Protocol analysis via probabilistic model checking with PRISM

Marta Kwiatkowska
School of Computer Science

UNIVERSITY OF BIRMINGHAM

www.cs.bham.ac.uk/~mzk

UC Berkeley, 7th December 2006

Overview

- Ubiquitous computing: the trends and challenges
  - Why probability?
  - Role of verification techniques
- Probabilistic/quantitative model checking
  - What does it involve?
  - Models and specifications
  - Quantitative analysis
- Lessons learnt
  - Bluetooth device discovery
- Future directions
  - and challenges that remain!

Ubiquitous computing: the trends…

- Devices, ever smaller
  - Laptops, phones, PDAs, …
  - Sensors, motes, Smartdust, …
- Networking, wireless, wired & global
  - Mobile ad hoc
  - Wireless everywhere
  - Internet everywhere
  - Global connectivity
- Systems/software
  - Decentralised
  - Self-*
  - Autonomous
  - Mobile
  - Adaptive
  - Context-aware

Ubiquitous computing: the challenge

- Grand Challenge in Computing
  - Initiated in the UK but global
- How to design & engineer
  - Complex
  - Reliable
  - Reconfigurable
  - Fast & efficient
  - Large-scale networked systems?
- How to ensure correctness, dependability, and performance?
  - The science, or rigorous foundation
  - Verification, techniques and tools
- How to embed within the natural environment
  - Human/computer interfaces
  - Social & ethical implications

This talk: emphasis on verification

- Automated techniques for the assurance of properties below, e.g. using model checking,
  - safety
  - security, privacy & trust
  - performance
  - dependability
- NB, quantitative, as well as qualitative requirements:
  - How quickly will I receive a reply in my car network?
  - Is my mobile phone energy efficient?
  - How trustworthy is the supplier?
- Our focus, probabilistic model checking
  - Probabilistic models, also costs/rewards
  - Range of quantitative analyses
  - Automated verification techniques

Why probability?

- Randomisation used in distributed coordination algorithms
  - As a symmetry breaker, in gossip routing to reduce flooding
- To model uncertainty and performance
  - To quantify rate of failures, express Quality of Service
- For quantitative analysis of software and systems
  - To quantify resource usage given a policy
    - "the minimum battery capacity for a given scenario is .."
- In evidence-based, statistical analysis of behaviours
  - To quantify trust, anonymity, etc
- In modelling of biological processes
    - "the probability that there are 50 Na molecules at time 10 is … "
    - "the expected long-run percentage of Na molecules is … "

Real-world protocol examples

- Protocols featuring randomisation
  - Randomised back-off schemes
    - E.g. IEEE 802.11 (WiFi) Wireless LAN MAC protocol
  - Random choice of waiting time
    - Bluetooth, device discovery phase
  - Random choice of routes to destination
    - Crowds, anonymity protocol for internet routing
  - Random choice over a set of possible addresses
    - IPv4 dynamic configuration (link-local addressing)
  - and more

- Continuous probability distribution needed to model network traffic, node mobility, random delays…

Verification techniques

- Probabilistic model checking, e.g. this lecture
  - Finite-state models only, but automatic
  - Quantitative analysis, including probabilistic reachability and expectations
  - Broad range of models
  - Emphasis on experimentation and bug-hunting rather than verification for all scenarios

- Program correctness proofs
  - Not limited to finite-state models
  - Mechanised using theorem provers
  - Not automatic and involved
  - More limited range of models and specifications
  - Emphasis on finding correctness proofs that apply in all scenarios

Probabilistic model checking inputs

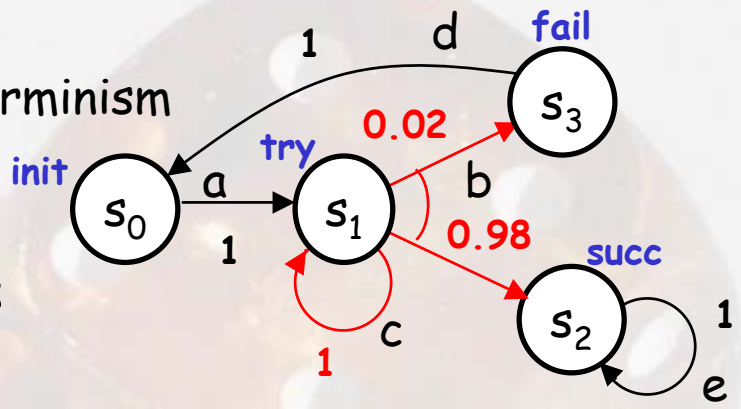
- Models : variants of Markov chains
  - Discrete-Time Markov Chains (DTMCs)
  - Markov Decision Processes (MDPs)
  - Continuous-Time Markov Chains (CTMCs)
  - Probabilistic Time Automata (PTAs)

- Specifications (informally)
  - "probability of shutdown occurring is at most…"
  - "probability of delivery within time deadline is …"
  - "expected time to message delivery is …"
  - "expected power consumption is …"

- Specifications (formally)
  - Extensions of temporal logic with probabilistic and expectation operators (PCTL, CSL, PTCTL)
  - Probability, time, cost/rewards

Markov Decision Processes (MDPs)

- Characteristics:
  - Discrete probability, nondeterminism
  - No real-time



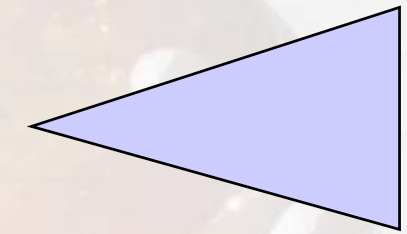
- Formally, $(S, s_0, Steps, L, Act, c)$:
  - S finite set of states
  - s_0 initial state
  - $Steps$ maps states s to sets of probability distributions μ over S
  - Act labelling of steps with actions
  - L: $S \rightarrow 2^{AP}$ atomic propositions
  - c: $S \times Act \rightarrow \mathbb{R}_{\geq 0}$ cost function

- Unfold into infinite paths $s_0 a_0 \mu_0 s_1 a_1 \mu_1 s_2 \ldots$ s.t. $\mu_i(s_i, s_{i+1}) > 0$, all i

- Probability space induced on $Path_s$ by adversary (policy) A mapping finite path $s_0 a_0 \mu_0 s_1 a_1 \mu_1 \ldots s_n$ to a distribution from s_n

Probability and cost

- Intuitively, for a fixed adversary $A \in Adv$:
  - Sample space = infinite paths $Path^A_s$ from s
  - Event = set of paths
  - Basic event = cone



- Formally, probability space $(Path^A_s, \Sigma, Pr^A)$
  - Assign probability P(.) to finite paths $\omega = s_0 a_0 \mu_0 s_1 a_1 \mu_1 s_2 a_2 \ldots s_n$
  - Define $Pr^A(C(\omega)) = P(\omega)$, for cones $C(\omega)$:
    $C(\omega) = \{ \pi \in Path^A_s \mid \omega \text{ is prefix of } \pi \}$

- Then cost for a finite path ω and target set $F \subseteq S$

$$cost(F)(\omega) = \begin{cases} \sum_{i=0}^{\min\{i : \omega(i) \in F\}} c(a) & \text{if } \exists\, i.\ \omega(i) \in F, \\ & (a,\mu) \in Steps(\omega(i-1)) \\ \infty \text{ otherwise} & \end{cases}$$

The logic PCTL: syntax

- Probabilistic Computation Tree Logic [HJ94,BdA95,BK98]
  - Based on CTL, for DTMCs/MDPs
  - Add probabilistic operator, e.g. send $\rightarrow \mathcal{P}_{\geq 0.9}(\Diamond \text{deliver})$
  - and expected cost operator, e.g. $\mathcal{E}_{> 1}(\text{heads})$

- The syntax of state and path formulas of PCTL is:

$$\phi ::= \text{true} \mid a \mid \phi \text{ Æ } \phi \mid \neg\phi \mid \mathcal{P}_{\sim p}(\alpha) \mid \mathcal{E}_{\sim d}(\phi)$$

$$\alpha ::= X\ \phi \mid \phi\ U\ \phi$$

where $p \in [0,1]$ is a probability threshold, $d \in \mathbb{R}_{\geq 0}$ is a cost bound and $\sim \in \{ <, >, \ldots \}$

The logic PCTL: semantics

- Semantics is parameterised by a class of adversaries Adv
  - "under any scheduling, the probability/cost is … at state s"
  - reasoning about worst-case/best-case scenario

- The probabilistic operator:

$$s \models_{Adv} \mathcal{P}_{\sim p}(\alpha) \quad \Leftrightarrow \quad Pr^A_s \{ \pi \in Path^A_s \mid \pi \models_{Adv} \alpha \} \sim p \text{ for all } A \in Adv$$

- The expectation operator:

$$s \models_{Adv} \mathcal{E}_{\sim d}(\phi) \quad \Leftrightarrow \quad E^A_s (cost\{ s' \in S \mid s' \models_{Adv} \phi \}) \sim d \text{ wrt } Pr^A_s, \text{ for all } A \in Adv$$

- Semantics of remaining formulas standard

PCTL Until model checking for MDPs

- Reduces to minimum/maximum probability computation
  - $p^{max}_s(\alpha)$, $p^{min}_s(\alpha)$, defined as sup/inf over all adversaries

- Obtain linear optimisation problems solvable iteratively, e.g.

  $Sat(\mathcal{P}_{\geq p}(\phi_1 \, U \, \phi_2)) \quad \Leftrightarrow \quad \{s \in S \mid x_s \geq p\}$

  Maximise $\sum_{s \in S?} x_s$ subject to the constraints:
  $x_s \leq \sum_{s' \in S?} \mu(s') \cdot x_{s'} + \sum_{s' \in Syes} \mu(s')$
  for all $s \in S?$, $(a, \mu) \in Steps(s)$

- Note
  - S^{yes} (S^{no}) satisfy until with probability 1 (0) resp.
  - $S? = S \setminus (S^{no} \cup S^{yes})$

PCTL Until: iterative solution

- S^{yes}, S^{no} can also be precomputed by graph traversal (BDD fixed point) [dAKNP97]

- The linear equation generalises to linear optimisation problems solvable iteratively, e.g.

$$\text{Sat}(\mathcal{P}_{\geq p}(\phi_1 \,U\, \phi_2)) \quad \Leftrightarrow \quad \{s \in S \mid x_s \geq p\}$$

$$x_s^{n+1} = \begin{cases} 0 & \text{if } s \in S^{no}, \text{ all } n \\ 1 & \text{if } s \in S^{yes}, \text{ all } n \\ \min_{\mu \in \text{Steps}(s)} \sum_{s' \in S} \mu(s') \cdot x_{s'}^{\,n} & \text{if } s \in S \setminus (S^{no} \cup S^{yes}) \end{cases}$$

- Need to combine
  - Conventional graph-theoretic traversal
  - Linear optimisation (simplified value iteration)

Expected cost model checking

- Reduces to minimum/maximum expected cost computation
  - $e^{max}_s(\phi)$, $e^{min}_s(\phi)$, defined as sup/inf over all adversaries

- Again, obtain linear optimisation problems solvable iteratively, e.g.

  $Sat(\mathcal{E}_{\geq d}(\phi)) \quad \Leftrightarrow \quad \{s \in S \mid x_s \geq d\}$

  Maximise $\sum_{s \in S?} x_s$ subject to the constraints:
  $x_s \leq c(a) + \sum_{s' \in Syes} \mu(s') \cdot x_{s'}$
  for all $s \in S?$, $(a, \mu) \in Steps(s)$

- Note
  - S^{yes}, S^{no} , $S? = S \setminus (S^{no} \cup S^{yes})$ can be precomputed as before
  - Unique solution, cost possibly ∞
  - Algorithm of [dAlf'97]

Probabilistic model checking involves…

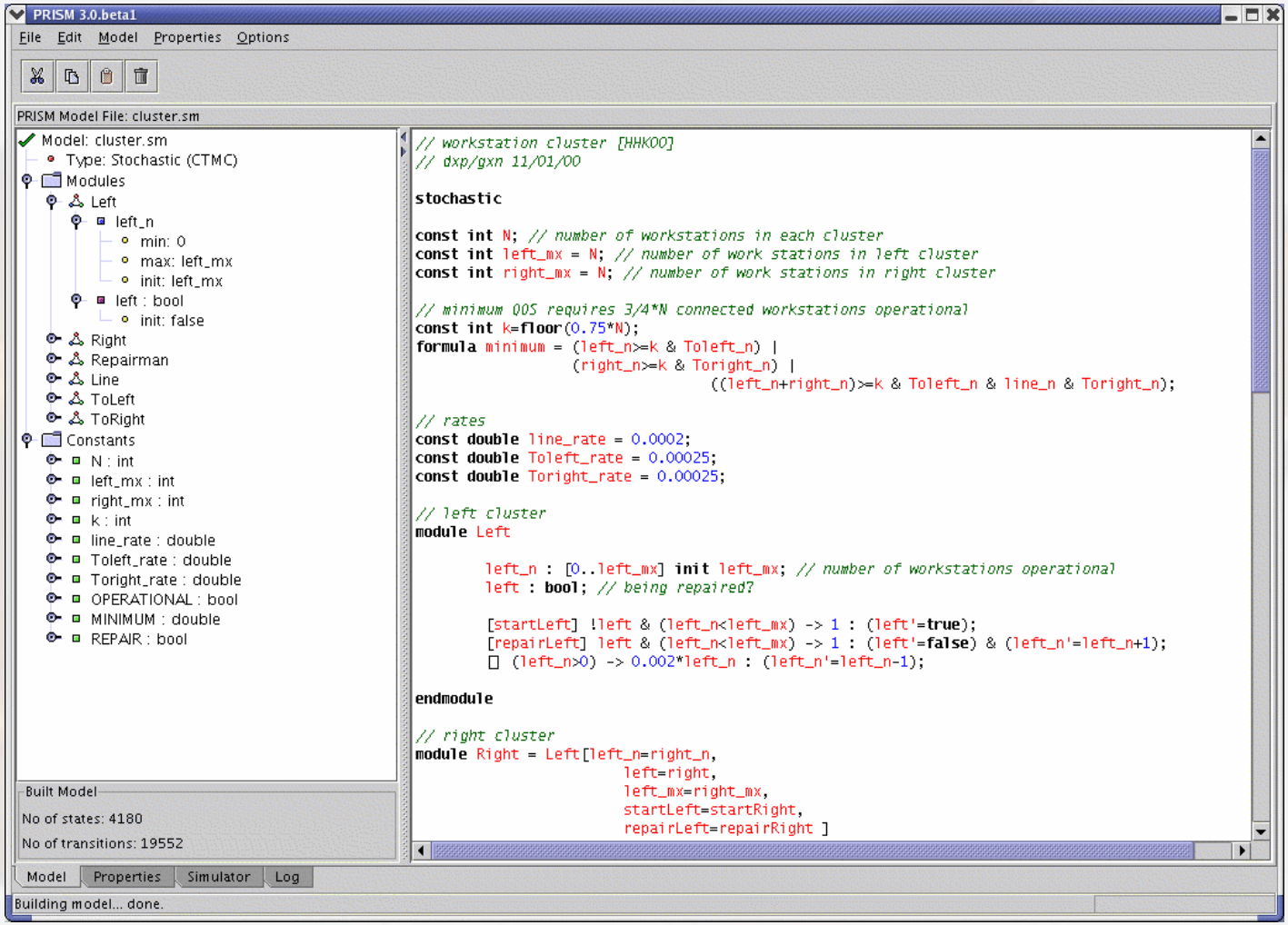
- Construction of models
  - from a high-level modelling language, such as a probabilistic process algebra or probabilistic automaton
  - e.g. Markov decision processes (MDPs)

- Implementation of probabilistic model checking algorithms
  - graph-theoretical algorithms, combined with
    - (probabilistic) reachability
    - qualitative model checking (for 0/1 probability)
  - numerical computation – iterative methods
    - quantitative model checking (plot probability values, expectations, rewards, steady-state, etc, for a range of parameters)
    - typically, linear equation or linear optimisation
    - exhaustive, unlike simulation
  - also sampling-based (statistical) for approximate analysis
    - e.g. hypothesis testing based on simulation runs

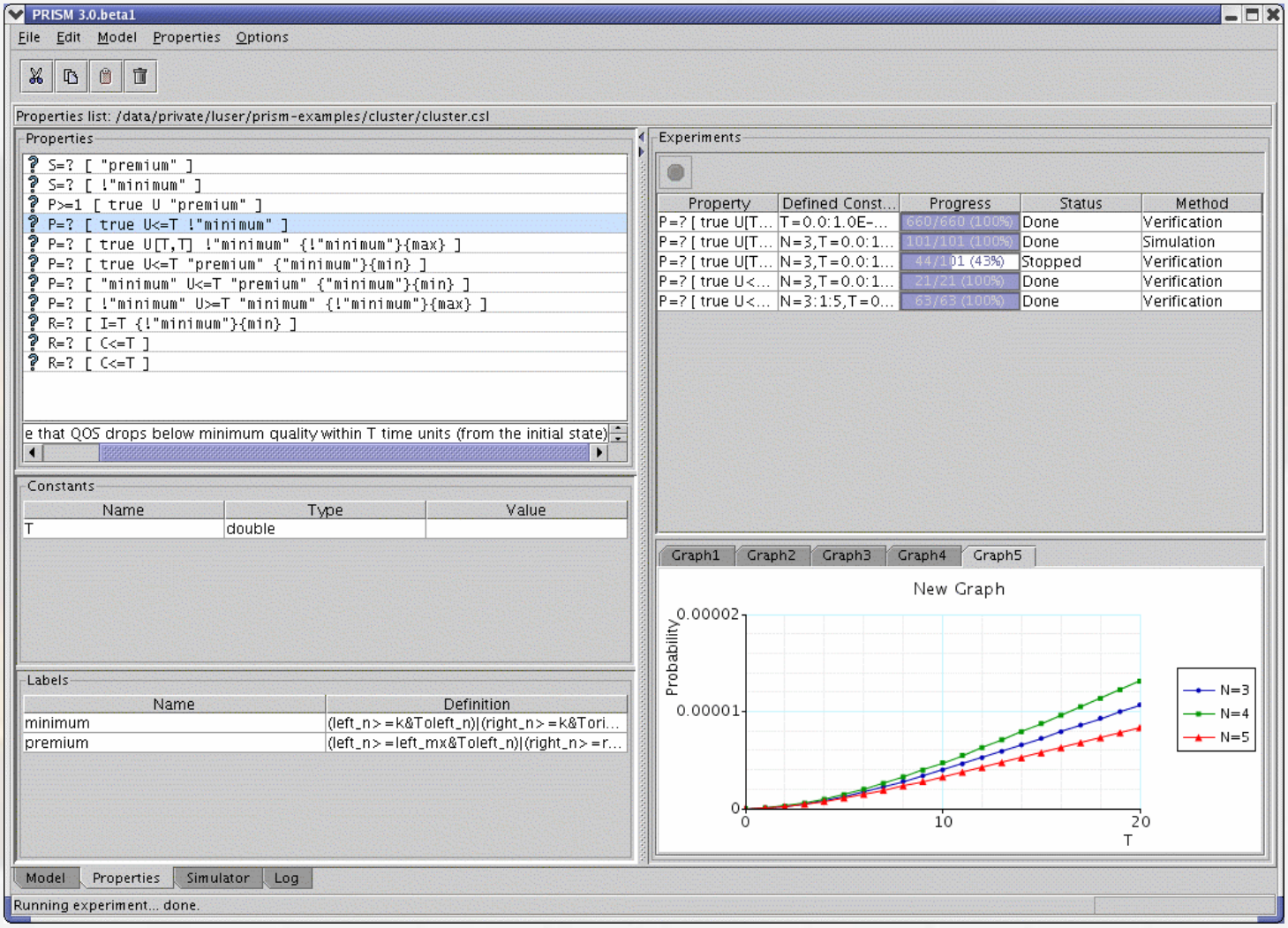
The PRISM probabilistic model checker

- Functionality
  - Modelling language: probabilistic reactive modules
  - Construction of models: DTMCs/CTMCs, MDPs
  - Probabilistic temporal logics: PCTL and CSL
  - Extension with costs/rewards, expectation operator
- Underlying computation involves
  - Reachability, qualitative model checking, BDD-based
  - Linear equation system solution - Jacobi, Gauss-Seidel, ...
  - Uniformisation (CTMCs)
  - Dynamic programming (MDPs)
  - Explicit and symbolic (MTBDDs, etc.)
- Similar tools
  - ETMCC/MRMC, PROBMELA, Vesta, Rapture, Ymer, APMC, APNN-Toolbox, SMART, Mobius

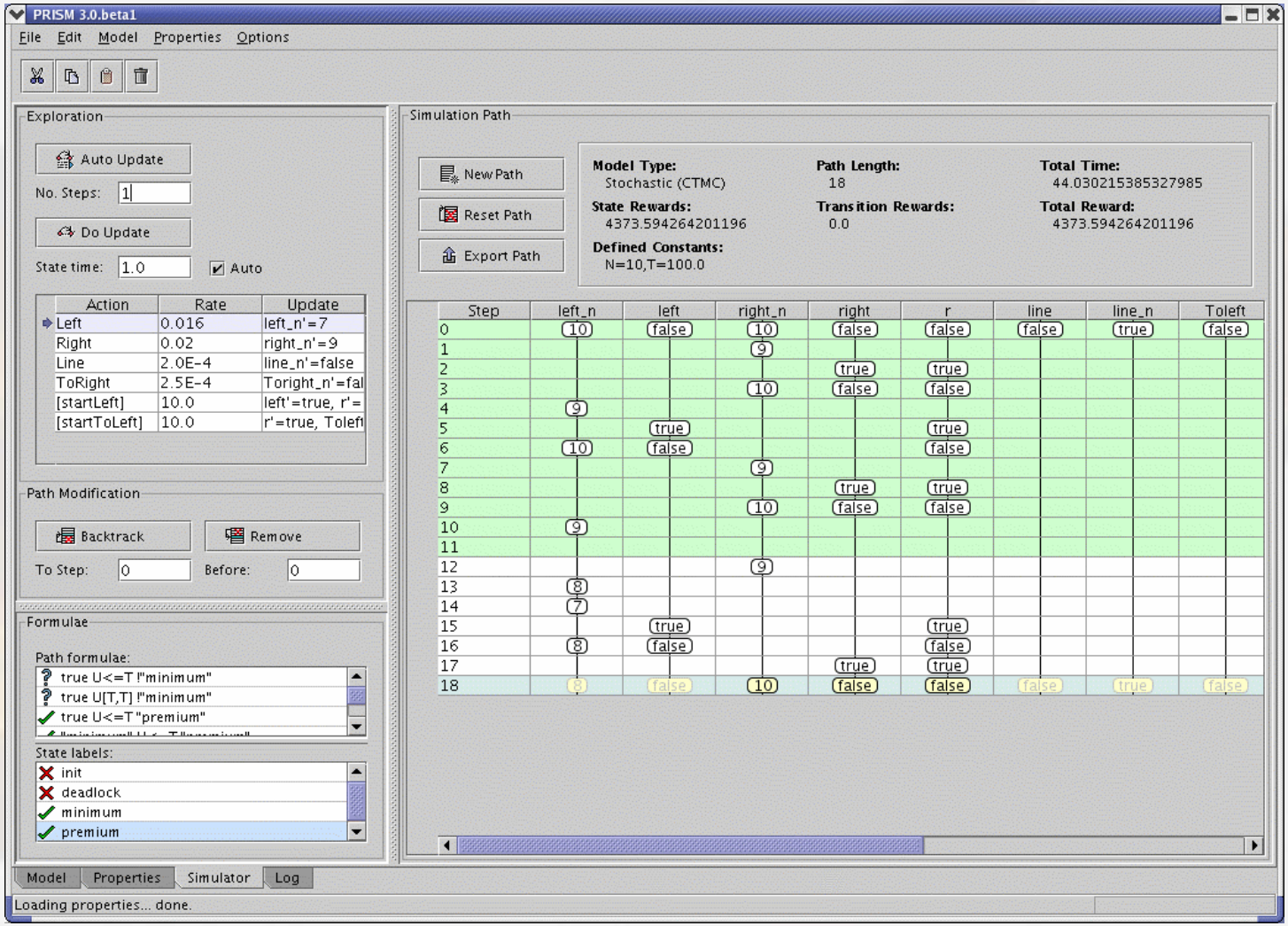
PRISM Screenshots



PRISM Screenshots



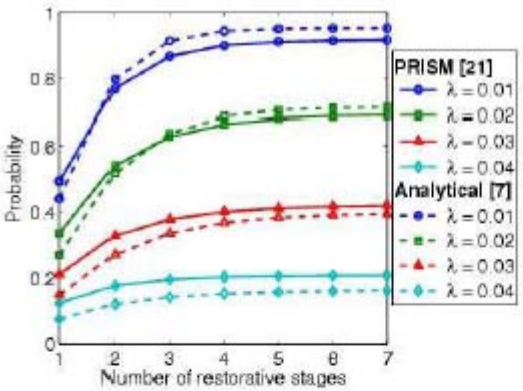
PRISM Screenshots



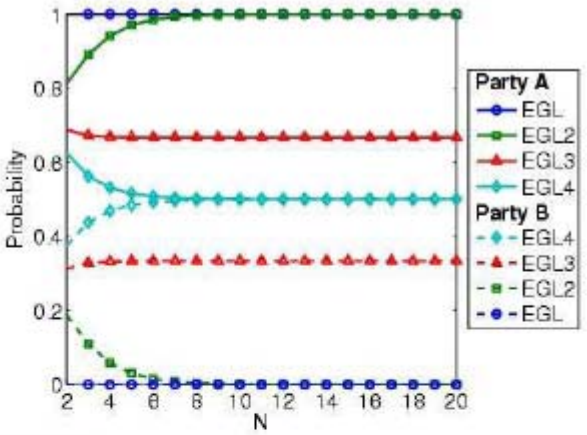
Probabilistic specifications

- As in probabilistic extensions of temporal logic
  - incorporates PCTL for DTMCs/MDPs, CSL for CTMCs
- Examples:
  - $P<0.001$ [F shutdown] - "shutdown eventually occurs with probability at most 0.001"
  - $P<0.2$ [$F[t,t]$ (deliv_rate < min)] "the probability that the current packet delivery rate has dropped below minimum at time t is less than 0.2"
  - $P\geq 0.95$ [!repair $U^{\leq 200}$ done] - "with probability 0.95 or greater, the process will successfully complete within 200 hours and without requiring any repairs"
- No counterexamples (yet) in prob. model checking

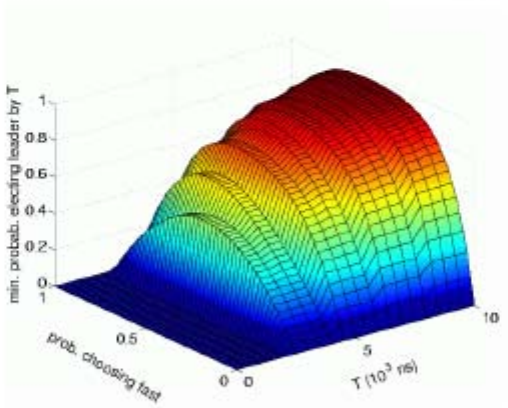
Results: probabilistic analysis



Probability that 10% of gate outputs are erroneous for varying gate failure rates and numbers of stages



Probability that parties gain unfair advantage for varying numbers of secret packets sent



Optimum probability of leader election by time T for various coin biases

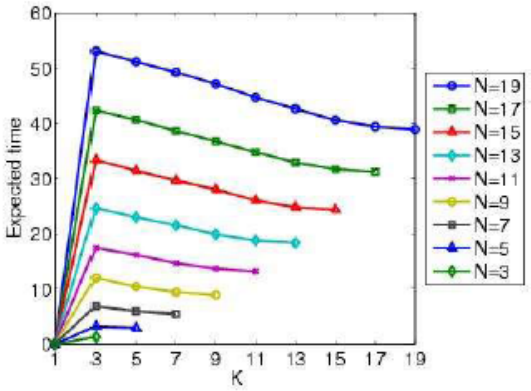
Quantitative specifications

- Focus on quantitative properties, compute actual values
  - P=? [F≤T "shutdown"] - "what is the probability of shutdown occurring within T hours?"
- Best/worst-case scenarios
  - P=? [F "error" {"init"}{max}] - "what is the worst-case error probability over all possible initial configurations?"
  - Pmin=? [!end2 U end1] - "what is the minimum probability of process 1 finishing before process 2, over all possible schedulings of the processes?"
- Experiments – ranges of model/property parameters
  - P=? [F≤T error] for N=1..5, T=1..100
  - identify patterns, trends, anomalies in results

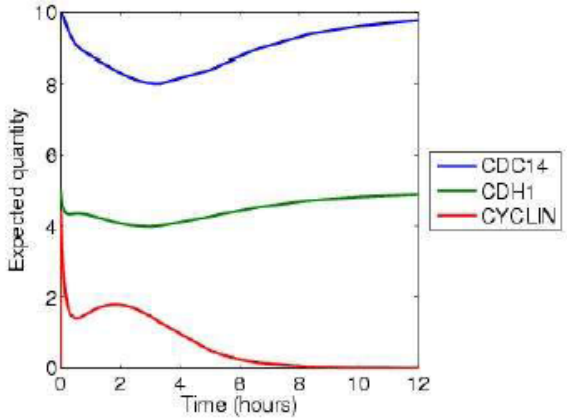
Cost- and reward-based properties

- Costs and rewards
  - real-valued quantities assigned to states/transitions
- Instantaneous – state-based measures
  - current queue size, number of operational channels, ...
  - "what is the expected size of the message queue at time t?"
  - "what is the long-run expected size of the queue?"
- Cumulative – state or transition (impulse) costs/rewards
  - time, power consumption, messages lost, ...
  - "what is the expected power consumption during the first 2 hours of operation?"
  - "what is the worst-case expected time taken for the protocol to terminate?"

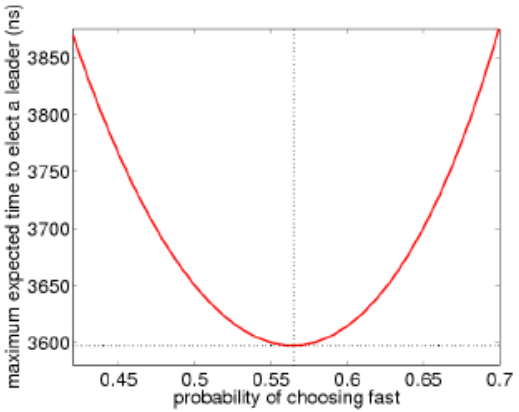
Results: quantitative analysis



Worst-case expected number of steps to stabilise for initial configurations with K tokens amongst N processes



Expected reactant concentrations over the first 12 hours



Maximum expected time for leader election for various coin biases

Efficiency: Symbolic techniques

- State space explosion
  - models of real-life systems typically huge
- Symbolic probabilistic model checking
  - data structures based on binary decision diagrams (BDDs)
  - compact storage: exploit model structure and regularity
  - efficient implementation of graph traversal fixed point algorithms
- PRISM: multiple computation engines
  - MTBDDs (BDD extension): storage/analysis of very large models (given structure/regularity), numerical computation can blow up
  - sparse matrices: fastest solution for smaller models (<10^6 states), prohibitive memory consumption for larger models
  - hybrid: combine MTBDD storage with explicit storage, ten-fold increase in analysable model size (~10^7 states)

Efficiency: Other strategies

- Approximate model checking (see also APMC [LHP06])
  - sampling using Monte Carlo discrete-event simulation
  - performed at modelling language level – better scalability
  - potentially huge number of samples for accurate answers
  - also: statistical hypothesis testing, see e.g. [YS02]
- Parallelisation of model checking
  - distribution of storage/computation across multi-processor machines [KPZM04], networked clusters [ZPK05], grids
  - potentially promising for symbolic approaches – reduced I/O
  - simulation-based computations much easier to distribute

Ongoing research areas

- Abstraction and refinement, see e.g. [DJJL01,KNP06a]
  - construct smaller, abstract model by removing information/variables not relevant to property being checked, iteratively refine abstraction if analysis fails
- Symmetry reduction [DM06, KNP06b]
  - exploit replication of identical components
- Partial order reduction, see e.g. [BGC04], [DN04]
  - exploit commutativity of concurrently executed transitions
- Compositionality, see e.g. [dAHJ01,Che06]
  - analyse full model based on analysis of sub-components

PRISM real-world case studies

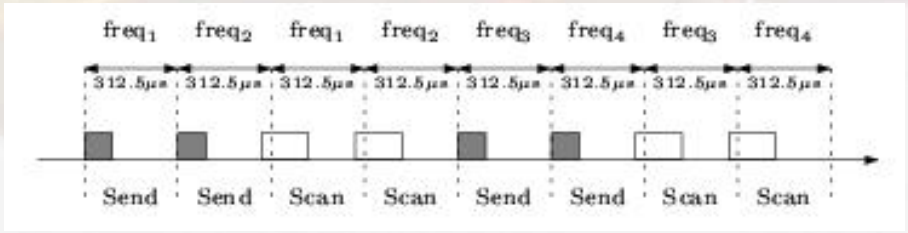
- Network and coordination protocols
  - Bluetooth device discovery [ISOLA'04, STTT'06]
  - IEEE 802.15.4 (Zigbee) [ISOLA'06]
  - IPv4 Zeroconf dynamic configuration [FORMATS'03, FMSD'06]
  - Randomised consensus [CAV'01]
  - Power Management (with Shukla & Gupta) [HLDVT'02, FAC'05]
- Security protocols
  - Crowds anonymity protocol (by Shmatikov) [JCS'04]
  - Probabilistic contract signing (with Shmatikov) [JCS'04]
- Reliability
  - NAND multiplexing for nanotech (with Shukla) [VLSI'04, TCAD'05]
  - Dependability of embedded controller [INCOM'04]
- Biology
  - FGF signalling pathway [CMSB'06]

Bluetooth device discovery

- Bluetooth: short-range low-power wireless protocol
  - widely available in phones, PDAs, laptops, ...
  - personal area networks (PANs)
  - open standard, specification freely available
- Uses frequency hopping scheme
  - to avoid interference (uses unregulated 2.4GHz band)
  - pseudo-random selection over 32 of 79 frequencies
- Network formation
  - piconets (1 master, up to 7 slaves)
  - self-configuring: devices discover themselves
- Device discovery
  - manadatory first step before any communication possible
  - performance crucial to reduce power consumption

Master (sender) behaviour

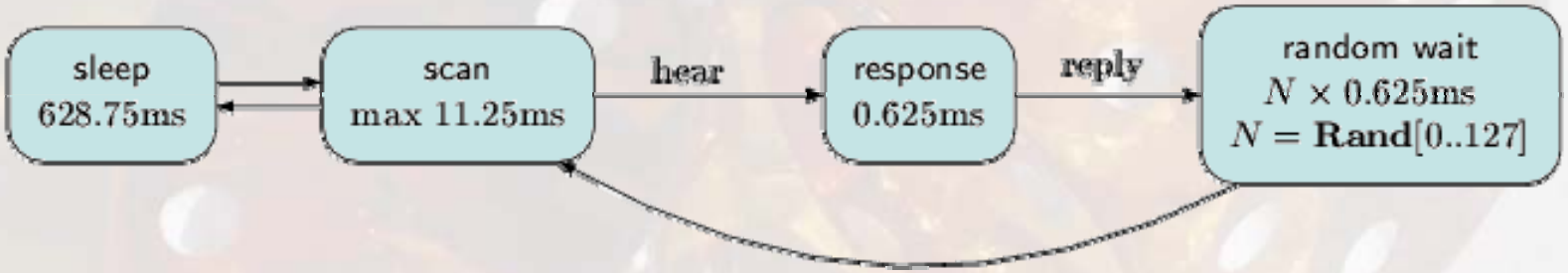
- 28 bit free-running clock CLK, ticks every 312.5µs
- Frequency hopping sequence determined by clock:
  - freq = [CLK_{16-12}+k+ ($CLK_{4-2,0}$- CLK_{16-12}) mod 16] mod 32
  - 2 trains of 16 frequencies (determined by offset k), 128 times each, swap between every 2.56s
- Broadcasts inquiry packets on two consecutive frequencies, then listens on the same two



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	16

Slave (receiver) behaviour

- Listens (scans) on frequencies for inquiry packets
  - must listen on right frequency at right time
  - cycles through frequency sequence at much slower speed (every 1.28s)



- On hearing packet, pause, send reply and then wait for a random delay before listening for subsequent packets
  - avoid repeated collisions with other slaves

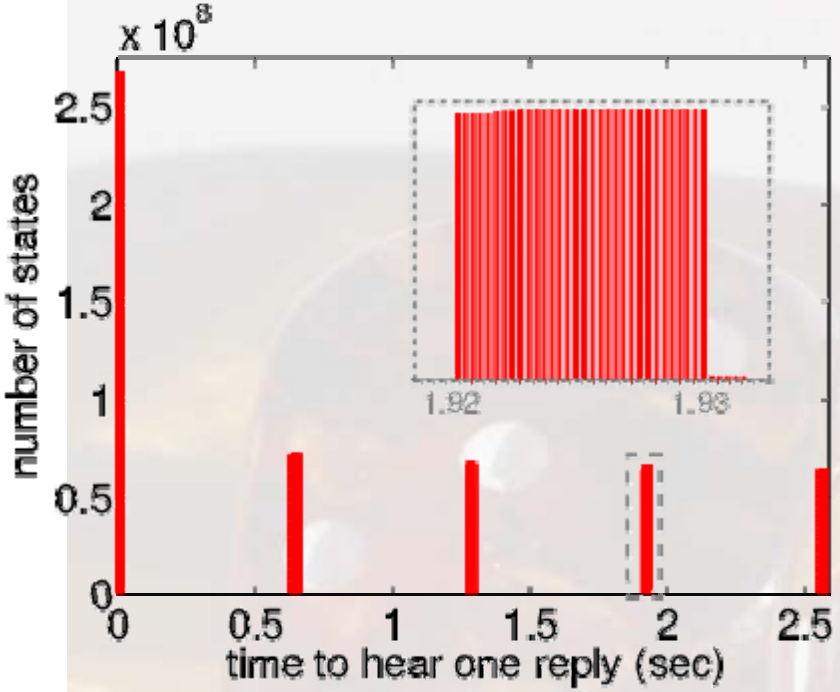
Bluetooth – PRISM model

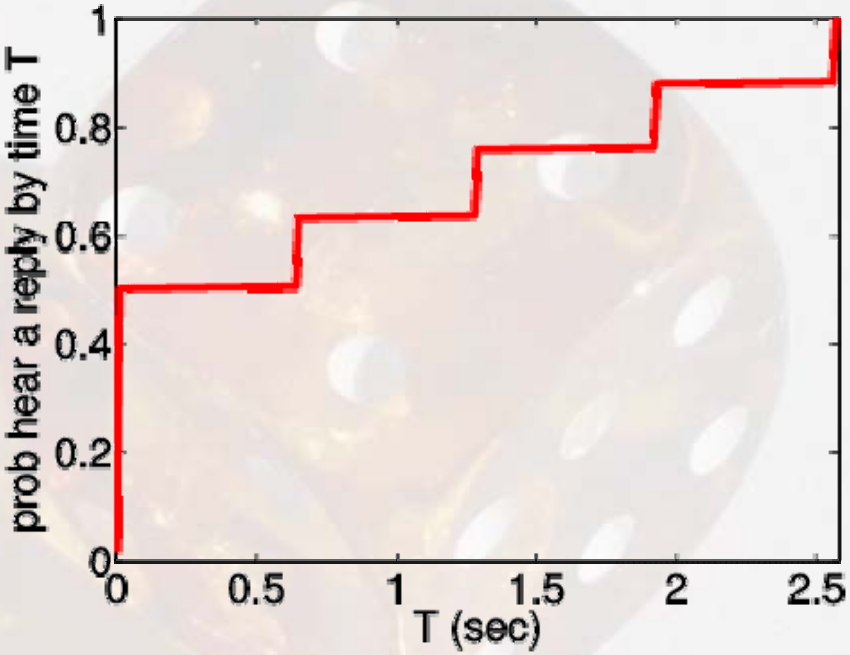
- Modelling in PRISM [DKNP06]
  - model one sender and one receiver
  - synchronous (clock speed defined by Bluetooth spec)
  - randomised behaviour – use DTMC
  - model at lowest-level (one clock-tick = one transition)
  - use real values for delays, etc. from Bluetooth spec
- Modelling challenges
  - complex interaction between sender/receiver
  - combination of short/long time-scales – cannot scale down
  - sender/receiver not initially synchronised, huge number of possible initial configurations (17,179,869,184)

Bluetooth - Results

- Huge DTMC – initially, model checking infeasible
  - partition into 32 scenarios, i.e. 32 separate DTMCs
  - on average, approx. 3.4×10^9 states, 536,870,912 initial
  - can be built/analysed with PRISM's MTBDD engine
- Compute for the first time:
  - R=? [F replies=K {"init"}{max}]
  - "worst-case expected time to hear K replies over all possible initial configurations"
  - also look at:
    - how many initial states for each possible expected time
    - cumulative distribution function assuming equal probability for each initial state

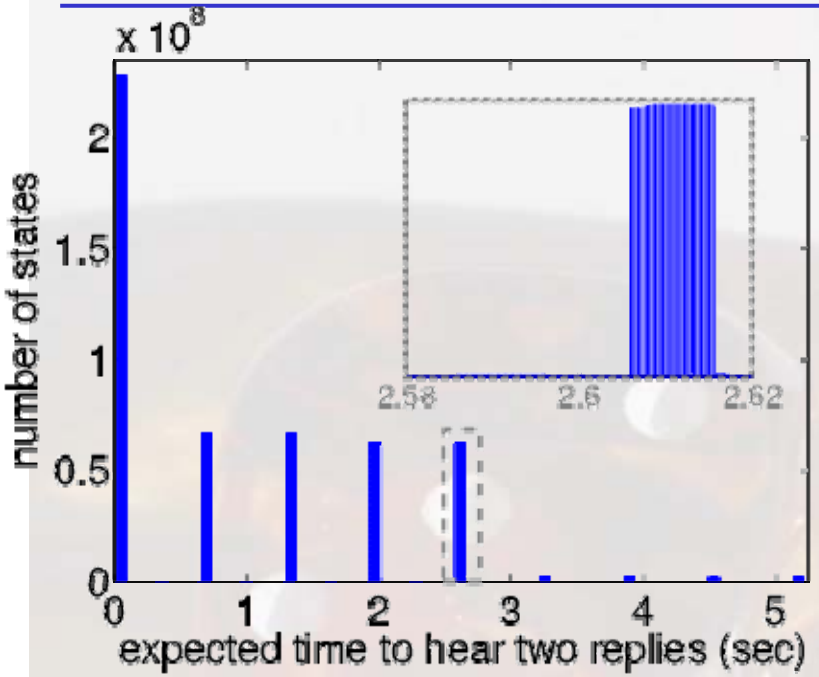
Bluetooth - Time to hear 1 reply

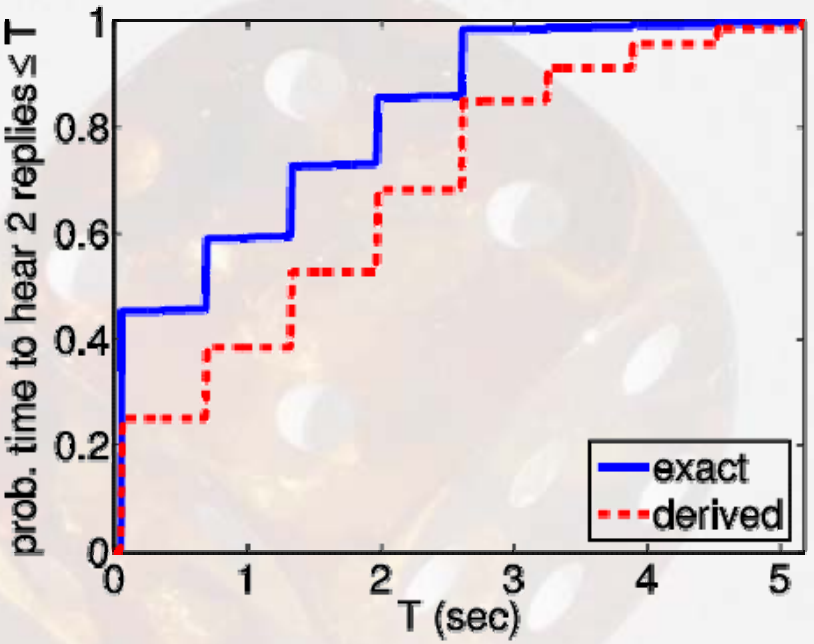




- worst-case expected time = 2.5716 sec
- in 921,600 possible initial states
- best-case = 635 μs

Bluetooth - Time to hear 2 replies





- worst-case expected time = 5.177 sec
- in 444 possible initial states
- compare actual CDF with derived version which assumes times to reply to first/second messages are independent

Related projects

- Modelling and verification of mobile ad hoc network protocols
  - Modelling language with mobility and randomisation
  - Model checking algorithms & techniques
  - Formal methods for power management in PANs
- Ubival (forthcoming)
  - Verification via model checking, testing and simulation of ubiquitous software
  - Focus on mobility, context-awareness and adaptiveness
  - Apply probabilistic model checking to quantify the likelihood of context
- Probabilistic model checking of biological processes
  - Using PRISM and CTMC models to analyse signalling networks

Ongoing work

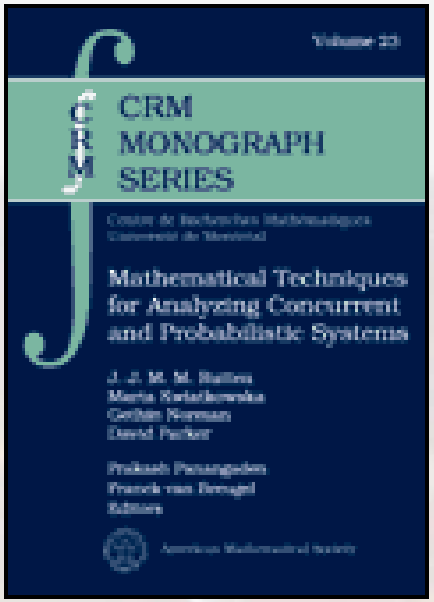
- Exploiting structure, model reductions
  - Abstraction, employing games-based approaches, [QEST'06]
  - Symmetry reduction, [CAV'06]
  - Partial order reduction, (with Baier et al) [FST&TCS'06]
- Extension for mobility
  - Extension for ambients and spatial logic (with Vigliotti)
  - Combine with techniques for pi-calculus (with Peng Wu)
  - Applications in systems biology
- Multi-criteria optimisation
  - For MDPs (with Etessami et al)
- Real software, not models! E.g. verify source code containing

```
x := random(1..65024); {ZeroConf}
```

Challenges for future

- Scalability!
  - Abstraction/refinement
  - Compositionality, e.g. assume-guarantee
- Proof assistant for probabilistic verification
  - Verify for all scenarios, parametric analysis
- Efficiency
  - Approximation methods
  - Statistical analysis
  - Parallelisation
- Extension for hybrid systems
  - E.g. context-awareness for geographical coordinates

For more information…



J. Rutten, M. Kwiatkowska, G. Norman and D. Parker
Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems
P. Panangaden and F. van Breugel (editors), CRM Monograph Series, vol. 23, AMS
March 2004

www.cs.bham.ac.uk/~dxp/prism/

- Case studies, statistics, group publications
- Download, version 3.1 (4700+ downloads)
- Unix/Linux, Windows, Apple platforms
- Publications by others and courses that feature PRISM…

PRISM Contributors