or 22 (at 62.5 ksymbols/s)
const int BACKOFF_PERIOD = 1; // aUnitBackoffPeriod = 20 [symbols]
const int CCA = 1; // 8 symbols
const int DATA_MAX = 54; // aMaxPHYPacketSize
                         // = 127 [octets] on MAC layer + 6 octets SHR+PHR
                         // = 1064 symbols (at 20 and 40 ksymbols/s)
                         // or 266 (at 62.5 ksymbols/s)
//const int DATA_MIN = 6; // 15 octets = 120 symbols (at 20 and 40 ksymbols/s)
```

```
                        // or 30 (at 62.5 ksymbols/s)
//const int OCTET = 1; // 1 octet = 8 symbols (at 20 and 40 ksymbols/s)
                       // or 2 (at 62.5 ksymbols/s)
                       // enable for nondeterministic data frame length only
const int TURNAROUND = 1; // aTurnaroundTime = 12 [symbols]
const int VULN = 1; // vulnerable period
    // VULN = AIRPROP + CCA + TURNAROUND = 8 + 12 + 0.002 = 20.002 symbols
    // AIRPROP: 0.1 microseconds, that is 0.002, 0.004, or 0.00625 symbols
    //          (at 20, 40, 62.5 ksymbols/s, respectively)
    // CCA: 1 analysis, starting on a backoff boundary (6.7.9)

// maximum constant used in timing constraints + 1
const int TIME_MAX = DATA_MAX+1;

// BACKOFF CONSTRAINTS
const int RETR_MAX = 3; // aMaxFrameRetries = 3
const int BE_MIN; // macMinBE = 0..3 (default 3)
const int BE_MAX = 5; // aMaxBE = 5
const int NB_MAX; // macCSMABackoffs = 0..5 (default 4)

// DATA FRAME TRANSMISSION TIMES
const int data1; // disable for nondeterministic data frame length only
const int data2; // disable for nondeterministic data frame length only

// OTHER CONSTANTS
//const int COL_MAX; // enable for collision probability properties only
```

B.1.2 Model for the time

```
module timer
        // global time
        t : [0..T];

        [time] (t<T) -> (t'=min(t+1,T));
        // loop when time passes deadline T to prevent further transitions
        [] (t>=T) -> (t'=t);

endmodule
```

B.1.3 Model of the channel, based on collision model

```
module channel

// number of collisions
//col : [0..COL_MAX]; // enable for collision probability properties only

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

[time] true -> true;

// begin sending message and nothing else currently being sent
[send1] c1=0 & c2=0 -> (c1'=1);
[send2] c2=0 & c1=0 -> (c2'=1);

// begin sending message and something is already being sent
// in this case both messages become garbled
[send1] c1=0 & c2>0 -> (c1'=2) & (c2'=2); // disable for collision probability
                                          // properties only
[send2] c2=0 & c1>0 -> (c1'=2) & (c2'=2); // disable for collision probability
                                          // properties only
//[send1] c1=0 & c2>0 -> (c1'=2) & (c2'=2) & col'=min(col+1,COL_MAX);
    // enable for collision probability properties only
//[send2] c2=0 & c1>0 -> (c1'=2) & (c2'=2) & col'=min(col+1,COL_MAX);
    // enable for collision probability properties only

// finish sending message
[finish1] c1>0 -> (c1'=0);
[finish2] c2>0 -> (c2'=0);

endmodule

// FORMULAE FOR THE CHANNEL
```

```
formula busy = c1>0 | c2>0;
formula free = c1=0 & c2=0;
```

B.1.4 Model of the channel, based on collision-free model

```
// begin sending message
[send1] c1=0 -> (c1'=1);
[send2] c2=0 -> (c2'=1);

// finish sending message
[finish1] c1>0 -> (c1'=0);
[finish2] c2>0 -> (c2'=0);

// FORMULAE FOR THE CHANNEL
formula busy = false;
formula free = true;
```

B.1.5 Model of the channel, based on additive interference

```
// begin sending message
[send1] c1=0 -> p_nocoll: (c1'=1) + (1-p_nocoll): (c1'=2);
[send2] c2=0 -> p_nocoll: (c2'=1) + (1-p_nocoll): (c2'=2);

// finish sending message
[finish1] c1>0 -> (c1'=0);
[finish2] c2>0 -> (c2'=0);

// FORMULAE FOR THE CHANNEL
formula busy = c1>0 | c2>0;
formula free = c1=0 & c2=0;
```

B.1.6 Model of the first node

```
module node1

// clocks for node 1
x1 : [0..TIME_MAX];
```

```
// local state
s1 : [1..9]; // disable for nondeterministic data frame length only
//s1 : [0..9]; // enable for nondeterministic data frame length only
// 0 set data frame length // for nondeterministic data frame length only
// 1 set backoff counter
// 2 backoff
// 3 vulnerable
// 4 failure
// 5 transmit
// 6 wait for acknowledgement
// 7 transmit acknowledgement
// 8 no acknowledgement received
// 9 done

// NUMBER OF RETRANSMISSIONS
retr1 : [0..RETR_MAX]; // disable for: noack, noackcol, noRETRlimit

// NUMBER OF BACKOFFS
nb1 : [0..NB_MAX]; // disable for: noBOlimit

// BACKOFF EXPONENT
be1 : [0..BE_MAX] init BE_MIN;

// BACKOFF COUNTER
backoff1 : [0..floor(func(pow,2,BE_MAX))-1];

// data frame length
//data1 : [DATA_MIN..DATA_MAX]; // enable for nondet. data frame length only

// SET DATA FRAME LENGTH
[] s1=0 & data1<DATA_MAX -> (data1'=min(data1+OCTET,DATA_MAX));
    // enable for nondeterministic data frame length only
[] s1=0 & data1>=DATA_MIN -> (s1'=1); // enable for nondeterministic data frame
                                      // length only

// SET BACKOFF COUNTER
// uniformly set backoff counter
```

```
// backoff exponent 0
[] s1=1 & be1=0 -> (s1'=2) & (x1'=0) & (backoff1'=0);
// backoff exponent 1
[] s1=1 & be1=1 -> 1/2 : (s1'=2) & (x1'=0) & (backoff1'=0)
                 + 1/2 : (s1'=2) & (x1'=0) & (backoff1'=1);
// backoff exponent 2
[] s1=1 & be1=2 -> 1/4 : (s1'=2) & (x1'=0) & (backoff1'=0)
                 + 1/4 : (s1'=2) & (x1'=0) & (backoff1'=1)
                 + 1/4 : (s1'=2) & (x1'=0) & (backoff1'=2)
                 + 1/4 : (s1'=2) & (x1'=0) & (backoff1'=3);
// backoff exponent 3
[] s1=1 & be1=3 -> 1/8 : (s1'=2) & (x1'=0) & (backoff1'=0)
                 + 1/8 : (s1'=2) & (x1'=0) & (backoff1'=1)
                 + 1/8 : (s1'=2) & (x1'=0) & (backoff1'=2)
                 + 1/8 : (s1'=2) & (x1'=0) & (backoff1'=3)
                 + 1/8 : (s1'=2) & (x1'=0) & (backoff1'=4)
                 + 1/8 : (s1'=2) & (x1'=0) & (backoff1'=5)
                 + 1/8 : (s1'=2) & (x1'=0) & (backoff1'=6)
                 + 1/8 : (s1'=2) & (x1'=0) & (backoff1'=7);
// backoff exponent 4
[] s1=1 & be1=4 -> 1/16 : (s1'=2) & (x1'=0) & (backoff1'=0)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=1)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=2)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=3)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=4)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=5)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=6)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=7)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=8)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=9)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=10)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=11)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=12)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=13)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=14)
                 + 1/16 : (s1'=2) & (x1'=0) & (backoff1'=15);
// backoff exponent 5
[] s1=1 & be1=5 -> 1/32 : (s1'=2) & (x1'=0) & (backoff1'=0)
```

```
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=1)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=2)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=3)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=4)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=5)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=6)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=7)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=8)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=9)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=10)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=11)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=12)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=13)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=14)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=15)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=16)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=17)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=18)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=19)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=20)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=21)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=22)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=23)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=24)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=25)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=26)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=27)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=28)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=29)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=30)
                    + 1/32 : (s1'=2) & (x1'=0) & (backoff1'=31);

// BACKOFF
// let time pass
[time] s1=2 & x1<BACKOFF_PERIOD & backoff1>0 -> (x1'=min(x1+1,TIME_MAX));
// decrement backoff
[] s1=2 & x1=BACKOFF_PERIOD & backoff1>0 -> (x1'=0) & (backoff1'=backoff1-1);
// finish backoff
```

```
[] s1=2 & backoff1=0 -> (s1'=3);

// VULNERABLE
// if channel is not detected busy, then let time pass
[time] s1=3 & x1<VULN & free -> (x1'=min(x1+1,TIME_MAX));
// if channel is detected busy, then backoff
[] s1=3 & busy & nb1=NB_MAX -> (s1'=4); // disable for: noBOlimit
[] s1=3 & busy & nb1<NB_MAX -> (s1'=1) & (x1'=0) & (nb1'=min(nb1+1,NB_MAX))
    & (be1'=min(be1+1,BE_MAX)); // disable for: noBOlimit
//[] s1=3 & busy -> (s1'=1) & (x1'=0) & (be1'=min(be1+1,BE_MAX));
    // enable for: noBOlimit
// move to transmit
[send1] s1=3 & x1=VULN -> (s1'=5) & (x1'=0);

// FAILURE
[] s1=4 -> true;

// TRANSMIT
// let time pass
[time] s1=5 & x1<data1 -> (x1'=min(x1+1,TIME_MAX));
// finish transmission successfully
[finish1] s1=5 & x1>=data1 & c1=1 -> (s1'=6) & (x1'=0); // disable for: noack
//[finish1] s1=5 & x1>=data1 -> (s1'=9); // enable for: noack
// finish transmission garbled
[finish1] s1=5 & x1>=data1 & c1=2 -> (s1'=8) & (x1'=0); // disable for: noack

// WAITING FOR ACKNOWLEDGEMENT
// let time pass
[time] s1=6 & x1<TURNAROUND -> (x1'=min(x1+1,TIME_MAX)); // disable for: noack
// turn from sending to receiving
[send1] s1=6 -> (s1'=7) & (x1'=0); // disable for: noack

// TRANSMIT ACKNOWLEDGEMENT
// let time pass
[time] s1=7 & x1<ACK -> (x1'=min(x1+1,TIME_MAX)); // disable for: noack
// acknowledgement sent successfully
[finish1] s1=7 & (x1=ACK | x1=ACK-1) & c1=1 -> (s1'=9); // disable for: noack,
                                                        // noackcol
```

```
//[finish1] s1=7 & (x1=ACK | x1=ACK-1) -> (s1'=9); // enable for: noackcol
// acknowledgement sent garbled (timer is not reset)
[finish1] s1=7 & (x1=ACK | x1=ACK-1) & c1=2 -> (s1'=8); // disable for: noack,
                                                        // noackcol

// NO ACKNOWLEDGEMENT
// let time pass
[time] s1=8 & x1<ACK_TIMEOUT -> (x1'=min(x1+1,TIME_MAX)); // disable for: noack
// no acknowledgement received after timeout
[] s1=8 & x1=ACK_TIMEOUT & retr1<RETR_MAX -> (s1'=1) & (x1'=0)
    & (retr1'=retr1+1) & (nb1'=0) & (be1'=BE_MIN);
    // disable for: noBOlimit, noack, noackcol, noRETRlimit
//[] s1=8 & x1=ACK_TIMEOUT & retr1<RETR_MAX -> (s1'=1) & (x1'=0)
    & (retr1'=retr1+1) & (be1'=BE_MIN);
    // enable for: noBOlimit; disable for: noBOlimit_noack, noBOlimit_noackcol,
    // noBOlimit_noRETRlimit
[] s1=8 & x1=ACK_TIMEOUT & retr1=RETR_MAX -> (s1'=4);
    // disable for: noack, noackcol, noRETRlimit
//[] s1=8 & x1=ACK_TIMEOUT -> (s1'=1) & (x1'=0) & (nb1'=0) & (be1'=BE_MIN);
    // enable for: noackcol, noRETRlimit; disable for: noBOlimit_noackcol and
    // noBOlimit_noRETRlimit
//[] s1=8 & x1=ACK_TIMEOUT -> (s1'=1) & (x1'=0) & (be1'=BE_MIN);
    // enable for: noBOlimit_noackcol, noBOlimit_noRETRlimit

// DONE
[time] s1=9 -> true;

endmodule
```

B.1.7 Model of an additional node

```
module

node2=node1[
  backoff1=backoff2,
  be1=be2,
  c1=c2,
  c2=c1,
```

```
  data1=data2,
  finish1=finish2,
  nb1=nb2, // disable for: noBOlimit
  retr1=retr2, // disable for: noack, noackcol, noRETRlimit
  s1=s2,
  send1=send2,
  x1=x2
]

endmodule
```

B.2 Model of the slotted mode

B.2.1 Parameter declarations

Based on parameter declarations for unslotted mode, only changes are listed.

```
// VARIABLE PARAMETERS
const int BO; // macBeaconOrder = 0..15 (has to be <15 for beacon-enabled PANs)
const int SO; // macSuperframeOrder = 0..15, SO <= BO (has to be <15 for
              // beacon-enabled PANs)

// SUPERFRAME CONSTRAINTS
const int A_SUPERFRAME_SLOT
    = floor(func(pow,2,SO)) * ASLOT; // 2^SO * aBaseSlotDuration
const int ASLOT = 3; // aBaseSlotDuration = 60 [symbols]
const int BI = floor(func(pow,2,BO)) * SUPERFRAME; // beacon interval
const int CAP_MIN = 22; // aMinCAPLength = 440 [symbols]
const int SUPERFRAME // aBaseSuperframeDuration
    = ASLOT * SUPERFRAME_SLOTS; // = aBaseSlotDuration * aNumSuperframeSlots
const int SUPERFRAME_SLOTS = 16; // aNumSuperframeSlots = 16
const int SD = floor(func(pow,2,SO)) * SUPERFRAME; // superframe duration

// TIMING CONSTRAINTS
//const int BEACON_MAX = 56; // BEACON_MAX+LIFS=100+40=140 octets
                             // = 1120 symbols (at 20 and 40 ksymbols/s)
                             // or 280 (at 62.5 ksymbols/s)
                             // enable for: nondet. CAP length; disable otherwise
```

```
const int BEACON_MIN = 14; // BEACON_MIN+SIFS=23+12=35 octets
                           // = 280 symbols (at 20 and 40 ksymbols/s)
                           // or 70 (at 62.5 ksymbols/s)
const int LIFS = 2; // aMinLIFSPeriod = 40 [symbols]
const int SIFS = 1; // aMinSIFSPeriod = 12 [symbols]
const int SIFS_FRAME_MAX = 10; // aMaxSIFSFrameSize
                               // = 18 [octets] on MAC layer + 6 octets SHR+PHR
                               // = 192 symbols (at 20 and 40 ksymbols/s)
                               // or 48 (at 62.5 ksymbols/s)
const int VULN = 2; // vulnerable period [heindl03performance, singh:performance]
    // VULN = AIRPROP + CCA + TURNAROUND = 20 + 8 + 12 + 0.002 = 40.002 symbols
    // AIRPROP: 0.1 microseconds, that is 0.002, 0.004, or 0.00625 symbols
    //          (at 20, 40, 62.5 ksymbols/s, respectively)
    // CCA: 2 analyses, each starting on a backoff boundary (6.7.9)

// end of CAP (that is, beacon length plus CAP length)
const int cap; // disable for: nondeterministic CAP length; enable otherwise
```

B.2.2 Model of the coordinator

```
// COORDINATOR FOR CSMA-CA SYNCHRONISATION
module coordinator

// clock for superframe
x0 : [0..BI];

// local state
s0 : [0..3];
// 0 transmit beacon
// 1 define CAP length
// 2 CAP
// 3 CFP and inactive superframe duration

// end of CAP (that is, beacon length plus CAP length)
//cap : [0..SD]; // enable for: nondeterministic CAP length; disable otherwise

// let time pass (transmit beacon)
```

```
[time] s0=0 & x0<BEACON_MIN & x0<SD -> (x0'=x0+1) & (s0'=0);
    // disable for: nondeterministic CAP length; enable otherwise
//[time] s0=0 & x0<BEACON_MAX & x0<SD -> (x0'=x0+1) & (s0'=0);
    // enable for: nondeterministic CAP length; disable otherwise
// finish beacon transmission
[] s0=0 & x0>=BEACON_MIN -> (s0'=2);
    // disable for: nondeterministic CAP length; enable otherwise
//[] s0=0 & x0>=BEACON_MIN -> (s0'=1);
    // enable for: nondeterministic CAP length; disable otherwise

// define CAP length
//[] s0=1 & cap<16*A_SUPERFRAME_SLOT & cap<=SD-A_SUPERFRAME_SLOT
    -> (cap'=cap+A_SUPERFRAME_SLOT);
    // enable for: nondeterministic CAP length; disable otherwise
//[] s0=1 & cap>=x0+CAP_MIN -> (s0'=2);
    // enable for: nondeterministic CAP length; disable otherwise
//[] s0=1 & x0+CAP_MIN>16*A_SUPERFRAME_SLOT -> (s0'=0);
    // enable for: nondeterministic CAP length; disable otherwise

// let time pass (during active superframe)
[time] s0=2 & x0<cap -> (x0'=x0+1);
// finish CAP
[] s0=2 & x0=cap -> (s0'=3);

// let time pass (during CFP and inactive superframe duration)
[time] s0=3 & x0<BI -> (x0'=x0+1);
// start new superframe
[] x0=BI -> (s0'=0) & (x0'=0);
    // disable for: nondeterministic CAP length; enable otherwise
// [] x0=BI -> (s0'=0) & (x0'=0) & (cap'=0);
    // enable for: nondeterministic CAP length; disable otherwise
endmodule
```

B.2.3 Model of node 1

Based on parameter declarations for unslotted mode, only changes are listed.

```
// BACKOFF
// let time pass
```

```
[time] s1=2 & s0=2 & x1<BACKOFF_PERIOD & backoff1>0 -> (x1'=min(x1+1,TIME_MAX));
// decrement backoff
[] s1=2 & x1=BACKOFF_PERIOD & backoff1>0 -> (x1'=0) & (backoff1'=backoff1-1);
// let time pass (remaining backoff postponed until next superframe)
[time] s1=2 & x1<BACKOFF_PERIOD & (s0=0 | s0=3) -> (x1'=0);
[time] s1=2 & backoff1=0 & s0=2 & data1<=SIFS_FRAME_MAX
    & x0+VULN+data1+ACK_TIMEOUT+SIFS>cap -> true;
[time] s1=2 & backoff1=0 & s0=2 & data1>SIFS_FRAME_MAX-1
    & x0+VULN+data1+ACK_TIMEOUT+LIFS>cap -> true;
// finish backoff
[] s1=2 & backoff1=0 & s0=2 & data1<=SIFS_FRAME_MAX
    & x0+VULN+data1+ACK_TIMEOUT+SIFS<=cap -> (s1'=3);
[] s1=2 & backoff1=0 & s0=2 & data1>SIFS_FRAME_MAX-1
    & x0+VULN+data1+ACK_TIMEOUT+LIFS<=cap -> (s1'=3);
```

B.3 Model of the slotted mode with battery life extension

```
// clock for battery life extension
x_ble: [0..6*BACKOFF_PERIOD];

// let time pass (transmit beacon)
[time] s0=0 & x0<BEACON_MIN & x0<SD -> (x0'=x0+1) & (s0'=0);
    // disable for: nondeterministic CAP length; enable otherwise
//[time] s0=0 & x0<BEACON_MAX & x0<SD -> (x0'=x0+1) & (s0'=0);
    // enable for: nondeterministic CAP length; disable otherwise
// finish beacon transmission
[] s0=0 & x0>=BEACON_MIN -> (x_ble'=0) & (s0'=2);
    // disable for: nondeterministic CAP length; enable otherwise
//[] s0=0 & x0>=BEACON_MIN -> (s0'=1);
    // enable for: nondeterministic CAP length; disable otherwise

// let time pass (during active superframe)
[time] s0=2 & x0<cap -> (x0'=x0& x_ble<6*BACKOFF_PERIOD
    -> (x0'=x0+1) & (x_ble'=x_ble+1);
// finish CAP
[] s0=2 & x0=cap(x0=cap | x_ble=6*BACKOFF_PERIOD) -> (s0'=3);
```

B.4 Reward definitions

```
// for expected time properties // enable for expected time properties only
rewards "time"
  [time] true : 1;
endrewards

// for expected number of collisions properties // enable for expected number of
// collisions properties only
rewards "collisions"
  [send1] c1=0 & c2>0 : 1;
  [send2] c2=0 & c1>0 : 1;
endrewards

// for expected energy properties // enable for expected energy properties only
rewards "energy"
  s1=1 : 0; // 1 set backoff counter
  [time] s1=2 : 1.536; // 2 backoff
  [send1] s1=3 : 23.424; // 3 vulnerable, CCA successful
  [send2] s1=3 : 23.424; // 3 vulnerable, CCA successful
  [] s1=3 & busy : 8.5632; // 3 vulnerable, CCA failed
  s1=4 : 0; // 4 failure
  [time] s1=5 : 24.768; // 5 transmit
  [send1] s1=6 : 16.3968; // 6 wait for acknowledgement
  [send2] s1=6 : 16.3968; // 6 wait for acknowledgement
  [finish1] s1=7 & c1=1 : 19.536 ; // 7 transmit acknowledgement
  [finish2] s1=7 & c1=2 : 19.536 ; // 7 transmit acknowledgement
  [] s1=8 & x1=ACK_TIMEOUT: 57.8016; // 8 no acknowledgement received
  [] s1=8 & x2=ACK_TIMEOUT: 57.8016; // 8 no acknowledgement received
  s1=9 : 0; // 9 done
endrewards
```

Appendix C

PRISM model for bandwidth-aware dynamic slot allocation protocols for low-rate wireless networks

This appendix contains the source code of the PRISM models for the study on *bandwidth-aware dynamic slot allocation protocols for low-rate wireless networks*, which has been presented in Chapter 6.

C.1 The environment

C.1.1 Model of the environment

```
module feeder
mode: [0..3] init 1;

[put1] mode=1 -> (mode'=2);
[put2] mode=2 -> (mode'=3);

[choose1] mode=3 & ownc=index1 -> (mode'=0);
[choose2] mode=3 & ownc=index2 -> (mode'=0);

[ack1] mode=0 & ownc=index1 -> (mode'=1);
[ack2] mode=0 & ownc=index2 -> (mode'=1);
```

```
[idle1] mode=0 & ownc=index1 -> (mode'=1);
[idle2] mode=0 & ownc=index2 -> (mode'=1);
[req1] mode=0 & ownc=index1 -> (mode'=1);
[req2] mode=0 & ownc=index2 -> (mode'=1);
[send1] mode=0 & ownc=index1 -> (mode'=1);
[send2] mode=0 & ownc=index2 -> (mode'=1);
endmodule
```

C.1.2 Declaration of environment parameters

```
const double UTILISATION1;
const double UTILISATION2;

const double P_PUT0_1 = (1-P_PUT1_1) * (UTILISATION1/2) / (1-UTILISATION1/2);
    // sending prob. to forwarder 1 if already sending
const double P_PUT0_2 = (1-P_PUT1_2) * (UTILISATION2/2) / (1-UTILISATION2/2);
    // sending prob. to forwarder 2 if not already sending
const double P_PUT1_1; // sending prob. to forwarder 1 if already sending
const double P_PUT1_2; // sending prob. to forwarder 2 if not already sending
```

C.2 The network

C.2.1 Declaration of forwarder parameters

```
const int ACK1;   // max. buffer occupation for forw. 1 to 'ack'
const int ACK2;   // max. buffer occupation for forw. 2 to 'ack'
const int BDIFF1; // min. diff. in buffer occupations of forw. 1 over 2 to 'req'
const int BDIFF2; // min. diff. in buffer occupations of forw. 2 over 1 to 'req'
const int BS1;    // buffer size of forw. 1
const int BS2;    // buffer size of forw. 2
const int REQ1;   // min. buffer occupation for forw. 1 to 'req'
const int REQ2;   // min. buffer occupation for forw. 2 to 'req'

const double P_SEND1; // prob. for forw. 1 to 'send' instead of 'ack' or 'req'
const double P_SEND2; // prob. for forw. 2 to 'send' instead of 'ack' or 'req'
const double P_REQ1;  // prob. for forw. 1 to 'req' instead of 'ack'
const double P_REQ2;  // prob. for forw. 2 to 'req' instead of 'ack'
```

C.2.2 Abstractions for the nodes

```
formula can_ack1  = r2>0 & buffer1<=ACK1 & (s1min+s2min+r2<=FS | s2min<FS-1);
formula can_ack2  = r1>0;
formula can_req1  = r1=0 & buffer1>=REQ1 & buffer1-buffer2>=BDIFF1;
formula can_send1 = buffer1>0;
```

C.2.3 Model of the first node

```
module forwarder1
buffer1: [0..BS1]; // buffer to store messages to be forwarded
dropped1: [0..1];  // message dropped in previous slot: 0...no, 1...yes
received1: [0..1]; // message received in prev. slot: 0...no, 1...yes
r1: [0..1];     // number of current requests
choice1: [0..3];   // 0...send, 1...req, 2...ack, 3...idle

// receiving a message and storing it in the buffer if not full
[put1] buffer1<BS1 & received1=0
    -> P_PUT0_1: (received1'=1) & (buffer1'=min(buffer1+1,BS1)) & (dropped1'=0)
    + (1-P_PUT0_1): (received1'=0) & (dropped1'=0);
[put1] buffer1=BS1 & received1=0
    -> P_PUT0_1: (received1'=1) & (dropped1'=1)
    + (1-P_PUT0_1): (received1'=0) & (dropped1'=0);
[put1] buffer1<BS1 & received1=1
    -> P_PUT1_1: (received1'=1) & (buffer1'=min(buffer1+1,BS1)) & (dropped1'=0)
    + (1-P_PUT1_1): (received1'=0) & (dropped1'=0);
[put1] buffer1=BS1 & received1=1
    -> P_PUT1_1: (received1'=1) & (dropped1'=1)
    + (1-P_PUT1_1): (received1'=0) & (dropped1'=0);

// resolving choices
[choose1] can_send1 & !can_req1 & !can_ack1 -> (choice1'=0);
[choose1] !can_send1 & can_req1 & !can_ack1 -> (choice1'=1);
[choose1] !can_send1 & !can_req1 & can_ack1 -> (choice1'=2);
[choose1] can_send1 & can_req1 & !can_ack1
    -> P_SEND1: (choice1'=0) + (1-P_SEND1): (choice1'=1);
[choose1] can_send1 & !can_req1 & can_ack1
    -> P_SEND1: (choice1'=0) + (1-P_SEND1): (choice1'=2);
[choose1] !can_send1 & can_req1 & can_ack1
```

```
    -> P_REQ1: (choice1'=1) + (1-P_REQ1): (choice1'=2);
[choose1] can_send1 & can_req1 & can_ack1
    -> P_SEND1: (choice1'=0) + (1-P_SEND1)*(P_REQ1): (choice1'=1)
    + (1-P_SEND1)*(1-P_REQ1): (choice1'=2);
[choose1] !can_send1 & !can_req1 & !can_ack1 -> (choice1'=3);

// forwarding a message and removing it from the buffer
// (assuming successful delivery)
[send1] choice1=0 -> (buffer1'=max(buffer1-1,0));

// requesting ownership of a slot
//   aggressive strategy: always send request if there are at least REQ
//   messages in this node's buffer and it does not control all slots
[req1] choice1=1 -> (r1'=1);

// sending an ack (during this node's slot)
//   hesitant strategy: only send acknowledgement if there are at most ACK
//   messages in this node's buffer
[ack1] choice1=2 -> true;

// letting time pass if no other action is possible
[idle1] choice1=3 -> true;

// waiting to receive an ack (during the other node's slot)
[ack2] can_ack2 -> (r1'=0);
endmodule
```

C.2.4 Model of an additional node

```
module forwarder2=forwarder1[
  ack1=ack2,
  ack2=ack1,
  ACK1=ACK2,
  ACK2=ACK1,
  BDIFF1=BDIFF2,
  BS1=BS2,
  buffer1=buffer2,
  buffer2=buffer1,
```

```
  choice1=choice2,
  choose1=choose2,
  dropped1=dropped2,
  idle1=idle2,
  index1=index2,
  index2=index1,
  P_PUT0_1=P_PUT0_2,
  P_PUT1_1=P_PUT1_2,
  P_REQ1=P_REQ2,
  P_SEND1=P_SEND2,
  put1=put2,
  r1=r2,
  r2=r1,
  received1=received2,
  s1=s2,
  s2=s1,
  s1min=s2min,
  s2min=s1min,
  req1=req2,
  REQ1=REQ2,
  send1=send2
]
endmodule
```

C.2.5 Model for resolving choices, based on additive interference

```
[choose1] can_send1 & !can_req1 & !can_ack1 -> p_nocoll: (choice1'=0)
                                             + (1-p_nocoll): (choice1'=3);
[choose1] !can_send1 & can_req1 & !can_ack1 -> (choice1'=1);
[choose1] !can_send1 & !can_req1 & can_ack1 -> (choice1'=2);
[choose1] can_send1 & can_req1 & !can_ack1 -> P_SEND1*p_nocoll: (choice1'=0)
                                            + (1-P_SEND1): (choice1'=1)
                                            + P_SEND1*(1-p_nocoll): (choice1'=3);
[choose1] can_send1 & !can_req1 & can_ack1 -> P_SEND1*p_nocoll: (choice1'=0)
                                            + (1-P_SEND1): (choice1'=2)
                                            + P_SEND1*(1-p_nocoll): (choice1'=3);
[choose1] !can_send1 & can_req1 & can_ack1 -> P_REQ1: (choice1'=1)
                                            + (1-P_REQ1): (choice1'=2);
```

```
[choose1] can_send1 & can_req1 & can_ack1 -> P_SEND1*p_nocoll: (choice1'=0)
                                          + (1-P_SEND1)*(P_REQ1): (choice1'=1)
                                          + (1-P_SEND1)*(1-P_REQ1): (choice1'=2)
                                          + P_SEND1*(1-p_nocoll): (choice1'=3);
[choose1] !can_send1 & !can_req1 & !can_ack1 -> (choice1'=3);
```

C.3 The protocol

C.3.1 Declaration of protocol parameters

```
const int FS; // frame size, i.e. number of slots per a frame
```

C.3.2 Abstractions for the protocol

```
formula last_slot = c=FS;
formula next1_ack2 = s1<min(s1min+buffer1,FS-s2min);
formula next2_ack1 = s2<min(s2min+buffer2,FS-s1min);
formula next1 = s1<s1min;
formula next2 = s2<s2min;
formula full_ack1 = s1min+s2min+buffer2<=FS;
formula full_ack2 = s1min+s2min+buffer1<=FS;
formula part_ack1 = s2min<FS-1;
formula part_ack2 = s1min<FS-1;
```

C.3.3 Model of the protocol

```
module slotCounter
// forwarder1 always owns slot 1 mod FS, forwarder2 always owns slot 2 mod FS,
// and the other slots are available for reallocation

c: [1..FS]; // index of the current slot (w.r.t. the current period)
ownc: [1..2]; // owner of the current slot
s1: [1..FS-1]; // number of slots in current frame used by forwarder 1
s2: [0..FS-1]; // number of slots in current frame used by forwarder 2
s1min: [1..FS-1] init ceil(FS/2); // minimal number of slots in the next frame
                                  // to be allocated to forwarder 1
```

```
s2min: [1..FS-1] init floor(FS/2); // minimal number of slots in the next frame
                                   // to be allocated to forwarder 2

// time passing
[ack1] !last_slot & next2_ack1 & full_ack1
    -> (c'=c+1) & (ownc'=2) & (s2'=s2+1) & (s2min'=min(s2min+buffer2,FS-s1min));
[ack1] !last_slot & next2_ack1 & !full_ack1 & part_ack1
    -> (c'=c+1) & (ownc'=2) & (s2'=s2+1) & (s1min'=max(s1min-buffer2,1))
       & (s2min'=min(s2min+buffer2,FS-1));
[ack1] !last_slot & !next2_ack1 & next1 & full_ack1
    -> (c'=c+1) & (ownc'=1) & (s1'=s1+1) & (s2min'=min(s2min+buffer2,FS-s1min));
[ack1] !last_slot & !next2_ack1 & next1 & !full_ack1 & part_ack1
    -> (c'=c+1) & (ownc'=1) & (s1'=s1+1) & (s1min'=max(s1min-buffer2,1))
       & (s2min'=min(s2min+buffer2,FS-1));
[ack1] last_slot & full_ack1
    -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0)
       & (s2min'=min(s2min+buffer2,FS-s1min));
[ack1] last_slot & !full_ack1 & part_ack1
    -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0) & (s1min'=max(s1min-buffer2,1))
       & (s2min'=min(s2min+buffer2,FS-1));

[ack2] !last_slot & next1_ack2 & full_ack2
    -> (c'=c+1) & (ownc'=1) & (s1'=s1+1) & (s1min'=min(s1min+buffer1,FS-s2min));
[ack2] !last_slot & next1_ack2 & !full_ack2 & part_ack2
    -> (c'=c+1) & (ownc'=1) & (s1'=s1+1) & (s2min'=max(s2min-buffer1,1))
       & (s1min'=min(s1min+buffer1,FS-1));
[ack2] !last_slot & !next1_ack2 & next2 & full_ack2
    -> (c'=c+1) & (ownc'=2) & (s2'=s2+1) & (s1min'=min(s1min+buffer1,FS-s2min));
[ack2] !last_slot & !next1_ack2 & next2 & !full_ack2 & part_ack2
    -> (c'=c+1) & (ownc'=2) & (s2'=s2+1) & (s2min'=max(s2min-buffer1,1))
       & (s1min'=min(s1min+buffer1,FS-1));
[ack2] last_slot & full_ack2
    -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0)
       & (s1min'=min(s1min+buffer1,FS-s2min));
[ack2] last_slot & !full_ack2 & part_ack2
    -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0) & (s2min'=max(s2min-buffer1,1))
       & (s1min'=min(s1min+buffer1,FS-1));
```

```
[idle1] !last_slot & next2 -> (c'=c+1) & (ownc'=2) & (s2'=s2+1);
[idle1] !last_slot & !next2 & next1 -> (c'=c+1) & (ownc'=1) & (s1'=s1+1);
[idle1] last_slot -> (c'=1) & (ownc'=2) & (s1'=1) & (s2'=0);
[idle2] !last_slot & next1 -> (c'=c+1) & (ownc'=1) & (s1'=s1+1);
[idle2] !last_slot & !next1 & next2 -> (c'=c+1) & (ownc'=2) & (s2'=s2+1);
[idle2] last_slot -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0);

[req1] !last_slot & next2 -> (c'=c+1) & (ownc'=2) & (s2'=s2+1);
[req1] !last_slot & !next2 & next1 -> (c'=c+1) & (ownc'=1) & (s1'=s1+1);
[req1] last_slot -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0);

[req2] !last_slot & next1 -> (c'=c+1) & (ownc'=1) & (s1'=s1+1);
[req2] !last_slot & !next1 & next2 -> (c'=c+1) & (ownc'=2) & (s2'=s2+1);
[req2] last_slot -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0);

[send1] !last_slot & next2 -> (c'=c+1) & (ownc'=2) & (s2'=s2+1);
[send1] !last_slot & !next2 & next1 -> (c'=c+1) & (ownc'=1) & (s1'=s1+1);
[send1] last_slot -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0);

[send2] !last_slot & next1 -> (c'=c+1) & (ownc'=1) & (s1'=s1+1);
[send2] !last_slot & !next1 & next2 -> (c'=c+1) & (ownc'=2) & (s2'=s2+1);
[send2] last_slot -> (c'=1) & (ownc'=1) & (s1'=1) & (s2'=0);
endmodule
```

C.4 Reward definitions

```
rewards "throughput"
  [send1] true : 4;
  [send2] true : 4;
endrewards

rewards "throughput1"
  [send1] true : 4;
endrewards

rewards "loss"
  [put1] true : 4*dropped1;
```

```
  [put2] true : 4*dropped2;
endrewards

rewards "buffer"
  [put1] true : 4*buffer1 + 4*buffer2;
endrewards

rewards "idle"
  [idle] true : 4;
endrewards

rewards "acknowledged"
  [ack1] true : 4;
  [ack2] true : 4;
endrewards

rewards "reallocated"
  [ack1] true : buffer2;
  [ack2] true : buffer1;
endrewards

rewards "traffic"
  [put1] true : 4*received1;
  [put2] true : 4*received2;
endrewards

rewards "energy"
  [put1] true : 77.4;
  [put2] true : 77.4;
  [ack1] true : 132.9;
  [ack2] true : 132.9;
  [idle1] true : 9.6;
  [idle2] true : 9.6;
  [req1] true : 132.9;
  [req2] true : 132.9;
  [send1] true : 82.2;
  [send2] true : 82.2;
  [choose1] true : 0;
```

```
  [choose2] true : 0;
endrewards
```