Project no.:	**PIRSES-GA-2011-295261**
Project full title:	**Mobility between Europe and Argentina applying Logics to Systems**
Project Acronym:	**MEALS**
Deliverable no.:	**3.4 / 2**
Title of Deliverable:	**Rewards Estimation for Partial Explorations of Nondeterministic Stochastic Systems**

Contractual Date of Delivery to the CEC:	**30-Sep-2014**
Actual Date of Delivery to the CEC:	**15-Sep-2014**
Organisation name of lead contractor for this deliverable:	**INR**
Author(s):	**Esteban Pavese, Victor Braberman, Sebastian Uchitel**
Participants(s):	**UBA, IMP**
Work package contributing to the deliverable:	**WP3**
Nature:	**R**
Dissemination Level:	**Public**
Total number of pages:	**18**
Start date of project:	1 Oct. 2011 **Duration:** 48 month

Abstract:

Estimation of quantitative properties of software models can provide useful insights early in the development process. However, computational complexity of estimating reliability or service-level metrics such as mean time to first failure (MTTF) or turnaround time (TAT) can be prohibitive both in time, space and precision. A promising approach to tackling this complexity is to compute lower bounds to these measures, based on partial explorations of the system under analysis: *(i)* Model simulation to obtain samples of system behaviour; *(ii)* Invariant inference from samples; and *(iii)* exhaustive model checking of the submodel defined by the invariant. However, existing work on quantitative analysis using partial explorations is limited to deterministic models (which cannot be easily simulated) and MTTF properties.
In this paper, we extend existing work to nondeterministic models and general properties on reward structures, by performing a nondeterministic choice resolution, and show experiments that suggest that quantitative estimation using this technique can be more effective than (full model) probabilistic model checking.

This project has received funding from the European Union Seventh Framework Programme (FP7 2007-2013) under Grant Agreement Nr. 295261.

Contents

1 Introduction

Model-based automated verification for assessing quantitative properties of software systems aims to provide insights early in the development process that can reduce significantly both development costs, and costs associated with deploying software that does not conform to its service-level requirements. However, traditional metrics for these properties require models that support describing probabilistic, non-deterministic and timed behaviour; models for which estimating such metrics can be prohibitive in time, space and precision.

Model checking is an effective software verification method. In the context of quantitative guarantees, such results can be computed for complex models using the techniques developed in the area known as probabilistic model checking [2,15]. These techniques take as input probabilistic models (such as Markov Decision Processes) and can assess quantitative properties through exhaustive exploration of the model state space and subsequent numerical analysis.

Applicability of probabilistic model checking is threatened by the size of the model to be analysed. Although state space reduction techniques exist, they may still fail to prevent state explosion. Further, even if the entire state space can be explored, its size typically impedes exact numerical calculation (e.g. Gaussian elimination) of these metrics, so iterative methods (such as Jacobi or Gauss-Seidel) that approximate metrics must be used. However, some of these methods do not have convergence guarantees, and when they do converge they may do so (intractably) slowly. The latter can be a problem, for example, for reliability metrics of models with unlikely failures (e.g. probability of failure in a fixed period below 10^{-5}), and can lead to iterations being cut short far from the actual value of the metric being estimated.

In summary, although probabilistic model checking may seem to promise exact calculation of quantitatively measured properties, state space explosion and numerical methods can be computationally prohibitive or result in poor approximations. Despite these limitations, probabilistic model checking can provide bounds with 100% confidence for quantitative metrics even though the distance of these bounds to the real value cannot be known.

In this paper we expand on the alternative to exhaustive model exploration –as in probabilistic model checking– and partial random exploration –as in statistical model checking– presented in [13], which has shown to counter some of the limitations of existing model-based quantitative verification techniques. More specifically, in this paper we expand the referenced technique for reward estimation over nondeterministic stochastic systems. We implement an analysis technique for Markov Decision Processes (MDP). We use simulation to produce a set of traces that represent likely behaviour of the full model using a naive resolution strategy for non-deterministic choice: we replace non-determinism with a probabilistic choice that makes each choice equiprobable. These traces are used to infer invariants of the state space explored during the simulation. The resulting invariant represents a submodel of the MDP which contains all states that satisfy the invariant and all probabilistic and nondeterministic choices between these states. We then model check this submodel for quantitative properties. Our choice of strategy for resolving the problem of simulation for nondeterminism, although arguably arbitrary and simplistic, can be shown to not affect the correctness of the estimation technique. We present results that support that the technique can be applied effectively to nondeterministic stochastic systems.

The remainder of the paper is organized as follows. In section 2 we provide background on reward estimation. In Section 3 we describe our approach to reward estimation and nondeterminism resolution. Section 4 provides case studies illustrating the approach and comparing results to existing techniques. In Section 5 we present a discussion of our results and of related work. Finally we offer our conclusions and discuss future work in Section 6.

2 Background

Quantitative measures such as *mean time to first failure (MTTF)* and *turnaround time (TAT)* [12] are widely accepted metrics for reliability and service-level agreements in software systems. These metrics establish the time a client can expect to operate a software system until it experiences its first failure (MTTF) or the expected time until a computation or the response to a request takes place (TAT). In order to calculate such measures, practitioners base their efforts on *failure models* [12], which describe conditions under which the component is known to fail, or the time it takes to perform certain actions. Most often, these conditions are probabilistic in nature. This behaviour is usually modelled with a Markov chain. However, information regarding the probability of these conditions is usually partial. For example, the rate at which a packet is dropped in a network system might be known, but on the other hand the order in which stations along the network generate requests might not be known. In the first case, it might be easy to specify a probabilistic distribution describing the dropped packet phenomenon. In the second case, however, such information is not easily available and practitioners must resort to underspecify this order, resulting in nondeterministic models. *Markov Decision Processes* are models that can specify both these probabilistic and nondeterministic choices under a same formalism. Complex MDPs can be built compositionally using parallel composition [14] to model components that run asynchronously but synchronise on shared actions.

Definition 1. Markov Decision Process [3]: A *Markov Decision Process* (MDP) is a tuple $\langle S, s_0, A, R\rangle$ where $S \subseteq V \rightarrow C$ is a finite set of states, defined by mapping a finite set of variables V to values on a finite subset of $\mathbb{Z}$, C. $s_0 \in S$ is the initial state. A is a finite set of action labels. The finite relation $R \subseteq S \times (A \cup \{\tau\}) \times \mathcal{D}(S)$ is the transition relation where the transition target is defined by a distribution on target states.

A Discrete Time Markov Chain (DTMC) is similar to an MDP, but they do not allow nondeterminism. That is, R is such that for any state $s \in S$ there exists only one transition (s, a, δ).

pCTL [1], a probabilistic extension of CTL, is widely used to describe MDP model properties. pCTL formulae differ from CTL in that, instead of predicating about properties that may hold globally or for some execution paths, they aim at quantifying the probability of witnessing traces that satisfy a given property.

Besides using logics such as pCTL for reasoning about MDP models, we are also interested in conveying some sense of *value* to different MDP traces. As traces will possibly convey different values, we might be interested in knowing what the *expected value* of a trace will be, given a typical execution. Reward structures are used to specify this notion of value to MDP traces. For example, a transition reward structure that assigns a value of 1 to each transition is a standard way of defining overall time steps cost for the traces of an MDP.

Definition 2. Reward Structures: Given an MDP $M = \langle S, s_0, A, R\rangle$, a *transition reward structure* is a function $\rho : S \times A \times \mathcal{D}(S) \times S \rightarrow \mathbb{R}_{\geq 0}$.

The behaviour of an MDP is defined by the set of its generated *paths*. Paths are infinite chains $s_i a_{i+1} s_{i+1} a_{i+2} \ldots$ that start with a state s_i. Each state is followed by an action a and a target state, in such a way that this target state has a positive probability under one of the source state's distributions governed by the action a. We will note *paths*(M) the set of paths for an MDP M, and *paths* $*\,(M)$ the set of all finite prefixes of *paths*(M). In order to be able to reason about the probability of these paths, it is necessary to introduce the notion of a *scheduler* (also called *adversary*). The combination of an MDP and a scheduler yields a purely stochastic Markov chain, i.e., a DTMC.

Definition 3. Scheduler: A *scheduler* for an MDP M is a total function $S\ :\ paths * (M) \rightarrow R$, such that if $S(\alpha) = (a, s)$ it must be that (a, s) is a possible transition from the last state in α.

When analysing MDPs, it is not possible in general to assign a single probability to the satisfiability of pCTL formulae, but rather a set of probabilities (one for each scheduler). In general, we are interested in knowing the minimum and maximum values in this set of probabilities. A similar phenomenon happens for reward structures. The reward associated with the paths satisfying a certain formula will also be dependent on the scheduler selected.

3 Approach

This section formally defines an approach to computing bounds to reward structures of nondeterministic and stochastic software models. The approach is based on the calculation of said rewards for a partial systematic exploration of the model's state space. We first define what is meant by a partial exploration and show that expected rewards computed over these partial explorations are guaranteed bounds to the expected rewards of the entire system model. We then show that some partial explorations can be specified declaratively through invariant properties. Finally, we discuss how these declaratively specified partial explorations can be obtained automatically from any given model, without need for human intervention.

3.1 Partial Explorations

We refer to a partial exploration of a system model as a submodel. We recall the definition of submodel from [13], which can be adapted straightforwardly to MDPs. Intuitively, a submodel of an MDP M is a model that retains a subset of the states and transitions of M – including and reachable from the initial state – and in which all other states in M have been abstracted away into a new λ trap state.

Submodels are key to our approach since they conservatively approximate the value of reward structures for their maximum and minimum limit values. In other words, the *minimum* and *maximum* expected reward associated to a CTL property ϕ on a model M is *at least as much* as the *minimum* and *maximum* reward associated to the same property on any of its submodels. In the following, the notation $\overline{X}$ for a set X will denote the sample mean of the set X.

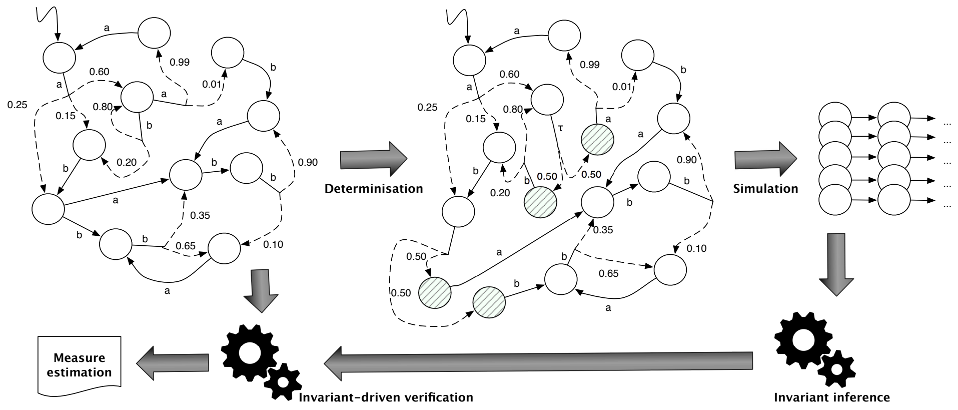

Figure 1: Example partial exploration of a state space

Theorem 1. *Let M and M' be two MDPs with state spaces S and S' and such that M' is a submodel of M. Then, for a given CTL formula ϕ and reward structure ρ, it holds that* $\max_{\sigma \in Sched(M)} \overline{\rho\{\pi \cdot \pi \models \phi\}} \geq \max_{\sigma' \in Sched(M')} \overline{\rho\{\pi' \cdot \pi' \models \phi\}}$*, and, likewise,* $\min_{\sigma \in Sched(M)} \overline{\rho\{\pi \cdot \pi \models \phi\}} \geq \min_{\sigma' \in Sched(M')} \overline{\rho\{\pi' \cdot \pi' \models \phi\}}$.

Proof. The intuition is that, for every path in the complete model, it either exists completely in the submodel, or the submodel contains only a prefix that is extended by the λ state. Since reward structures are based on transitions, every path in the full model accumulates *at least* as much reward (possibly ∞) as the corresponding path (or prefix) in the submodel. Hence these prefixes contribute to $\min / \max_{\sigma' \in Sched(M')} \overline{\rho\{\pi \cdot \pi \models \phi\}}$ at most as much as what their extensions in M contribute to $\min / \max_{\sigma \in Sched(M)} \overline{\rho\{\pi \cdot \pi \models \phi\}}$ □

The above result entails that if computing the extremum value of a reward structure of a system model is intractable, it can be conservatively approximated on any of its submodels.

3.2 Automatic submodel generation

Although any submodel will provide a lower bound for the reward structures, the key to a tractable quantitative measure estimation technique is to identify submodels for which this calculation can be computed within a reasonable time budget, and for which the resulting bound is a reasonable approximation to the reward of the full model. Independently of the fact that the reward value for the full model is unknown, this is a problem for which coming up with an exact solution (i.e. the “best” submodel) is intractable [11]. In [13] we detailed a heuristic approach to a submodel construction strategy that aims at building *probabilistically dense* submodels, that is, submodels for which the probability of reaching the λ trap state in a fixed time is low. We recall that approach here for convenience, but the interested reader is referred to [13] for details.

Our suggested approach attempts to approximate these probabilistically dense submodels through the use of bounded simulation. The resulting set of finite paths, if sufficiently large and consisting of sufficiently long paths, is likely to cover a good part of a probabilistically dense submodel. These traces form the basis for building our submodels. We observed that the behavioural information conveyed by these paths is concurrent with the behaviour most likely to be exhibited by the system in an arbitrary execution. As such, we aim at capturing the semantic information of these paths to build submodels. Our approach involves computing *state invariants* based on the states visited during the simulation and then add to the submodel any states and transitions that satisfy the invariant. Further, our approach aims at automatically obtaining these invariants. To this end, we produce probabilistically driven walks over the full system model, recording the states (i.e. variable valuations) traversed. We use Daikon [6], an invariant inference engine, to obtain predicates that hold over all traversed states. These invariant predicates, in turn, are used to synthesise an observer MDP that can drive the generation of a submodel via parallel composition.

It is important to note that, in our present approach, it is necessary to resolve nondeterministic transitions during the probabilistically driven walk generation. For the work presented in this paper, we have opted to resolve nondeterminism in a naïve way, simply by assuming an equiprobable distribution between nondeterministic transitions. Therefore, the first step of the approach is to perform simulation over an *equiprobably determinised* version of the original MDP. Once the invariant is inferred simulations, it is be used to generate the partial submodel of the *original* MDP. Figure 1 depicts this approach.

Definition 4. Equiprobably Determinised MDP: Let $M = \langle S, s_0, A, R\rangle$ be an MDP. The *equiprobably determinised MDP* of M is a DTMC $M_{det} = \langle S_{det}, s_0, A, R_{det}\rangle$ constructed in such a way that $S \subseteq S_{det}$, and for every $(s, a, \delta) \in R$:

- If (s, a, δ) is the only transition for s in M, add the transition to R_{det};
- otherwise, take all (s, a_i, δ_i). Add i states $t^s_1, \ldots, t^s_i$ to S_{det}. Add a transition (s, τ, δ) to R_{det} where $\delta(t^s_j) = 1/i$ for those added states, and 0 everywhere else. Finally, add transitions (t^s_i, a_i, δ_i) to R_{det} for each of the added states.

The correctness of our approach is not hampered by this choice, as in fact any method of resolving nondeterminism would serve our needs – any nondeterminism resolution approach yields a valid submodel. However, it is left to be studied if this is the best way to resolve nondeterminism. We discuss on this decision and possible alternatives in Section 5.

4 Validation

In this section we set out to validate our approach. The question we study is whether our approach can, when compared to probabilistic model checking over full explorations, produce higher bounds, in less time, for expected rewards of system models. Note that comparison with current Monte Carlo based techniques would be unfair, as they currently lack the capabilities to perform simulation while resolving nondeterminism. Works such as [8] take a step in this direction. However, it must be noted that they are geared towards probabilistic model checking (i.e., expecting a quantitative answer in the range $[0, 1]$), rather than estimation of rewards, where the results are unbounded. It is unclear whether these approaches are directly applicable to reward estimation, we discuss this further in Section 5.

4.1 Methodology

For both of the case studies taken from the literature, we took an MDP model (in one case the MDP existed, in the other it was reverse engineered), modelled interesting behaviour as state formulae, and defined an appropriate reward structure. We used the same input for both our approach as well as classical estimation techniques.

We ran our approach for both case studies for several automatically generated invariants varying the number and length of traces used for invariant inference. We used Daikon v4.6.4 [6] configured to produce invariants that are conjunctions of terms of the form $x \sim y$, where x and y are either variables in the model, or integer constants, and $\sim \in \{<, \leq, =, \geq >\}$. The invariants we obtained were used to automatically build an *observer* O, an MDP that monitors the validity of the invariant. This observer, when composed with the system model M, synchronises with all actions and forces transitioning into the λ trap state whenever the destination state of the intended transition would result in an invariant violation.

We used a modified version of PRISM v4.0.3 [9] to perform probabilistic model checking to estimate expected reward both for the full state space and for its invariant-driven submodels. Modifications allow for batch trace generation (used for invariant inference) and time and memory-use tracking (used for generating intermediate results and for timing out when time budget is up). In practice, no equiprobably determinised version of the MDP is generated, the nondeterministic choice is resolved on-the-fly in an equiprobable manner. PRISM was deployed on an 8x Core Intel Xeon CPU @1.60 GHz with 8GB RAM.

PRISM provides different numerical methods for reward calculation. Previous experimentation showed that it is usually the case that the Gauss-Seidel method with backwards propagation provides the best result in terms of bounds obtained for same time budgets, so we chose to run our experiments under that setting.

PRISM runs were considered complete when any of the following criteria held: either *a)* the *absolute* difference between results of successive iterations of the numerical method was less than 0.01 (relative differences are not adequate because of slow convergence, which causes iterative methods to cut too early). Alternatively, *b)* running time reached 3 hours; or *c)* memory was exhausted.

4.2 Case studies

Bounded Retransmission Protocol.

Our first case study [5] models a robust communication protocols that attempts to ensure delivery of data, the bounded retransmission protocol (BRP) [7].

BRP is a variant of the alternating bit protocol, which allows for a bounded number of retransmissions of a given chunk (i.e., a part of a file). The protocol consists of a sender, a receiver, and two lossy channels, used for data and acknowledgements respectively. The sender transmits a file composed of a number of chunks, by way of *frames*. Each frame contains the chunk itself and three bits. The first bit indicates whether the chunk is the first one; the second one if it is the last chunk; and the third bit alternates to avoid data duplication.

The sender waits for acknowledgement of each frame sent. The sender may timeout if either the frame or the corresponding acknowledgement are dropped. When this happens, the sender resends the frame and does so repeatedly up to a specified retry limit. If the limit is reached and the transmission is terminated, the sender may be able to establish that the file was not sent (if some chunks were left unsent) or it may not know the outcome (if the last frame was sent but no acknowledgement was received). In any case, the sender may send a new file, resetting the retry count. A maximum of 256 retransmissions are attempted per file before the sender aborts transmission of the file. Once a file is sent successfully or its transmission fails, the system waits for another file to be sent.

Protocol clients send files one at a time. Each of these files is of a different size (in number of chunks). The size is selected in a nondeterministic fashion for each file, between just a few and 1500 chunks, modelling the fact that we do not have certain information regarding this aspect of usage.

We wish to estimate the mean time to the first failure, where failure is defined as the sender failing to send a complete file (*incomplete*) or not being able to establish if a file was sent successfully (*unknown*). Consequently, the state predicate describing failures is $\mathit{incomplete} \lor \mathit{unknown}$. The definition of time for this case study aims at establishing how many data packets can be expected to be sent successfully before failure.

IEEE 802.11 Wireless LAN.

The second case study depicts the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism of the IEEE 802.11 protocol [10]. The protocol uses a randomised exponential backoff rule to minimise the likelihood of transmission collision. That is, whenever a collision was averted by a component sensing the busy carrier when trying to send data over busy media, the component is backed off (it needs to wait until trying to resend) for a time. This time is chosen randomly from a specified range of delay, and successive failures cause this range to increase exponentially. The goal of the protocol is to divide, as equally as possible, the access to the channel between all participants that may collide.

The model used depicts a two-way handshake mechanism of the IEEE 802.11 medium access control scheme, operating in a fixed network topology. The probabilistic model itself was lifted from [8]. It has both stochastic behaviour (for example, in the randomised backoff procedure, that allows up to seven exponential backoff levels) and nondeterministic behaviour (for example, in modelling the interleaving of actions between the two independent emitter stations).

We are interested in estimating the turnaround time for the two stations to be able to successfully send their packets and advance to their *done* state, while avoiding potential collisions. As such, the state predicate that describes this final state is $\mathit{station}1 = \mathit{done} \land \mathit{station}2 = \mathit{done}$. Note that, unlike the previous case study, both stations managing to send their messages is not a rare event at all. However, the sheer size of the model does hamper direct estimation.

Simulation for invariant inference and submodel generation			Model checking			
Traces	Length	States	Min. MTTF	Time	Max. MTTF	Time
7000	7000	362818	7012.95	289.40s	599588.06	TO
7000	8000	377758	8009.93	207.56s	449767.59	TO
7000	9000	392786	437502.87	TO	449818.72	TO
7000	10000	392786	356057.11	TO	487287.50	TO
8000	7000	362818	7012.95	216.32s	412304.48	TO
8000	8000	377758	8009.93	400.32s	385532.89	TO
8000	9000	392786	424594.70	TO	599678.92	TO
8000	10000	393127	509826.12	TO	712182.59	TO
9000	7000	362818	7012.95	150.81s	799269.07	TO
9000	8000	378099	8009.97	187.61s	749648.41	TO
9000	9000	392786	633351.44	TO	599678.92	TO
9000	10000	392786	612614.19	TO	649623.47	TO
10000	7000	362818	7012.95	163.30s	711919.82	TO
10000	8000	377758	8009.93	175.83s	699439.94	TO
10000	9000	392786	577921.47	TO	786948.15	TO
10000	10000	392786	761073.07	TO	899279.83	TO

Table 1: Selection of BRP submodel MTTF evaluation results for different simulation parameter configurations.

4.3 Experimental results

We now present some of the experimental results obtained for the research question presented above. The models used and complete experimental results can be found at `http://lafhis.dc.uba.ar/~epave`

When comparing probabilistic model checking of both full and partial models we are interested in considering the impact between the inferred invariant, the resulting submodel's size and the value of the reward estimation obtained from it. We are also interested in gaining insight on combinations of trace length and number that are likely to yield the best overall result.

BRP mean time to failure estimation.

In this case study, we first attempted to obtain the MTTF extreme values for the full model. After the 3 hours of allotted time elapsed for each extreme value estimation, the results yielded a model comprising nearly 29 million states, while the reward estimation set a minimum MTTF value of 8444 and a maximum MTTF of 9468. Both results were obtained after the estimation timed out at 3 hours. It must be noted that in this case, the notion of *time* is not that of calendar time but is measured in terms of actions taken by the protocol. The reason for this definition of time is that the protocol is highly reactive. There may be long idle times during which a client does not try to send any data, and it would be meaningless to compare that time with time incurred in actually carrying out actions within the protocol.

We then put our approach to work, with the results obtained depicted in Tables 1 and 2. Cases that computation did not converge before the 3 hours are reported as timeout (TO).

Traces	Length	Invariant
7000	7000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1167 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
7000	8000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1333 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
7000	9000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
7000	10000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
8000	7000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1167 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
8000	8000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1333 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
8000	9000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
8000	10000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
9000	7000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1167 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
9000	8000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1333 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
9000	9000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
9000	10000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
10000	7000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1167 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
10000	8000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1333 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
10000	9000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$
10000	10000	$s \leq 7 \land srep \leq 3 \land nrtr \leq 2 \land fileSize \leq 1500 \land i \leq 1500 \land r \leq 4 \land rrep \leq 3 \land k \leq 2 \land l \leq 2 \land s \geq k \land s \geq l \land srep \leq fileSize \land srep \leq i \land srep \leq r \land srep \leq rrep \land nrtr \leq fileSize \land nrtr \leq i \land fileSize \geq r \land fileSize \geq rrep \land fileSize \geq k \land fileSize \geq l \land r \geq l$

Table 2: Selection of BRP inferred invariants for different parameter configurations.

It is interesting to note several things about these results. First, the submodels analysed represent at most 2% of the size of the full model, a very low percentage. Regarding the bounds themselves, it is noteworthy that the length of traces simulated is critical, particularly in the case of estimating the minimum MTTF. Note that simulating traces less than 9000 actions long, the results obtained for minimum MTTF are meager. Coincidentally, they are similar to those obtained by the full model estimation at the moment of timeout, although the estimation over submodels converges to these values much faster. However, for those submodels obtained from the simulation of longer traces, the minimum MTTF estimated spikes and in fact approaches the maximum estimated. This seems to have its correlation with the invariants that were inferred in each case, depicted in Table 2. Note that, in the invariants obtained with traces less than 9000 steps long, the variable i is restricted to no more than 1333. Recall that i indicates the number of packets of the file that have already been set. These invariants show that, for the traces analysed, some times the maximum file size (1500) was chosen, but never completely sent. For our approach, such situations would lie in the *unknown* set of the state space, and thus conservatively evaluated as failing states. However, invariants obtained for longer traces do allow i to reach its maximum of 1500, which explains the dramatic increase of the estimations. Even more, increasing the simulation length to 10000 actions does pay off in some cases, although the

Simulation for invariant inference and submodel generation			Model checking			
Traces	Length	States	Min. TAT	Time	Max. TAT	Time
500	100	117976	1725.00	0.72s	4213.90	1.15s
500	150	117976	1725.00	0.74s	4213.90	1.28s
500	200	118252	1725.00	0.76s	4278.24	1.16s
500	250	118252	1725.00	0.80s	4278.24	1.24s
1000	100	117976	1725.00	0.80s	4213.90	1.33s
1000	150	118252	1725.00	0.73s	4278.24	1.12s
1000	200	118232	1725.00	0.74s	4278.24	1.12s
1000	250	118252	1725.00	0.71s	4278.24	1.21s
1500	100	118104	1725.00	0.69s	4246.08	1.07s
1500	150	118252	1725.00	0.75s	4278.23	1.14s
1500	200	118252	1725.00	0.78s	4278.23	1.19s
1500	250	472809	1725.00	4.47s	4286.21	7.87s
2000	100	118240	1725.00	0.70s	4247.08	1.18s
2000	150	118252	1725.00	0.71s	4278.23	1.11s
2000	200	118252	1725.00	0.70s	4278.23	1.11s
2000	250	118232	1725.00	0.94s	4278.24	1.34s

Table 3: Selection of WLAN submodel TAT evaluation results for different simulation parameter configurations.

increase is not nearly as dramatic.

In the case of the maximum MTTF estimation, all submodels behave more or less uniformly. Although there are of course differences, the worst results still are much better than those obtained by full model evaluation. However, when compared with the result obtained for full model estimation, it can clearly be seen that estimation over submodels clearly pays off – the maximum MTTF estimated for submodels is, in all cases, at least 50 times as much than those obtained for the full model.

WLAN turnaround time estimation.

In this case study, we are interested in estimating the turnaround time (TAT) for both emitting stations to complete sending their intended data. That is, the time from the moment the first station intends to send data until both of them have successfully sent their data, including all necessary backoff time.

For this case study we also attempted to produce an estimate for the full model. Contrasting with the previous case study, the event under analysis is not a rare event. On the contrary, it is desirable that in every instance both stations are able to send their data in a reasonable time. During this analysis, we obtained a full model comprising about 75 million states. The minimum TAT was estimated at 1725 after timing out, while the maximum one was estimated at 4301.65, also at timeout. Turnaround time is measured in microseconds (μs).

Again, we compared this performance with our approach, with the results obtained depicted

Traces	Length	Invariant
500	100	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge slot1 \le bc2 \wedge bc1 < s2 \wedge bc1 \ge slot2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
500	150	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge slot1 \le bc2 \wedge bc1 < s2 \wedge bc1 \ge slot2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
500	200	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \ge bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
500	250	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 \le s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1000	100	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge slot1 \le bc2 \wedge bc1 < s2 \wedge bc1 \ge slot2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1000	150	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 \le s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1000	200	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1000	250	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \ge bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1500	100	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge bc1 \ge slot2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1500	150	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 \le s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1500	200	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \ge bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
1500	250	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 3 \wedge backoff1 \le 31 \wedge bc1 \le 3 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 3 \wedge col \le s1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \ge bc1 \wedge s1 > slot2 \wedge s1 \ge bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge s2 > slot2 \wedge s2 \ge bc2 \wedge slot2 \le bc2$
2000	100	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \ge bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge slot1 \le bc2 \wedge bc1 \le s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
2000	150	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 \le s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
2000	200	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 \le s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$
2000	250	$col \le 2 \wedge c1 \le 2 \wedge c2 \le 2 \wedge x1 \le 10 \wedge s1 \le 12 \wedge s1 \ge 1 \wedge slot1 \le 1 \wedge backoff1 \le 31 \wedge bc1 \le 2 \wedge x2 \le 10 \wedge s2 \le 12 \wedge s2 \ge 1 \wedge slot2 \le 1 \wedge backoff2 \le 31 \wedge bc2 \le 2 \wedge col \le s1 \wedge col \ge slot1 \wedge col \le s2 \wedge col \ge slot2 \wedge c1 < s1 \wedge c1 \le s2 \wedge c2 \le s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \le bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \le bc2$

Table 4: Selected WLAN inferred invariants for different parameter configurations.

in Tables 3 and 4.

In this case, the results are much easier to interpret. First, note the TAT estimations themselves from Table 3. The minimum turnaround time estimated is the same for all submodels evaluated and coincides with that obtained through the full model evaluation. In the case of the maximum turnaround estimation, they are not all the same, but they are all around the same value, and differ in no more than 2% from the actual value estimated through full model evaluation. However, the verification times that were necessary for estimating these results are what are most significant. For every submodel, both for the minimum TAT estimation as for the maximum, all reward estimations finished in less than 10 seconds, with most of those estimations taking much less time. This marks a stark contrast with the time needed for the full model verification. The size of the submodels evaluated is also striking. In all cases, this size is about 0.15% to 0.50% of the size of the whole model. This seems to suggest that the full model has a very large portion of behaviour that is largely irrelevant when contrasted to the behaviour that actually takes place in the wild. In fact, it is easy to see from Table 4 that although the waiting slots (*slot*1 and *slot*2) can be increased to as much as 128 different slots, the simulations only observed waiting times up to 4 of these slots. Since the slot is chosen equiprobably within the same backoff level, this seems to suggest that only the first two backoff levels were taken on all of the simulated executions. In other words, it was never necessary to increase the backoff to more than this second level.

As in the previous case study, the choice of parameters for the number of traces to simulate and the length of the simulated paths also plays a role. However, this is not as clear-cut as in the previous case. Note that the size of the submodels evaluated seems to lie either near the 120000 state mark except for one that lies near the 460000 state mark. Although the difference is notorious, the small verification times and the fact that all estimated results are similar mask these differences. We may, however, find an explanation for such a disparity in the invariants inferred–see Table 4. In the cases where a bigger submodel was generated, the second sender station was allowed to take the slot number 3 in some of the executions, while in the smaller ones it never did. Since the choice of slot is uniform, and whenever the slot 2 is available the slot 3 also is, we can only conclude that this differences are only a coincidental artefact of the stochasticity of the sampling procedure. The fact that these differences appear without regard for sample set size or length reinforces this theory.

Note that for both case studies the time measured includes only the execution of the numerical methods. The time spent on construction of the model state space, and on generating simulated traces and invariants is not considered. However, in both cases it turned out that the time spent building the model was comparable to that of generating and analysing traces–2 hours in the first case while no more than a few minutes in the second. In fact, in the second case the model can be built much faster than the generation of simulated traces. Even so, the process of trace generation by simulation is easily performed in parallel, while model building cannot in general be parallelised.

What both case studies and experiments indicate is that, through careful partial exploration of the model, we can obtain useful bounds for reward estimation (or even calculate the actual reward) with very low percentages ($< 1.5\%$) of actual state space exploration. Further, submodels that yield these results also converge very quickly to good MTTF results, much more so than when performing full model estimation. From these results it follows that, for these case studies, effort into estimating MTTF through automatically obtained submodels through model invariants of the full model pays off.

It must be noted that when a bound is obtained, it is possible that the *actual* reward value is much larger than any of those obtained. Of course, we are always limited by the fact that the actual reward may not be calculated at all, neither with partial nor full models. It can be argued, though, that it is often the case that the exact reward value is not needed as such; rather, satisfying a minimum degree of reliability or service-level values is a sufficient guarantee. Hence, methods which provide higher lower bounds faster are useful.

5 Discussion and Related Work

In this paper, we have presented a fully automated technique for reward estimation of nondeterministic and stochastic system models, which is based on the approach presented in [13]. We extend [13] to a broader class of properties than MTTF and also to support nondeterminism.

The invariant inference step of our approach, which determines the portion of the state space to be analysed over the following steps, relies on simulation of the original system. The fact that the system to simulate is nondeterministic poses additional difficulties to this simulation step.

Several works have attempted to solve this problem, especially in the context of generating simulations for Monte Carlo estimation. In these cases, it is critical that the simulation of nondeterministic transitions is performed in such a way that there is no bias in the generation (or alternatively, in such a way that this bias can be controlled and quantified), as doing so otherwise would introduce errors in the final estimation. In [8] the authors leverage on the fact that, usually, verification is performed while looking for the worst and best cases. In that sense, only the two schedulers that induce the best and worst results are of interest, and the authors propose a self-adjusting simulation algorithm that converges to these extremes.

In [4], rather than focusing on the problem of biasing scheduler selection, the authors aim at detecting whether nondeterminism can be ignored safely. As the authors point out, it is often the case that nondeterministic choices are actually behaviour-equivalent. By detecting these situations via partial order methods, it can be used to identify situations where nondeterminism can be ignored while keeping only one of the possible choices when performing simulation.

In our present work, we have opted to resolve nondeterminism by simply assuming an equiprobable distribution over the possible nondeterministic choices at a given state. However, it must be noted that, in the context of our work, *any* method of resolving nondeterminism would have been acceptable, as we are not relying on simulation to produce a statistically correct result. This is not to say that any nondeterminism resolution method will produce the same outcome, as different choices may lead to different invariants. Although the results presented in this paper are promising, it still remains to be seen if different approaches to the initial simulation might produce even better results. In particular, the choice of simulating via equiprobable distribution of nondeterministic transitions is a double-edged sword. On the one hand, by establishing a balanced choice, it maximises the chance of exploring most of the nondeterministic alternatives so that verification of all of them is carried out at a later step. But, on the other hand, some of this explored behaviour might possibly be irrelevant when calculating the maximum (or minimum) rewards, as the best and/or worst schedulers might never take some of the explored nondeterministic transitions. In this sense, adapting the approach of [8] to the simulation step of our framework might prove to be interesting. Although that proposed approach is geared towards model checking of probabilistic properties rather than reward calculations, it may be adapted to our needs. It is worth noting, however, that such an approach would need to carry out two simulation steps as opposed to the current one. This is because the approach in [8] aims at simulating executions that resemble those of the extreme scheduler that is of interest, which may be either the one providing the minimum value, or the maximum, but not both at the same time. In that sense, if we are interested in calculating both extreme values, we would need different simulation sets, one for each extreme.

6 Conclusions and further work

In this paper we have proposed an approach to estimation of quantitative properties of system models combining nondeterministic and stochasticity. The approach extends a novel combination of simulation, invariant inference and probabilistic model checking. We report on experiments that suggest that quantitative estimation using this technique can be more effective than (full model) probabilistic model checking. We believe the notion of quantitative analysis over partial yet systematic explorations offers an alternative to, and hence complements, exhaustive model exploration–as in probabilistic model checking–and partial random exploration–as in statistical model checking.

We believe the experimental results presented in this paper are promising. However further analysis is due on two fronts that need to be furthered. First, a better understanding of the relationship between the simulated set of traces (both its size as the trace length) and the submodels that result from them would be useful, as it should lead to heuristics for setting appropriate values to these parameters. Finally, the analysis of different ways to resolve nondeterminism during the simulation step should be deepened.

Bibliography

[1] A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. It Usually Works: The Temporal Logic of Stochastic Systems. *Lecture Notes in Computer Science*, pages 155–155, 1995.

[2] A. Bianco and L. De Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, pages 499–513. Springer, 1995.

[3] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. *Proc. Foundations of Software Technology and Theoretical Computer Science*, 1026:499–513, 1995.

[4] J. Bogdoll, L. Ferrer Fioriti, A. Hartmanns, and H. Hermanns. Partial order methods for statistical model checking and simulation. In *FMOODS/FORTE*, pages 59–74, 2011.

[5] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *PAPM/PROBMIV*, volume 2165 of *LNCS*, pages 39–56. Springer, 2001.

[6] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao. The Daikon system for dynamic detection of likely invariants. *Sci. Comput. Program.*, 69(1-3):35–45, Dec. 2007.

[7] L. Helmink, M. Sellink, and F. Vaandrager. Proof-checking a data link protocol. In *Proc. International Workshop on Types for Proofs and Programs (TYPES'93)*, volume 806 of *LNCS*. Springer, 1994.

[8] D. Henriques, J. Martins, P. Zuliani, A. Platzer, and E. Clarke. Statistical model checking for markov decision processes. In *QEST*, pages 84–93, 2012.

[9] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS'06 Proceedings*, volume 3920, pages 441–444. Springer, 2006.

[10] Institute of Electrical and Electronic Engineers. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1997.

[11] L. Jamieson and B. Dean. Weighted alliances in graphs. *Congressus Numerantium*, 187:76, 2007.

[12] M. R. Lyu. *Handbook of software reliability engineering*. McGraw-Hill, Inc., Hightstown, NJ, USA, 1996.

[13] E. Pavese, V. Braberman, and S. Uchitel. Automated reliability estimation over partial systematic explorations. In *Proceedings of the 35th International Conference on Software Engineering (ICSE). To appear.*, 2013. http://publicaciones.dc.uba.ar/Publications/2013/PBU13/Automated Reliability Estimation over Partial-icse2013.pdf.

[14] R. Segala. *Modelling and verification of randomized distributed real time systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[15] M. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *SFCS 1985*, pages 327–338. IEEE, Oct. 1985.

MEALS Partner Abbreviations

SAU: Saarland University, D

RWT: RWTH Aachen University, D

TUD: Technische Universität Dresden, D

INR: Institut National de Recherche en Informatique et en Automatique, FR

IMP: Imperial College of Science, Technology and Medicine, UK

ULEIC: University of Leicester, UK

TUE: Technische Universiteit Eindhoven, NL

UNC: Universidad Nacional de Córdoba, AR

UBA: Universidad de Buenos Aires, AR

UNR: Universidad Nacional de Río Cuarto, AR

ITBA: Instituto Técnológico Buenos Aires, AR