Graphical modelling for simulation and formal analysis of wireless network protocols

A. Fehnker[1], M. Fruth[2], and A. K. McIver[3]

[1] National ICT Australia, Sydney, Australia;[⋆] ansgar@nicta.com
[2] Computing Laboratory, Oxford University UK; [⋆⋆] m.fruth@comlab.ox.ac.uk
[3] Dept. Computer Science, Macquarie University, NSW 2109 Australia, and National ICT Australia; anabel@ics.mq.edu.au

Abstract. It is well-known that the performance of wireless protocols depends on the quality of the wireless links, which in turn is affected by the network topology. The aim of this paper is to investigate the use of probabilistic model checking in the analysis of performance of wireless protocols, using a probabilistic abstraction of wireless unreliability.
Our main contributions are first, to show how to formalise wireless link unreliability via probabilistic behaviour derived from the current best analytic models [12], and second, to show how such formal models can be generated automatically from a graphical representation of the network, and analysed with the PRISM model checker.
We also introduce CaVi, a graphical specification tool, which reduces the specification task to the design of the network layout, and provides a uniform design interface linking model checking with simulation. We illustrate our techniques with a randomised gossiping protocol.

Keywords: Graphical modelling, simulation, lossy communication channels, probabilistic model checking, wireless networks.

1 Introduction

Wireless networks comprise devices with limited computing power together with wireless communication. Protocols for organising large-scale activities over these networks must be tolerant to the random faults intrinsic to the wireless medium, and their effectiveness is judged by detailed performance evaluation. One of the major factors impacting on the accuracy of an evaluation method is the mathematical model for the "communication channels" and, especially important, is that it must account for the unexpected disturbances induced by noise and interference amongst close neighbours. Conventional analysis methods rely on simulators [1, 2] incorporating some measure of random faults, however simulation in this context suffers from a number of well-documented problems [9, 7] — most notable is that accurate channel models validated against physical data do

⋆ National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

⋆⋆ This work was in part supported by the EPSRC grant EP/D076625/2 not normally feature. This leads to unrealistic results of performance analyses, which can vary widely between different simulators.

An alternative to simulation is formal modelling and analysis, which is normally ideally suited to investigating complex protocols, and gives access to profiles of performance which exhaustively range over worst- and best-case behaviour. Inclusion of realistic models of wireless communication implies appeal to analytical formulae to determine the effect on performance of the spatial relationships between nodes, such as the distance and density of near neighbours. These context-dependent details however are not easily added to textual-style formal modelling languages, and indeed they militate against a clear and modular specification style.

In this paper we overcome these difficulties by proposing a simple graphical style of specification. We exploit the observations that (a) the distance between and density of nodes in a network is the major factor impacting on the integrity of wireless communication (together with physical parameters such as transmission strength); that (b) this unreliability can be abstracted to a probability that packets are lost; and that (c) the simplest way to express the crucial spatial relationships is graphically, so that the details of the abstracted probabilities are suppressed, and computed automatically from the graphical representation.

Besides its simplicity, the graphical style has other benefits in that it allows designers to visualise various performance indicators such as best- or worst-case signal strength between pairs of nodes, or the nodes' individual power consumption. Similarly the critical events occurring in a sample experiment may be "stepped through" in a typical debugging style. Finally — unlike other graphical visualisation tools — it acts as a "bridge" between formal analysis and the more conventional simulation, providing the option to investigate performance using probabilistic model checking, or to carry out more traditional system-wide simulation experiments. In both cases realistic models for wireless communication play a fundamental role.

Our specific contributions are

1. CaVi a graphical user interface specialised for modelling networks comprising wireless nodes. The tool gives immediate access to crucial performance indicators such as signal strength between pairs of nodes;
2. A translation from a CaVi model to either a formal transition-style model suitable for model checking in the PRISM model checker [10] or as input to the recently-developed Castalia simulator [3]. Castalia is novel in that it incorporates an accurate wireless channel model. The PRISM models are the first such formal models which take network topology into account. At present both Castalia and PRISM capture only flooding and gossiping protocols [4, 5].

In Sec. 2 and Sec. 3 we describe the context of wireless applications, and the challenges that arise in their formal modelling. In Sec. 4 we describe a well-known analytic model for describing unreliability of wireless links and explain how that can be used to compute the probabilistic abstractions. In Sec. 4 we illustrate how this can be incorporated in PRISM formal models for wireless protocols, and illustrate the effect on performance analysis. In Sec. 6 we introduce CaVi the graphical specification tool, and finally in Sec. 7 we demonstrate the techniques we have assembled with a case study based on gossiping.

2 Wireless communication and performance modelling

In abstract terms a wireless network consists of a collection of nodes deployed over a two-dimensional area which together run a combination of protocols in order to achieve some specific goal. During operation the nodes routinely communicate using *wireless links* which are known to be highly unreliable, and indeed can have a significant impact on the overall performance of the system. In particular not only does the reliability of the wireless links attenuates as the distance between nodes extends, but it also falls off as the density of closely clustered nodes increases, since simultaneous broadcasts from separate sources can interfere and be effectively destroyed.

Thus the operability of the wireless network can depend as much on the topology of the network as on the correctness of underlying protocols. In particular the design of protocols are specifically intended to tolerate or reduce, as much as possible, the frequency of faults arising due to the unreliability involved in wireless communication. This paper is concerned with methods and tool support to help designers understand and evaluate the effectiveness of their designs.

With this goal in mind we set out the three challenges implied by the specification and performance evaluation of emerging wireless network protocols.

1. **Network specification:** As mentioned above the network behaviour depends critically on the network topology, suggesting that the topology should be encoded as part of the specification.
   Our first problem is *how to incorporate details of distance and relative clustering as part of the specification without leading to an infeasibly complicated specification language?*
2. **Realistic mathematical models:** Currently simulation is the major tool for evaluating performance of wireless networks. Whilst emerging simulators are beginning to account for accurate mathematical models of communication [3], simulation still suffers from several drawbacks. Aside from the underlying mathematical model being remote from the specifier, the resulting performance analysis is essentially "second order", in the sense that it relies on a large number of simulation runs and, by implication, costly and time consuming.
   An alternative approach for protocol analysis is probabilistic model checking, so far under-explored as an evaluation method in the wireless domain. Model checking appears to overcome some of the problems surrounding simulation: the constructed models — Markov-style — are under direct control of the specifier, and the analysis involves direct algorithmic exploration of the associated mathematical structure. Thus the effect network parameters

have on performance can be analysed relatively easily. Despite this appealing access to sensitivity analysis, a typical formal model checking approach assumes unrealistically that the links are either completely reliable, or uniformly unreliable, which is not the case.

   Our second problem is *how should realistic wireless communication models be incorporated into formal model checking?*

3. **Scale versus accuracy:** Even if the modelling problem can be solved, the model checking technique is still only viable for small to moderately-sized networks, depending on the details of the protocol. Thus simulation is still the only feasible option for investigating system-wide properties over large networks. This indicates that if the analysis demands both an accurate evaluation of how parameters affect performance *and* a study of global network properties that both model checking *and* simulation are called for, with a consequent separate modelling effort for each.

   Our third problem is *how can the benefits of system-wide analyses be combined with the accuracy of model checking without doubling the modelling effort?*

In what follows we address all three problems. For problem (1) we extend probabilistic model checking in a novel way to account for unreliability of wireless links; for problem (2) we also introduce a graphical specification tool to make transparent the relevant details of the network topology, whilst still accounting for them in the analysis; and finally, for problem (3), we explore how the graphical specification can provide a bridge between model checking and simulation with minimal duplication of modelling effort.

To extend probabilistic model checking, we render the unreliability of wireless communication as a probability that packets get lost. To ensure that this abstraction is as realistic as possible we compute the probabilities using an analytic formula validated against experimental field data [12]. We show how these probabilities can be translated in the PRISM model checker [10] allowing accurate formal model checking to be performed after all.

Next, inspired by other graphical tools [1], we propose a graphical style of specification to reduce the specification effort, exploiting two of the above observations: specifying topological details can be done most naturally by drawing a diagram, and the "run time" probabilities of communication failure — accounting for both distance between and density of nodes — can be computed automatically from the corresponding graphical representation.

Our graphical specification tool CaVi simplifies the specification task of communication details; moreover for simple protocols it can act as a uniform modelling language combining large-scale performance analyses based on simulation with the accurate sensitivity analysis offered by model checking.

In the remainder of the paper we set out the details.

2.1 The PRISM model checker

The PRISM model checker [10] takes a specification of a system using a modelling language for describing probabilistic state transition systems. In such a model, a system is regarded as a collection of "communicating" modules, each one consisting of a set of guarded commands, with each command composed of the guard (a predicate on the variables) and a probabilistic update relation (the probabilistic assignment to variables). Modules can communicate by enforcing the assumption that same-labelled guarded commands must fire simultaneously; additionally information can be shared between modules by their reading the values of others' variables. Once specified the PRISM model checker constructs an internal representation of the system model — a Markov-style transition system for the composition of specified modules — which can then be analysed exhaustively relative to a specified property using a suite of numerical algorithms.

Property specification is via probabilistic temporal logic [6], which is expressive enough to describe many performance style properties; PRISM computes the best- and worst-case probability of satisfaction. In this paper we shall use that to analyse whether nodes eventually receive a message sent in a network protocol.

The importance of this approach (as compared to simulation for example) is that precise probabilistic results are computed exhaustively and are relevant to the entire set of executions, rather than a simulated subset.

3 Modelling lossy wireless communication

Wireless nodes typically broadcast a message on a particular frequency — in the case that several nodes broadcast using the same frequency at approximately the same time, the messages can interfere so that the receiving node only detects noise. In this section we discuss the effect on performance evaluations of the adopting various modelling assumptions used to address unreliable links.

Consider the simple network in Fig. 2 depicting a four node network. Suppose now that Source attempts to send a message to Target, but that they are too far apart to be connected directly by a wireless link. In this case Source must rely on relaying the message via the intermediate nodes $Node_A$ and $Node_B$. We consider a simple communication protocol in which Source broadcasts a message to be picked up by $Node_A$ and $Node_B$, both of which then forward the message on to Target.

Depending on the assumptions in the mathematical model used to handle the reliability of communication, very different conclusions as to the behaviour of the system can be drawn. To illustrate this we provide three simple formal models of Fig. 2, each based on different assumptions, and we discuss the implications of each one.

We model the behaviour of each node as a simple state transition system; in this small example we assume that the behaviour of Source, $Node_A$ and $Node_B$ is given by the systems set out in Fig. 1, leaving the assumptions about the reliability of communications to vary only in the model for Target [4] , set out

[4] Strictly speaking we should also include the possibility of unreliability in the communications for $Node_A$ and $Node_B$, but for the moment we assume that Source's communications to $Node_A$ and $Node_B$ are fully reliable.

below. Source has only one action, to send a message, whilst each of $Node_A$ and $Node_B$ receive the message, synchronising on the event **recv**, and attempt to forward it to Target. The message is deemed to have been delivered successfully on the occurrence of either $\mathbf{send}_{\mathsf{A}}$ or $\mathbf{send}_{\mathsf{B}}$. The case of messages interference is modelled by the event **clash**.

Next we illustrate how the analysis depends crucially on the assumptions used in the formal model. The definitions below define three possible scenarios, each one formalising a different assumption concerning simultaneous broadcasts.

$$Source \mathrel{\hat{=}} \left(\begin{array}{l} \mathbf{var}\ t\colon \{sending,\ sent\} \\ \mathbf{recv}:\ \ (t = sending) \rightarrow\ \ t{:=}\ sent \end{array}\right)$$

$$Node_A \mathrel{\hat{=}} \left(\begin{array}{l} \mathbf{var}\ f_{\mathsf{a}}\colon \{listen,\ sending\} \\ \mathbf{recv}:\ \ (f_{\mathsf{a}} = listen) \rightarrow\ \ f_{\mathsf{a}}{:=}\ sending \\ \mathbf{send}_{\mathsf{A}}:\ \ (f_{\mathsf{a}} = sending) \rightarrow\ \ f_{\mathsf{a}}{:=}\ listen \\ \mathbf{clash}:\ \ (f_{\mathsf{a}} = sending) \rightarrow\ \ f_{\mathsf{a}}{:=}\ listen \end{array}\right)$$

The definition for $Node_B$ is the same as for $Node_A$, except that the state variable is f_b and the second event is named $\mathbf{send}_{\mathsf{B}}$.

Fig. 1. Behaviour of Target and intermediate nodes $Node_A$ and $Node_B$.

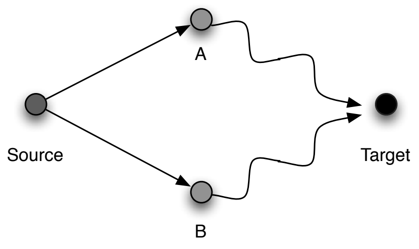


Fig. 2. Illustrating interference

(1) Worst case interference assumption: A worst case interference assumption implies that if messages are sent (almost) simultaneously from $Node_A$ and $Node_B$, then they will certainly interfere with each other. The model for $Target_1$ at Fig. 3 encodes this assumption by the state registering *noise* on the execution of the event **clash**. The whole system is now given by

$$System_1 \quad \hat{=} \quad Source \| Node_A \| Node_B \| Target_1\ ,$$

with synchronisation between same-named events. Not surprisingly, the probability of the Target's reception of the message is computed as 0.

$$
\mathit{Target}_1 \mathrel{\hat{=}} \left(\begin{array}{l}
\textbf{var } \mathit{src}\colon \{\mathit{listen,\ receive,\ noise}\} \\
\textbf{send}_{\mathsf{A}} :\ ((\mathit{src} = \mathit{listen}) \wedge (\mathsf{s_B} \neq \mathit{sending})) \rightarrow\ \mathit{src}\colon=\ \mathit{receive}; \\
\textbf{send}_{\mathsf{B}} :\ ((\mathit{src} = \mathit{listen} \wedge (\mathsf{s_A} \neq \mathit{sending})) \rightarrow\ \mathit{src}\colon=\ \mathit{receive}; \\
\textbf{clash} :\ (\mathit{src} = \mathit{listen} \wedge \mathsf{s_A} = \mathsf{s_b} = \mathit{sending}) \rightarrow\ \mathit{src}\colon=\ \mathit{noise};
\end{array}\right)
$$

Fig. 3. Worst-case interference assumption.

(b) Best case interference assumption: Alternatively we could encode the most optimistic assumption, that the Target receives the message if either one of Node_A or Node_B forward the message. Target_2 in Fig. 4 encodes this best-case assumption — it does not include an event clash in its repertoire of events, but rather only the possibility of receiving from either Node_A or Node_B, either possibility being considered depending on which of Node_A, or Node_B is ready to send. In this model,

$$\mathit{System}_2 \quad \hat{=} \quad \mathit{Source} \| \mathit{Node}_A \| \mathit{Node}_B \| \mathit{Target}_2 \ ,$$

the Target is certain to receive the message.

$$
\mathit{Target}_2 \mathrel{\hat{=}} \left(\begin{array}{l}
\textbf{var } \mathit{src}\colon \{\mathit{listen,\ receive,\ noise}\} \\
\textbf{send}_{\mathsf{A}} :\ ((\mathit{src} = \mathit{listen}) \wedge (\mathsf{s_A} = \mathit{sending})) \rightarrow\ \mathit{src}\colon=\ \mathit{receive}; \\
\textbf{send}_{\mathsf{B}} :\ ((\mathit{src} = \mathit{listen} \wedge (\mathsf{s_B} = \mathit{sending})) \rightarrow\ \mathit{src}\colon=\ \mathit{receive};
\end{array}\right)
$$

Fig. 4. Best-case interference assumption.

(c) Average case interference assumption: In reality experiments have shown that the situation lies somewhere between those worst- and best-case scenarios, and in fact the precise positioning of the Target relative to Node_A and Node_B can be crucial to the overall reliability of message relay from Source: if Target is located close to *both* the intermediate nodes (for example symmetrically between them), then their simultaneous forwarding of the message will interfere and Target will not get it. Conversely if Target is placed too far afield then, in any case, the signal strength of the received messages will be so weak as to effectively disconnect Target from the network.

We formalise this average-case assumption in Target_3 set out in Fig. 5. Here on execution of the event clash there is a probability p_r of either one of the messages (from Node_A or Node_B) arriving uncorrupted. The probability that Target now receives the message in the system defined by

$$\mathit{System}_3 \quad \hat{=} \quad \mathit{Source} \| \mathit{Node}_A \| \mathit{Node}_B \| \mathit{Target}_3$$

is now *at least* p_r.

As we shall see, the precise value of p_r — referred to below as the *link probability* — depends on a number of factors, including the distance and spatial orientation of $Node_A$ and $Node_B$ from Target, and from each other, and thus p_r itself can be thought of as an abstraction for the topological details of the network. In the next section we describe how that is done.

$$Target_3 \mathrel{\hat{=}} \left(\begin{array}{l} \mathbf{var}\ src\colon \{listen,\ receive,\ noise\} \\ \mathbf{send}_{\mathsf{A}} :\ ((src = listen) \wedge (\mathsf{s_B} \neq sending)) \rightarrow\ src\colon=\ receive; \\ \mathbf{send}_{\mathsf{B}} :\ ((src = listen \wedge (\mathsf{s_A} \neq sending)) \rightarrow\ src\colon=\ receive; \\ \mathbf{clash} :\ (src = listen \wedge \mathsf{s_A} = \mathsf{s_b} = sending) \rightarrow\ src\colon=\ receive\ {}_{\mathsf{p_r}}\!\oplus\ noise; \end{array} \right)$$

The probability p_r captures the uncertainty of receiving either one (but not both) of $Node_A$ or $Node_B$'s forwarded message. Its precise value depends on the relative distances between $Node_A$, $Node_B$ and Target.

Fig. 5. Average-case interference assumption.

4 Formal abstractions of signal strength and interference

In this section we discuss how the link probability mentioned above can be calculated more generally within arbitrary networks to take account of the distance between nodes and their relative clustering. We also discuss the other network parameters impacting on the probability.[5]

Consider first the simple case set out at Fig. 6(a) of two nodes i and j a distance $d(i,j)$ apart, with j sending a message to i. The probability that i receives j's message is computed as a function of the *signal-to-noise ratio*, $SNR_{i,j}$ which is the ratio of the *power* of the received message at i ($rx_{i,j}$), and the *noise* ($bgN_{i,j}$) generated in part by the other activities of the network, as well as general conditions of the environment. Thus $SNR_{i,j} \mathrel{\hat{=}} rx_{i,j}/bgN_{i,j}$. We discuss first the analytic formulae for the latter two quantities.

Power and noise levels: The signal strength of the received message $rxbB_{i,j}$ depends on the distance $d(i,j)$ between i and j, and the power at which j transmits, tx_j, and is given by the formula

$$rxbB_{i,j} \quad \hat{=} \quad tx_j - PLd_0 - 10(pLE)\ \log_{10}(d(i,j)/d_0)\ , \qquad (1)$$

where pLE is called the *path loss exponent*, and can be thought of as the rate at which the signal strength deteriorates with distance, and d_0 and PLd_0 are

[5] The analytic formulas referred to in this section are all taken from Zuniga and Krishnamachari [12].

scaling constants determined by the environment. The power at the receiver can now be computed directly:

$$rx_{i,j} \quad \hat{=} \quad 10^{rxdB_{i,j}/10} \tag{2}$$

Next we compute the background noise. In the simple case depicted in Fig. 6(a) where there are no neighbouring nodes, the background noise is assumed to be a constant $nbgN$ determined by the operating environment. In more complicated scenarios, depicted in Fig. 6(b), the noise generated by the other nodes in the network must be taken into account. Let $send_k$ be a function which is 1 or 0 according to whether node k is transmitting a message or not. The total background noise at receiver i interfering with the message transmitted by j is given by

$$bgN_{i,j} \quad \hat{=} \quad nbgN + \sum_{k \neq i,j} rx_{i,k} * send_k \ . \tag{3}$$

With the two quantities $bgN_{i,j}$ and $rx_{i,j}$ given at (2) and (3) respectively we can now compute the probability that i receives j's message.

Link probabilities: The current analytic models for computing the link probabilities predict that there is signal-to-noise threshold below which there is effectively zero probability that the message can be decoded by the receiver. That threshold depends on a number of network specific parameters: the *data rate* nDR, the *noise bandwidth* nBW, the *threshold probability* nTP, the *frame length* f of the message, and the *modulation type* of the transmission. Here, we use *Frequency Shift Keying* (FSK) which describes a simple method of encoding information into the carrier wave using only two states. For FSK modulation, the threshold is computed as

$$\varDelta_{i,j} \quad \hat{=} \quad -2\frac{nDR}{nBW}\log_e(2(1 - nTP^{\frac{1}{8f}}))\ . \tag{4}$$

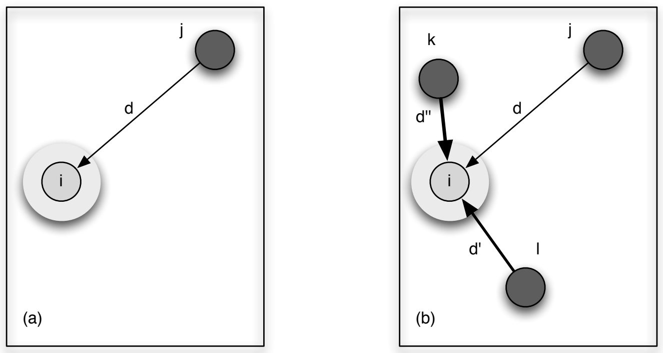
With (4) we can finally compute the link probabilities. First we compute the threshold-free probability that j's message is received by i

$$\text{snr2prob}(SNR_{i,j}) = (1 - 0.5 * \exp(-0.5\frac{nBW}{nDR}SNR_{i,j}))^{8f}\ , \tag{5}$$

where we recall that $SNR_{i,j} \hat{=} rx_{i,j}/bgN_{i,j}$, and that $bgN_{i,j}$ is given at (3). And now taking the threshold into account, we have

$$precv_{i,j} \quad \hat{=} \quad \begin{cases} 0 & \text{if } SNR_{i,j} < \varDelta_{i,j} \\ \text{snr2prob}(SNR_{i,j}) & \text{otherwise} \end{cases} \tag{6}$$

Note that since this formula depends on the mutual contribution to the noise of the surrounding nodes, this is actually a conditional probability, namely the probability that i receives j's message *given* that i does not receive the message from any of the other nodes. This together with the assumption that if $j \neq k$ then



Sender k is closest to receiver i, so its signal is strongest; Sender j's is weakest. All link probabilities are affected by the others' activities. Here d, d' and d'' are the distances from the senders to the receiver i.

Fig. 6. Signal strength varying with distance and interference

the events "i receives a message from node j" and "i receives a message from node k" are mutually disjoint, implies that we can can compute the probability P_i that any message is received by i (from whichever sender) as the sum of the individual link probabilities:

$$P_i = \sum_{j \neq i} precv_{i,j} * send_j \ , \tag{7}$$

where $send_j$ was defined above.

4.1 Translating the analytic model to PRISM

In this section we consider a PRISM model for a small example network based on that illustrated in Fig. 2 with the detailed link probabilities modelled as probabilistic transitions. The PRISM source file appears in the appendix, and here we transcribe and explain examples representing each separate section of that model.

Four nodes are numbered 0, 1, 2, 3, with 0 and 3 corresponding to the Source and Target respectively. All nodes are either active or inactive; when a node i is active (`activei = 1`) it can listen for and/or receive a message, or send one it received previously. If a node is not active, then it is inactive (`activei = 0`), in which case it only listens. We use the variable `sendi` to denote whether node i is in possession of an uncorrupted message (`sendi = 1`) which it must forward, or not (`sendi = 0`) (either because one was never received, or because it has already been forwarded). As described above, in this simple protocol, nodes listen for a message and then forward it; once it has received and sent a message a node becomes inactive. The following PRISM code formalises this behaviour for node 3, where `recvp3` is the link probability whose calculation we describe below.

```
module node3
active3:[0..1] init 1;
send3:  [0..1] init 0;

[tick] send3=0&active3=1 -> recvp3:(send3'=1)&(active3'=1)+
                                (1-recvp3):(send3'=0)&(active3'=1);
[tick] send3=1&active3=1 -> send3'=0&active3'=0;
[tick] active3=0 -> send3'=0&active3'=0;
endmodule
```

Note that this is a synchronous implementation of the abstract network described above in Sec. 3, which though easier to understand, if implemented directly would have a significant impact on space and model checking times, as well as the feasibility of automating the generation of PRISM models. In this synchronous style the nodes all behave in lockstep, synchronising on the action `tick`. The difference in behaviour between whether one node or several nodes broadcast at the same time is all accounted for in the link probabilities (rather than explicitly by including separate labelled guarded commands for each as it was explained in Sec. 3).

Next, to incorporate the link probabilities from (6) and (7) in a PRISM model, we need to compute the various quantities such as the transmission powers, the signal-to-noise ratios and thresholds for each node, taking into account their actual pairwise separations. Due to the limitations on the available arithmetical functions implemented in the current distributed version of PRISM, we precomputed $rx_{i,j}$ from (2), the power at the receiver i from message broadcast by j. These values are denoted in the PRISM model by `linRxSignal_i_j`, and appear as constant declarations for each pair of nodes. For example between nodes numbered 1 and 3, the reception power between pairs of nodes is:

```
const double linRxSignal_1_3 = 3.04330129123453E-8;
const double linRxSignal_3_1 = 3.04330129123453E-8;
```

Next the signal-to-noise ratio $SNR(i,j) = rx_{i,j}/bgN_{i,j}$ between pairs of nodes can be calculated from the above figures for reception power, given by equations (1), (2) and (3). As examples we give those quantities for $SNR(3,1)$.

```
formula snr_3_1 = (linRxSignal_3_1*send3)/
     (linRxSignal_0_1*send0 + linRxSignal_2_1*send2 + 1.0E-10);
```

Next the conditional link probabilities $precv_{i,j}$ at (7) are calculated from the precomputed thresholds, and combined in a single PRISM formula, with $precv_{3,2}$ given as an example,

```
formula Preceive_3_2 = func(max,0,(snr_3_2>=12.357925578002547)?
      func(pow,(1-0.5*func(pow,2.71828,-0.781*snr_3_2)), 8 * 25):0);
```

Finally the total link probabilities P_i are computed as the sum, as at (7), where we take the precaution of ensuring that the sum does not exceed 1, which can be caused by rounding errors.

```
formula recvp0 = 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);
formula recvp2 = 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);
```

5 Performance analysis with PRISM

In this section we demonstrate the effect on performance analysis using the analytically calculated probabilities in the PRISM model. Observe that once the background noise has been set, the only variables are the node specific parameters (such as power and signal strength) and the distances between nodes.

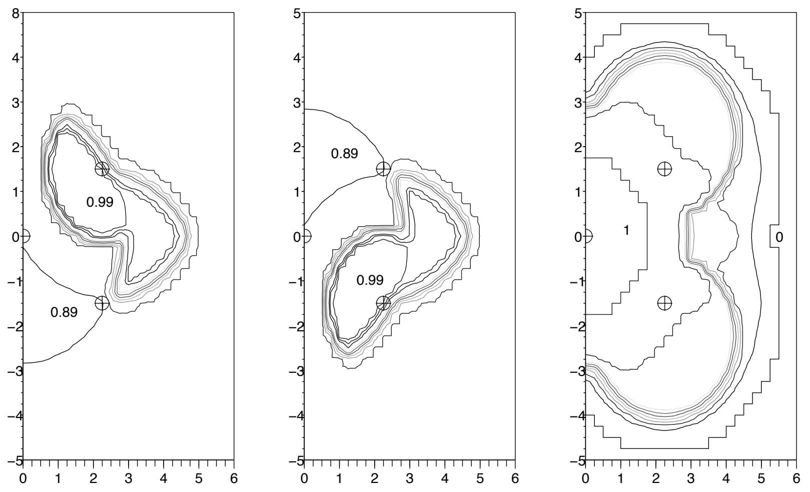
In this experiment the three nodes 0, 1 and 2 were placed so that 1 and 2 had a high chance of receiving the message broadcast by 0, and the effect on performance of nodes 3's position investigated. Thus for various positions of node 3 we computed the separate probabilities that nodes 1, 2 and 3 obtained the message. Specifically we used the PRISM model checker to compute the probability that the following temporal logic properties for strong until were satisfied:

```
Pmin=? [send1 = 0 U send1 = 1]
Pmin=? [send2 = 0 U send2 = 1]
Pmin=? [send3 = 0 U send3 = 1]
```

namely the chance that eventually `sendi = 1` for each of the nodes $i = 1, 2, 3$. The results appear as three contour plots in Fig. 7.

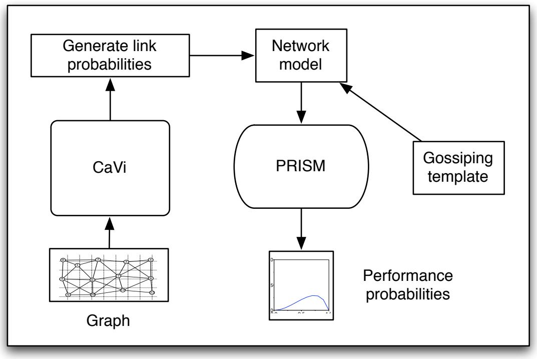
The right-hand plot, for node 3 illustrates in particular the effects of interference and distance. The 0 contour indicates that the distance is the major factor, with node 3 being far too far from any of the other nodes to receive any message. The cluster of contours around nodes 1 and 2 however shows clearly the impact of interference: if node 3 is at position $(x, y) \hat{=} (3.5, -2)$ say, it has a high chance of receiving the message from node 2, even it is too far to hear node 0 directly. As node 3 moves up vertically (increasing its y component), it becomes closer to node 1, so that interference between nodes 1 and 2 becomes the influential factor, and the probability of ever receiving the message falls rapidly. Finally the high chance of receiving the message when located *directly between* nodes 1 and 2 is due to node 3 receiving node 0's broadcast directly.

The plots for nodes 1 and 2 are symetrical, and from them we notice that the movement of node 3 has only a small effect on either of them eventually receiving the message, as the most likely scenario is that they receive the message directly from node 0. In the case that node 3 is placed closer to node 0 than either of the other two, node 3 actually acts as the intermediary, increasing the chance that node 1 say receives the message, in the case that it didn't receive it directly from the source.



The three placements of $\oplus$ in each diagram indicate the static positions of nodes $0, 1$ and 2. The contour plots show the probability that the respective nodes eventually receive a message. The contour lines for nodes 1 and 2 show the iso-probabilites in steps of 0.01, and for node 3 in steps of 0.1.

Fig. 7. Analysis of interference



This diagram illustrates how CaVi interfaces with PRISM for analysing gossiping protocols. The user develops a graphical representation of the network, from which CaVi can compute the link probabilities. These are combined with the flooding template to produce an input file for PRISM describing the model. The PRISM tool can then be used to analyse performance.

Fig. 8. CaVi and PRISM

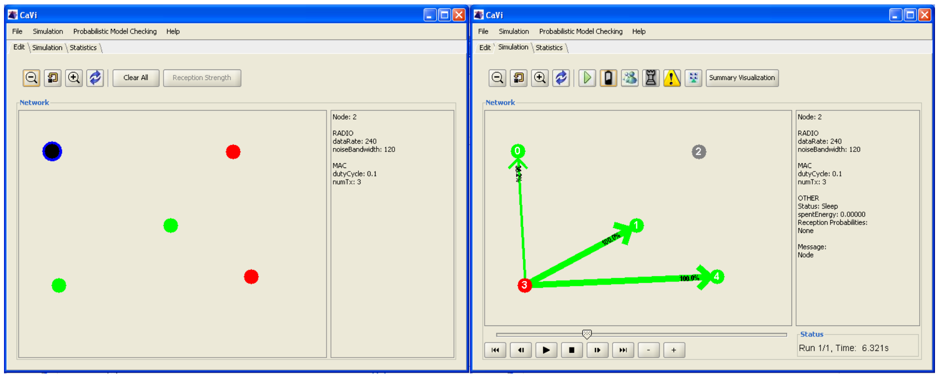
6 CaVi: A graphical specification tool

As we have seen, it is possible to formalise interference by using a probabilistic abstraction, but the detailed calculations required to introduce them into a PRISM model for example are too intricate to do by hand. To overcome this problem we have designed and implemented CaVi, a graphical specification tool which can automate the procedure of preparing a formal PRISM model with link probabilities computed directly from a graphical representation of a network. This eases considerably the task of specification in situations when the nodes all execute identical code.

The main feature of CaVi is its graphical interface, with which a user can design a specific network layout. Nodes may be created in a "drag-and-drop" fashion, and the properties of individual nodes (such as the power and signal strength) may be tuned as necessary via individual node menus of parameters. During development a user can visualise the worst- and best-case link probabilities, calculated from equation (6).

In Fig. 9 we illustrate two examples of how the graphical interface may be used in the design and analysis. The figure shows two panes, with the left being the pane where designers may create and edit a network, and the pane on the right is for visualising the results of simulation experiments. In the left-hand pane, for example, a user may indicate which node is the receiving node (in this case the central node), and the others are assumed to be senders. Colours then differentiate between nodes whose messages will be almost certainly lost (red), have a good chance of succeeding (green), or merely a variable chance (yellow).

The pane on the right indicates how the events may be viewed as a result of a simulation experiment. Simulation allows the user to "step through" an example execution sequence in a dynamic experiment. In this case the display shows which of the nodes is sending, to whom they are connected, and with what strength. In this snapshot the bottom left node is transmitting, and is connected to all but the top right node. The thickness of the arrows indicate the strength of the connection, with the vertical connection being somewhat weaker than the other two.



Node: 2

RADIO
dataRate: 240
noiseBandwidth: 120

MAC
dutyCycle: 0.1
numTx: 3

Node: 2

RADIO
dataRate: 240
noiseBandwidth: 120

MAC
dutyCycle: 0.1
numTx: 3

OTHER
Status: Sleep
spentEnergy: 0.00000
Reception Probabilities:
None

Message:
Node

Run 1/1, Time: 6.321s

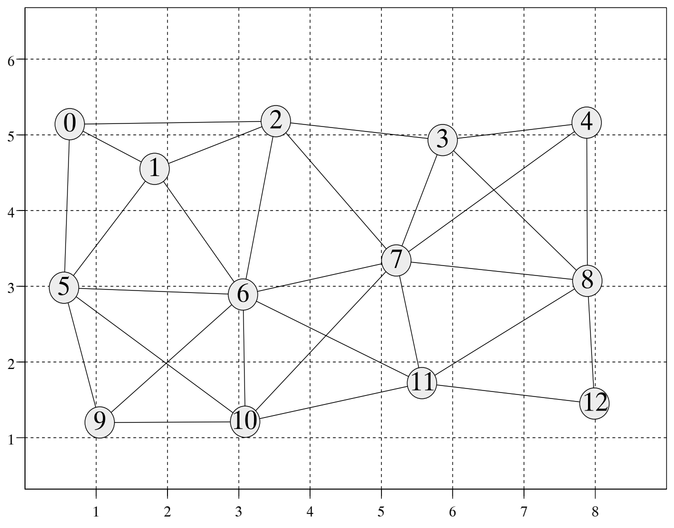
Fig. 9. CaVi:Visualising network performance indicators

Once the network is specified, the graphical representation forms the basis for formal models which take account of the effect of the topology in terms of the link probabilities. Using a template for the per-node behaviour of the protocol, the functions used to compute the actual probabilities are printed to a PRISM model file, together with an instantiation of a pre-prepared template for each numbered node. Currently we only have a template for gossiping- and flooding-style protocols, although the aim would be to expand that to other common patterns. An example of the automatically-generated PRISM model for the four node network is provided in the appendix. This model can then be fed into PRISM for detailed performance evaluation. The diagram at Fig. 8 illustrates the relation between CaVi and PRISM.

CaVi can also generate an input file for the Castalia wireless simulator, which we discuss briefly in Sec. 7.2 below. Other features of CaVi include some visualisation techniques of network performance indicators, and simulation runs, discussed in detail elsewhere [8].

7 Putting it all together: randomised gossiping

In this section we show how to use the CaVi tool for investigating a PRISM model for a probabilistic gossiping protocol.



Node 0 is the node from which the message originates; the other nodes follow a generic gossiping protocol. Lines connecting nodes indicate active connections in the network.

Fig. 10. Graphical specification of a 13-node network

We consider the problem of designing a flooding protocol to maximise the probability that the message is distributed over a network. It is known that using a simple flooding protocol such as described in Sec. 5, where nodes send as soon as they receive the message suffers from some serious performance issues, due to the high likelihood of interference. To mitigate this problem, variations of this basic scheme have been proposed in which nodes only forward a received message with probability p. The aim is to choose the value of p to optimise the probability of the message being received by all nodes.

Using the CaVi tool, we first specify the network topology by placing the nodes in the arrangement given in Fig. 10. Here node 0 is assumed to be the originator of the message to be distributed. The other nodes' behaviour is given as for the example in Sec. 5, but with an additional parameter, which is the probability of sending. In the PRISM model that is given by `Psend` and in this study it is the same for all nodes throughout the network.

For example, node 1's behaviour is described by the PRISM model below, where now the chance of a message being forwarded is combined with the link probability, so that the overall probability that a message is received is `Psend*recvpi`, and the chance of it not being received is `(1-Psend)*recvpi`.

```
module node1
active1:[0..1] init 1;
send1:  [0..1] init 0;

[tick] send1=0&active1=1 -> Psend*recvp1:(send1'=1)&(active1'=1)+
                          (1-Psend)*recvp1:(send1'=0)&(active1'=0)+
                           (1-recvp1):(send1'=0)&(active1'=1);
[tick] send1=1&active1=1 -> send1'=0&active1'=0;
[tick] active1=0 -> send1'=0&active1'=0;
endmodule
```

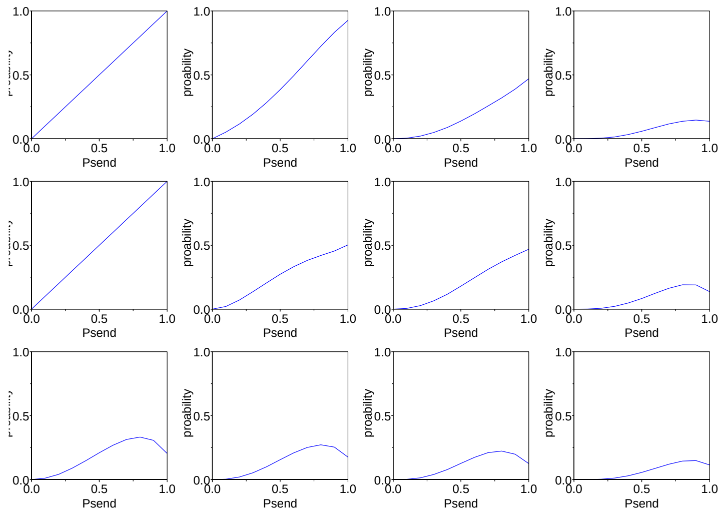
As before the probabilities for receiving `recvpi` are calculated automatically by the CaVi tool and combined with a node template to create automatically a PRISM model for gossiping in the network with layout at Fig. 10. Next we used PRISM to compute the per-node probabilities of eventually receiving the message. The results appear at Fig. 11 where we have plotted a separate graph for each node, with the vertical axes being the probability that the message is eventually received by that node, and the horizontal axis is `Psend`, which varies between 0 and 1.

The results show that the nodes 1, 2 and 5 clustered close to the originator node 0 have a probability of eventually receiving proportional to `Psend`, since they receive the message (if at all) directly from node 0, and since there can be no interference when only node 0 broadcasts. The nodes 3, 4, 7, 8, 11 and 12, all positioned too far away to receive it first-hand however have a non-linear relationship with `Psend`. When `Psend` is too low, then they have a very slim chance of receiving the message at all, since they are relying on a chain of nodes to forward their received message, and in this case the forwarding probability `Psend` is very low for each. On the other hand these nodes also have a low chance of ever receiving the message when `Psend` is high, because although the intermediate nodes will send with high probability, their messages also have a high chance of being destroyed by interference. The network-wide optimal value for `Psend` appears to be around 0.8.

7.1 Discussion

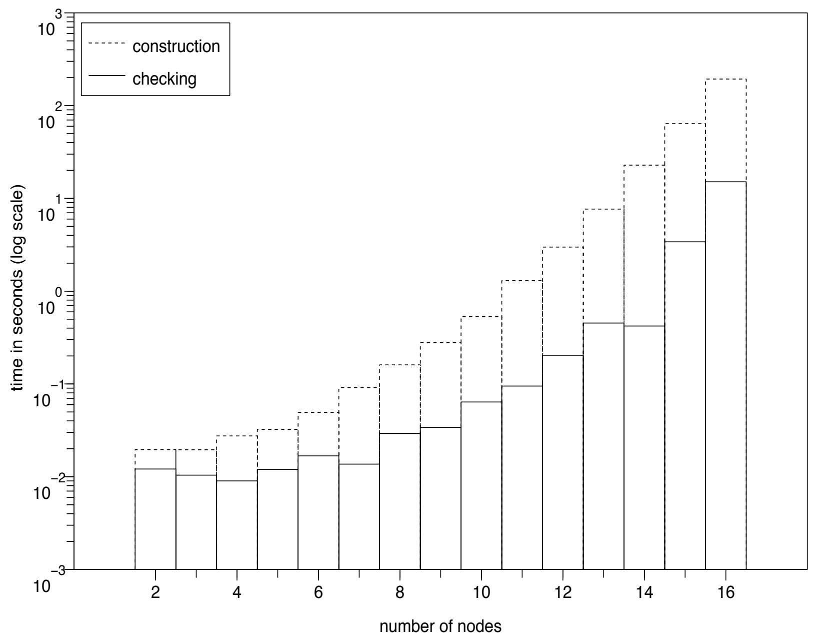
The use of CaVi to include the network topology characteristics to generate the formal model is an important step. Whilst the formulae for computing the probabilities are uniform, the task of preparing them by hand would be too time consuming and prone to error.

The templates for gossiping we use for the subsequent generation of the network models combine receiving and forwarding in a single probabilistic transition, leading to very compact internal PRISM representations of the constructed system. Fig. 13 and Fig. 14 for example give some idea as to the growth rate of the number of states and transitions as the number of network nodes increases. Though the growth is still exponential for substantially-sized networks (16 nodes) the actual size is still well within the capability of the PRISM. Similarly Fig. 12 shows the actual time spent by PRISM to construct and perform



Each graph represents the probability that node i eventually receives the message. The top row of graphs correspond to nodes 1, 2, 3, 4 Fig. 10; the second row to 5, 6, 7, 8, and the third row to 9, 10, 11, and 12.
For each graph the horizontal axis is the probability `Psend`, and the vertical axis is the probability of eventually receiving the message.

Fig. 11. Per-node probabilities of eventually receiving the message



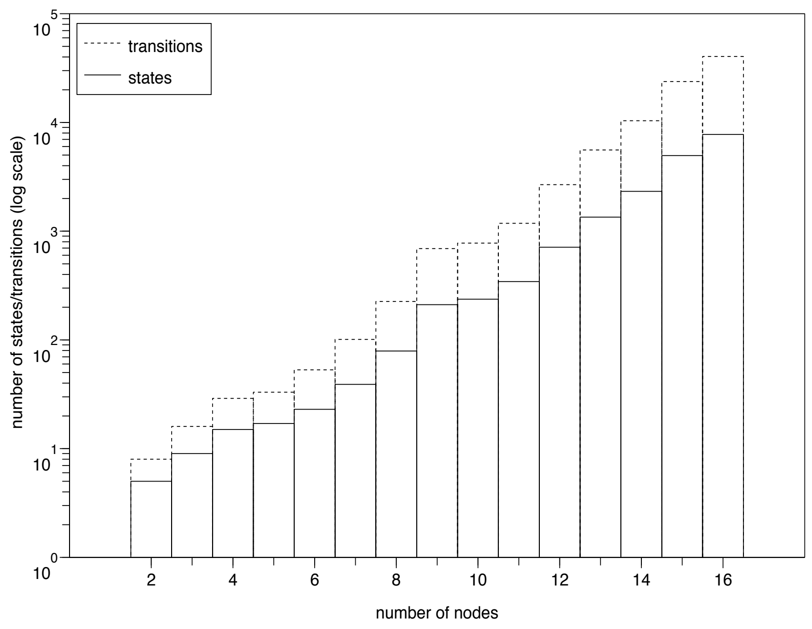
This shows the time in seconds (with a log scale) for the time to build and model check the generated gossiping models in PRISM. The dashed bars are the model building times, and the solid bars the model checking times.

Fig. 12. Model checking and model building times

the model checking to produce the plots in Fig. 11. Interestingly the time to construct these models is an order of magnitude greater than the time to do the model checking, and this is largely due to the time spent parsing the function definitions for the calculation of the link probabilities.

7.2 Model checking and simulation

Castalia [3] is a recently-developed simulator for wireless networks, whose novelty is that it incorporates an accurate model for wireless communication. It takes as input a file containing parameters describing the power, signal strength, "geographical" position of each node; once these have been specified, the simulator executes by effectively stepping through a sequence of possible states. Where the behaviour depends on the result of a random event, the simulator generates a random number to resolve the choice. Thus the simulator recreates, as far as possible, the results that would be obtained from testing the physical system. Statements about the performance of the system are based on statistical analysis



This shows the number of states and transitions (with a log scale) of the constructed gossiping models, varying with the number of nodes in the network.

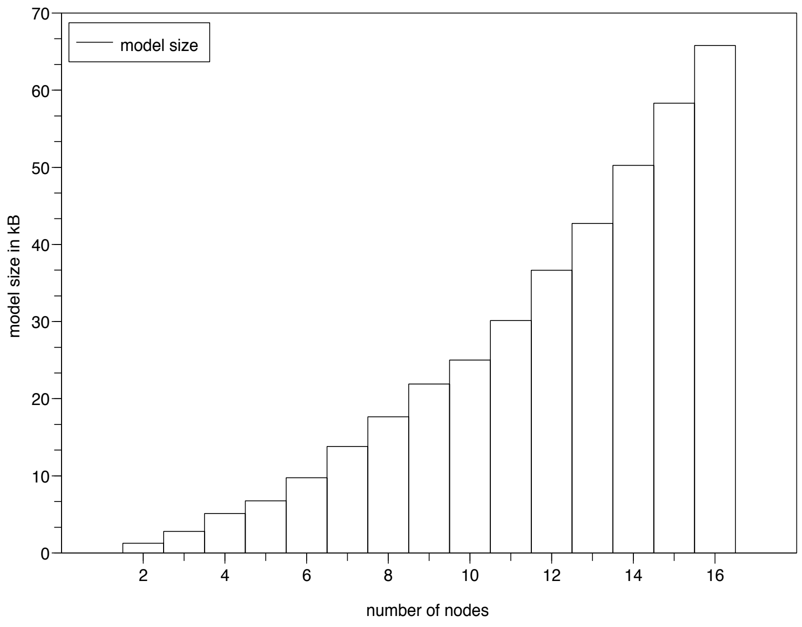
Fig. 13. The number of states and transitions

of a large collection of many simulation runs. The errors in this kind of analysis come from the statistical analysis as well as from inaccuracies in quantifying the random events in the simulation model.

The advantages of simulation however are that since it only records the result of an actual run, it is able to cope with large networks and thus can still give a performance forecast of a network made up of many nodes.

On the other hand a very large number of simulation are runs required to obtain the same accuracy as can be obtained with model checking [11] making the overall enterprise very costly in time.

Currently we are able to generate both PRISM or Castalia models from the graphical input to CaVi, and thus we have established it as a single graphical interface between model checking and simulation. We envisage that one use of such an interface would be the ability to visualise the results obtained from both in a uniform way. Such a "bridging language" would allow "counterexamples" computed via model checking to be validated in the simulator, for example, although how best to combine model checking and simulation most effectively is still a topic for research.



This shows how the size of the model (in kilobytes, with a log scale) varies with the number of nodes in the network.

Fig. 14. Memory requirements

8 Conclusions and future work

In this paper we have described a prototype tool which supports a uniform modelling approach optimised for specifying wireless protocols. Its main features include the capabilities to take account of the topology and other parameters of the network which, experiments have shown, have a major impact on the integrity of the communication. The CaVi tool allows the specification of a network via a graphical interface, and the automated generation of formats for simulation and model checking. Detailed performance indicators may be visualised during specification of the network, as well as the results of subsequent simulation and model checking experiments.

The principal difference between CaVi and other specification tools is the link it provides between simulation and formal model checking. To simplify the details related to the topology in the formal specification task, we use a translation directly to link probabilities. Those probabilities are calculated according to a validated analytic formula.

Currently we only supply templates for gossiping and flooding protocols; whilst we do not envisage a translation from a CaVi model of an arbitrary protocol to PRISM, we would aim rather to provide a library of templates for certain classes of protocol whose precise behaviour can be defined by a number of parameters, in the same way that models are defined in Castalia.

An exploration of how best to combine visualisation of results and network diagnostics using both model checking and simulation results is also an interesting topic for future research.

Acknowledgements: We thank Viet Cuong Nguyen and Michael Ma for help with the implementation of CaVi.

References

1. OPNET.
   `http://www.opnet.com/`.
2. The network simulator ns-2.
   `http://www.isi.edu/nsnam/ns/`.
3. A. Boulis. Castalia: A simulator for wireless sensor networks.
   `http://castalia.npc.nicta.com.au`.
4. A.Fehnker and P. Gao. Formal verification and simulation for performance analysis of probabilistic broadcast protocols. In *5'th International Conference, ADHOC-NOW*, volume 4104 of *LNCS*, pages 128–141. Springer, 2006.
5. A.Fehnker and A. McIver. Formal analysis of wireless protocols. In *Proc 2nd International Symposium in Leveraging applications in formal methods, verification and validation*, 2006.
6. A. Aziz, V. Singhal, F. Balarinand R.K. Brayton, and A.L. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *Computer-Aided Verification, 7th Intl. Workshop*, volume 939 of *LNCS*, pages 155–65. Springer Verlag, 1995.
7. D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 38–43. ACM Press, 2002.
8. A. Fehnker, M. Fruth, and A.K. McIver. Cavi: A graphical specification tool for wireless network protocols. In preparation., 2008.
9. K. Kotz, C. Newport, R.S.Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 78–82. ACM Press, 2004.
10. PRISM. Probabilistic symbolic model checker.
    `http://www.prismmodelchecker.org/`.
11. H. L. S. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking: An empirical study. In *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, LNCS, pages 40–60, 2004.
12. M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, pages 517–526. IEEE, 2004.

A Automatically-generated PRISM model for a four-node gossiping network

```
probabilistic

const double PsendingNode0 = 1.0;
const double PsendingNode1 = 1.0;
const double PsendingNode2 = 1.0;
const double PsendingNode3 = 1.0;

const double linRxSignal_0_1 = 1.5112050684404692E-9;
const double linRxSignal_0_2 = 1.5112050684404692E-9;
const double linRxSignal_0_3 = 1.0345994570907724E-8;
const double linRxSignal_1_0 = 1.5112050684404692E-9;
const double linRxSignal_1_2 = 1.6211305024389717E-9;
const double linRxSignal_1_3 = 3.04330129123453E-8;
const double linRxSignal_2_0 = 1.5112050684404692E-9;
const double linRxSignal_2_1 = 1.6211305024389717E-9;
const double linRxSignal_2_3 = 3.21510813957935E-8;
const double linRxSignal_3_0 = 1.0345994570907724E-8;
const double linRxSignal_3_1 = 3.04330129123453E-8;
const double linRxSignal_3_2 = 3.21510813957935E-8;




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>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_0_1)), 8 * 25):0);
formula Preceive_0_2 = func(max,0,
(snr_0_2>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_0_2)), 8 * 25):0);
formula Preceive_0_3 = func(max,0,
(snr_0_3>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_0_3)), 8 * 25):0);
formula Preceive_1_0 = func(max,0,
(snr_1_0>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_1_0)), 8 * 25):0);
formula Preceive_1_2 = func(max,0,
(snr_1_2>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_1_2)), 8 * 25):0);
formula Preceive_1_3 = func(max,0,
(snr_1_3>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_1_3)), 8 * 25):0);
formula Preceive_2_0 = func(max,0,
(snr_2_0>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_2_0)), 8 * 25):0);
formula Preceive_2_1 = func(max,0,
(snr_2_1>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_2_1)), 8 * 25):0);
formula Preceive_2_3 = func(max,0,
(snr_2_3>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_2_3)), 8 * 25):0);
formula Preceive_3_0 = func(max,0,
(snr_3_0>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_3_0)), 8 * 25):0);
formula Preceive_3_1 = func(max,0,
(snr_3_1>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_3_1)), 8 * 25):0);
formula Preceive_3_2 = func(max,0,
(snr_3_2>=12.357925578002547)?func(pow,(1-0.5*func(pow,
2.71828,-0.781*snr_3_2)), 8 * 25):0);


formula recvp0 = 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);
formula recvp2 = 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);
```

```
module node0
active0:[0..1] 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 node1
active1:[0..1] init 1;
send1:  [0..1] init 0;

[tick] send1=0&active1=1 -> PsendingNode1*recvp1:(send1'=1)&
(active1'=1)+(1-PsendingNode1)*recvp1:(send1'=0)&(active1'=0)+(1-
recvp1):(send1'=0)&(active1'=1);
[tick] send1=1&active1=1 -> send1'=0&active1'=0;
[tick] active1=0 -> send1'=0&active1'=0;
endmodule

module node2
active2:[0..1] init 1;
send2:  [0..1] init 0;

[tick] send2=0&active2=1 -> PsendingNode2*recvp2:(send2'=1)&
(active2'=1)+(1-PsendingNode2)*recvp2:(send2'=0)&(active2'=0)+(1-
recvp2):(send2'=0)&(active2'=1);
[tick] send2=1&active2=1 -> send2'=0&active2'=0;
[tick] active2=0 -> send2'=0&active2'=0;
endmodule

module node3
active3:[0..1] init 1;
send3:  [0..1] init 0;

[tick] send3=0&active3=1 -> PsendingNode3*recvp3:(send3'=1)&
(active3'=1)+(1-PsendingNode3)*recvp3:(send3'=0)&(active3'=0)+(1-
recvp3):(send3'=0)&(active3'=1);
[tick] send3=1&active3=1 -> send3'=0&active3'=0;
[tick] active3=0 -> send3'=0&active3'=0;
endmodule
```