PRISM - A Tutorial

Dave Parker

University of Birmingham

November 2006
IPA Herfstdagen on Stochastic Systems

Overview

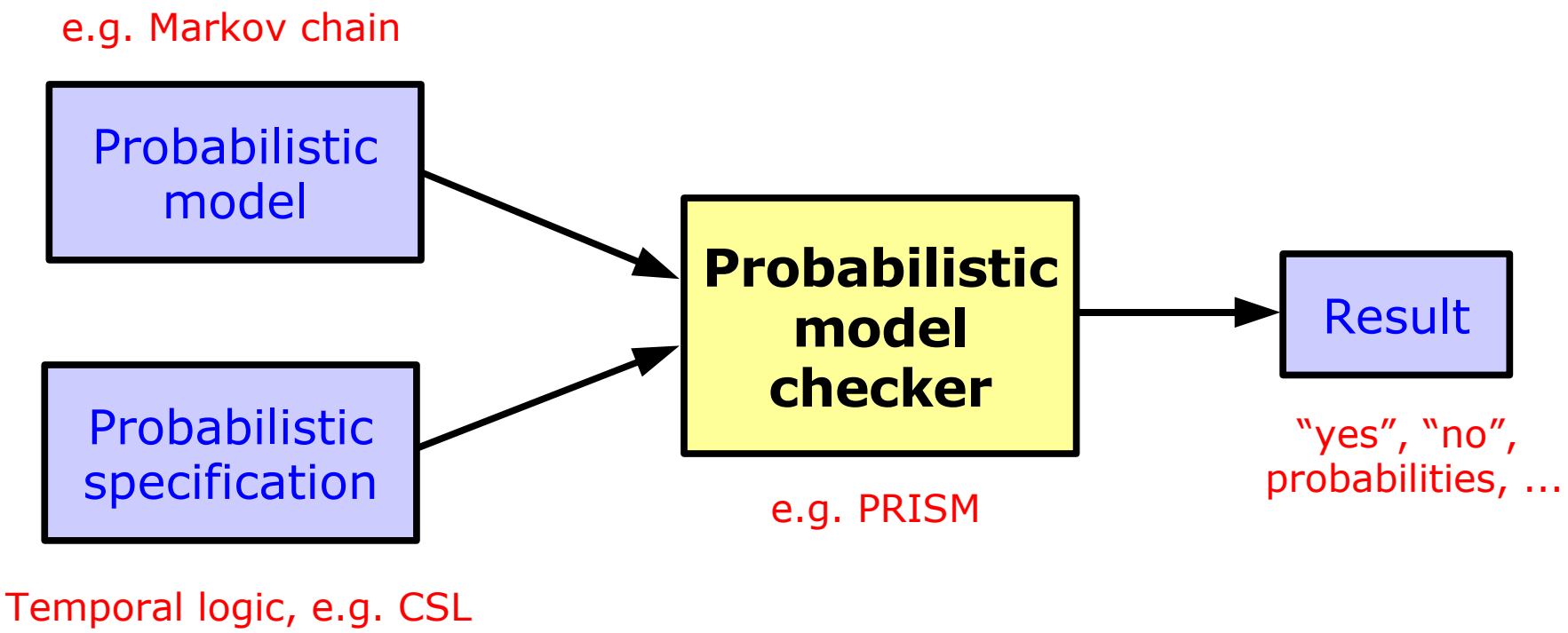
- I. Probabilistic model checking
  - overview, models, tools
- II. The PRISM tool
  - functionality, features, resources
  - the PRISM modelling language
  - property specification
  - tool demo
  - efficiency issues + research topics
- III. PRISM case studies

Part I

Probabilistic Model Checking

Probabilistic model checking

- Automatic formal verification of probabilistic systems
  - exhaustive exploration/analysis of model



Probabilistic Models

- Discrete-time Markov chains (DTMCs)
  - time modelled in discrete steps; discrete probabilities
- Continuous-time Markov chains (CTMCs)
  - continuous (real-valued) model of time: exponential distributions
- Markov decision processes (MDPs)
  - discrete time-steps, discrete probabilities, and nondeterminism
  - e.g. parallel composition of concurrent stochastic processes
- Probabilistic timed automata (PTAs)
  - discrete probabilities, nondeterminism, real-time clocks
  - in some cases, can transform to MDPs ("digital clocks")

Probabilistic model checking

+ wide range of quantitative measures
+ exact answers computed
+ fully automatic process
  - model construction + numerical solution
– difficult to generate good counterexamples
  - but can identify patterns, trends, anomalies in quantitative results
+ exhaustive analysis, good for 'corner cases', e.g.
  - all possible initial configurations/model parameter values
  - all possible process schedulings (nondeterminism)
– state space explosion: time and memory constraints
+ efficient algorithms, implementations, tool support

Probabilistic model checking tools

- PRISM - DTMCs, CTMCs, MDPs - this talk...
- ETMCC/MRMC – DTMCs, CTMCs + reward extensions
- LiQuor – MDPs (ProbMela language), LTL (and other automata-based specifications)
- Simulation-based probabilistic model checking:
  - APMC, Ymer (both based on PRISM language)
- CSL model checking for CTMCs: APNN-Toolbox, SMART
- Multiple formalism/tool solutions: CADP, Möbius
- RAPTURE - prototype for abstraction/refinement of MDPs

Part II

The PRISM Tool

The PRISM tool

- PRISM: Probabilistic symbolic model checker
  - developed at the University of Birmingham, since approx. 1999
  - free, open source (GPL)
  - versions for Linux, Unix, Mac OS X, Windows, (64-bit coming soon)

- Modelling of: DTMCs, MDPs , CTMCs + costs/rewards
- Verification of: PCTL, CSL + extensions + costs/rewards
- Features: high-level modelling language, wide range of model analysis methods, graphical user interface, efficient implementation
- www.cs.bham.ac.uk/~dxp/prism

Getting PRISM + Other Resources

- PRISM website: www.cs.bham.ac.uk/~dxp/prism
  - tool download: binaries, source code (GPL)
  - on-line example repository (40+ case studies)
  - on-line documentation:
    - PRISM manual
    - PRISM tutorial
  - support: help forum, bug tracking, feature requests
    - hosted on Sourceforge
  - related publications, talks, tutorials, links

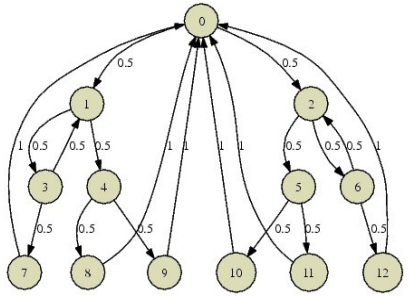
PRISM – Model building

- First step of verification = construct full probabilistic model
  (not always necessary in non-probabilistic model checking)

High-level model

DTMC, CTMC, MDP

```
// Hermans self-stabilisation algorithm [Her90]

dtmc // Algorithm is fully synchronous

module process1 // first of N=5 symmetric processes

    x1 : [0..1];  // one bit per process; xi=x(i-1) means proc i
    [step] (x1=x5) -> 0.5 : (x1'=0) + 0.5 : (x1'=1);
    [step] !x1=x5 -> (x1'=x5);

endmodule

// Add further processes through renaming
module process2 = process1 [ x1=x2, x5=x1 ] endmodule
module process3 = process1 [ x1=x3, x5=x2 ] endmodule
module process4 = process1 [ x1=x4, x5=x3 ] endmodule
module process5 = process1 [ x1=x5, x5=x4 ] endmodule

// Can start in any possible configuration
init true endinit
```



(PRISM language)

(matrix, MTBDD, ...)

PRISM – Imports and exports

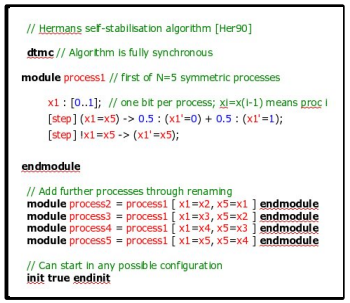
- Support for connections to other formats/tools:

Imports:

PEPA

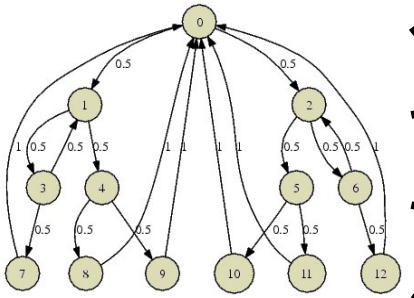
Text

High-level model



(PRISM language)

DTMC, CTMC, MDP



(matrix, MTBDD, ...)

Exports:

Text

Matlab

MRMC

Dot

In progress: probabilistic CSP, pi calculus, SBML, ProbMela, ...

PRISM modelling language

- Simple, state-based language for DTMCs/MDPs/CTMCs
  - based on Reactive Modules [Alur/Henzinger]
- Modules (system components, composed in parallel)
- Variables (finite-valued, local or global)
- Guarded commands (labelled with probabilities/rates)
- Synchronisation (CSP-style) + process-algebraic operators (parallel composition, action hiding/renaming)

[send] (s=2) -> p_{loss} : (s'=3)&(lost'=lost+1) + (1-p_{loss}) : (s'=4);

PRISM language example

```
// Herman's self-stabilisation algorithm [Her90]

dtmc // Algorithm is fully synchronous

module process1 // First of N=5 symmetric processes

    x1 : [0..1];  // One bit per process; xi=x(i-1) means proc i has a token
    [step] (x1=x5) -> 0.5 : (x1'=0) + 0.5 : (x1'=1);
    [step] !x1=x5 -> (x1'=x5);

endmodule

// Add further processes through renaming
module process2 = process1 [ x1=x2, x5=x1 ] endmodule
module process3 = process1 [ x1=x3, x5=x2 ] endmodule
module process4 = process1 [ x1=x4, x5=x3 ] endmodule
module process5 = process1 [ x1=x5, x5=x4 ] endmodule

// Can start in any possible configuration
init true endinit
```

PRISM language example 2 (fragment)

```
// Embedded control system
ctmc
const int MIN_SENSORS = 2;
const double lambda_p = 1/(365*24*60*60); // MTTF = 1 year
...
module sensors
	s : [0..3] init 3; // Number of sensors working
	[] s>1 -> s*lambda_s : (s'=s-1); // Failure of a single sensor
endmodule

module proci  // (takes data from sensors and passes onto main processor)
	i : [0..2] init 2; // 2=ok, 1=transient fault, 0=failed
	[] i>0 & s>=MIN_SENSORS -> lambda_p : (i'=0); // Failure of processor
	[] i=2 & s>=MIN_SENSORS -> delta_f : (i'=1); // Transient fault
	[reboot] i=1 & s>=MIN_SENSORS -> delta_r : (i'=2); // Transient reboot
endmodule
```

Costs and rewards

- Real-valued quantities assigned to model states/transitions
  - many possible uses, e.g. time, power consumption, current queue size, number of messages lost, ...
- No distinction between costs ("bad") and rewards ("good")
  - PRISM terminology is rewards
- The meaning of these rewards varies depending on:
  - the model type: for CTMCs, a state reward may indicate the rate at which reward is accumulated in that state
  - the type of property used to analyse the model: instantaneous or cumulative

Rewards in the PRISM language

```
rewards "total_queue_size"
    true : queue1+queue2;
endrewards
```

(instantaneous, state-based)

```
rewards "time"
    true : 1;
endrewards
```

(cumulative, state-based)

```
rewards "power"
    sleep=true : 0.25;
    sleep=false : 1.2 * up;
endrewards
```

(cumulative, state-based)
(up = number of operational components)

```
rewards "dropped"
    [receive] q=q_max : 1;
endrewards
```

(cumulative, transition-based)
(q = queue size, q_max = max queue size)

PRISM property specifications

- Based on (probabilistic extensions of) temporal logic
  - incorporates PCTL for DTMCs/MDPs, CSL for CTMCs
  - also includes: quantitative extensions, costs/rewards
- Simple PCTL/CSL example:
  - P<0.001 [true U shutdown] - "the system eventually shuts down **with probability at most 0.001**"
- Usually focus on quantitative properties:
  - P=? [true U shutdown] - "**what is the probability** that the system eventually shuts down**?**"
  - Nested probabilistic operators must be bounded

Basic types of property specifications

- (Unbounded) reachability:
  - P=? [true U shutdown] - "probability of **eventual** shutdown"
- Transient/time-bounded properties:
  - P=? [true U[t,t] (deliv_rate < min)] - "probability that the packet delivery rate has dropped below minimum **at time t**"
  - P=? [!repair U≤200 done] - "probability of the process completing **within 200 hours** and without requiring repairs"
- Steady-state properties:
  - S=? [num_sensors ≥ min] - "**long-run** probability that an adequate number of sensors are operational"

Cost- and reward-based properties

- Two different interpretations of model rewards
  - instantaneous and cumulative properties
  - reason about expected values of rewards
- Instantaneous reward properties
  - state rewards only
  - state-based measures: “queue size”, “number of operational channels”, “concentration of reactant X”, ...
- R=? [I=t]
  - e.g. “**expected size** of the message queue at time t?”
- R=? [S]
  - e.g. “**long-run expected size** of the queue?”

Cost- and reward-based properties

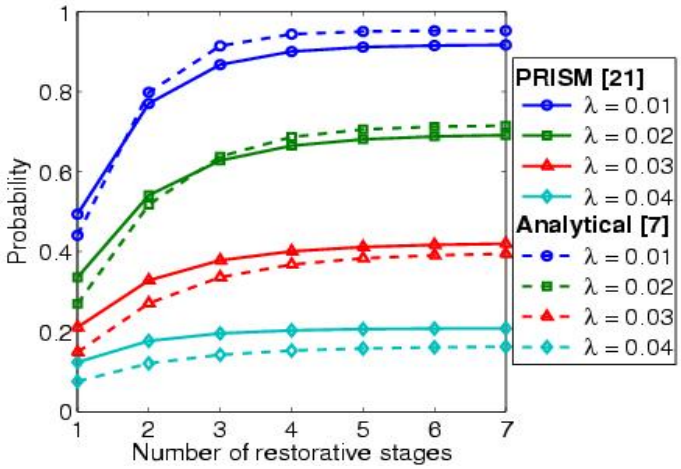
- Cumulative reward properties
  - both state and transition (impulse) rewards cumulated
  - CTMC state rewards interpreted as reward rates
  - e.g. “time”, “power consumption”, “number of messages lost”
- R=? [F end]
  - e.g. “**expected time** taken for the protocol to terminate?”
- R=? [C≤2]
  - e.g. “**expected power consumption** during the first 2 hours that the system is in operation?”
  - e.g. “**expected number of messages** lost during...”

Best/worst-case scenarios

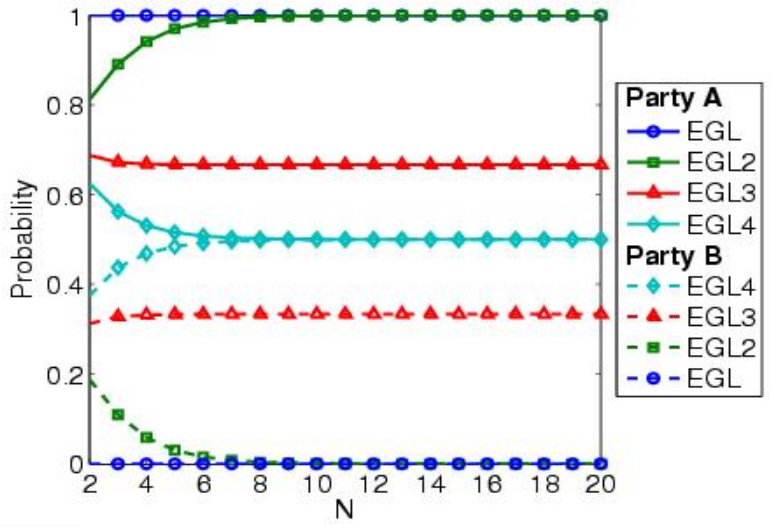
- Combining "quantitative" and "exhaustive" aspects
- Computing values for a range of states
  - R=? [F end {"init"}{max}] - "**maximum** expected run-time over all possible initial configurations?"
  - P=? [true U≤t elected {tokens≤k}{min}] - "**minimum** probability of the leader election algorithm completing within t steps from any state where there are at most k tokens?"
- All possible resolutions of nondeterminism (MDPs)
  - Pmin=? [!end2 U end1] - "**minimum** probability of process 1 finishing before process 2, for any scheduling of processes?"
  - Rmax=? [F end] - "**maximum** expected number of bits revealed revealed under any eavesdropping strategy?"

Identifying trends and anomalies

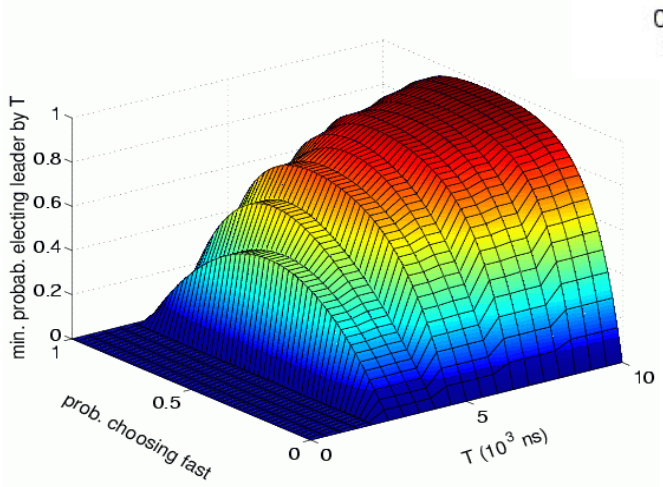
- Counterexamples (error traces)
  - widely used in non-probabilistic model checking
  - situation much less clear in probabilistic model checking
  - counterexample for P<p [true U error] ? And for P=? [...] ?
  - work in progress...
- Experiments: ranges of model/property parameters
  - e.g. P=? [true U≤T error] for N=1..5, T=1..100
    - where N is some model parameter and T a time bound
  - identify patterns, trends, anomalies in quantitative results



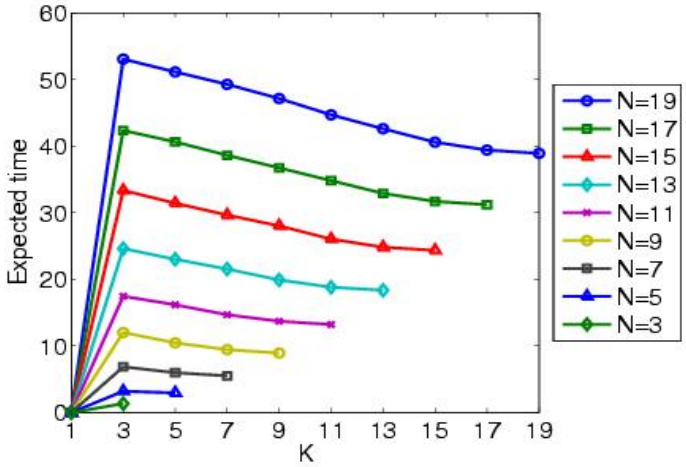
Probability that 10% of gate outputs are erroneous for varying gate failure rates and numbers of stages



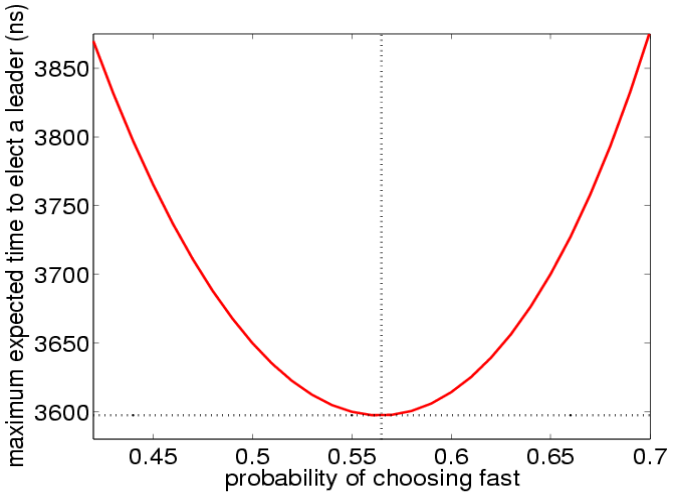
Probability that parties gain unfair advantage for varying numbers of secret packets sent



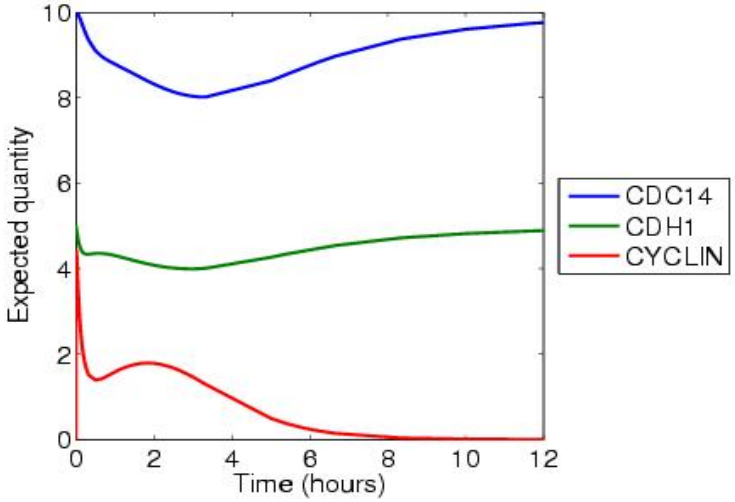
Optimum probability of leader election by time T for various coin biases



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



Maximum expected time for leader election for various coin biases



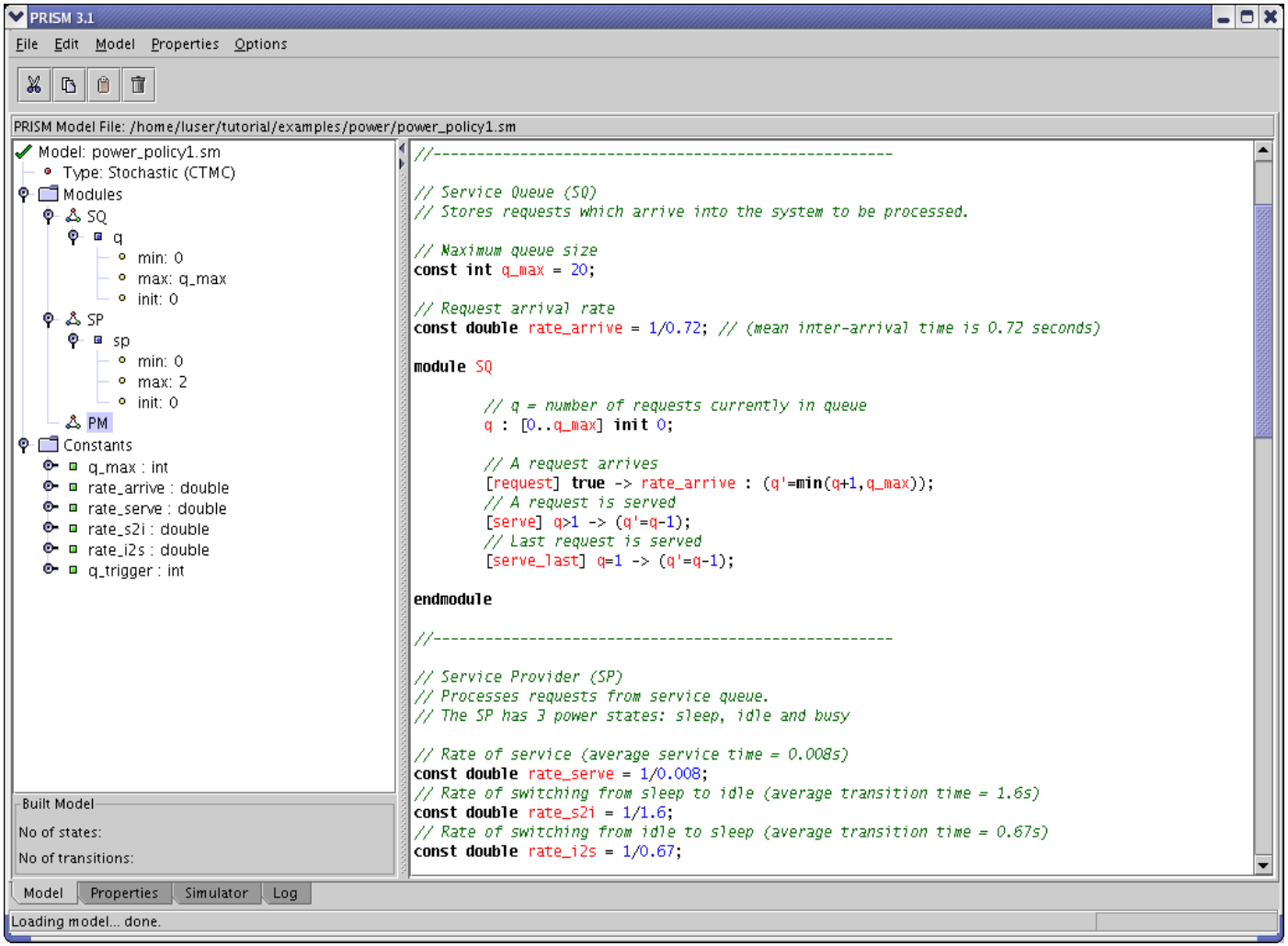
Expected reactant concentrations over the first 12 hours

PRISM functionality

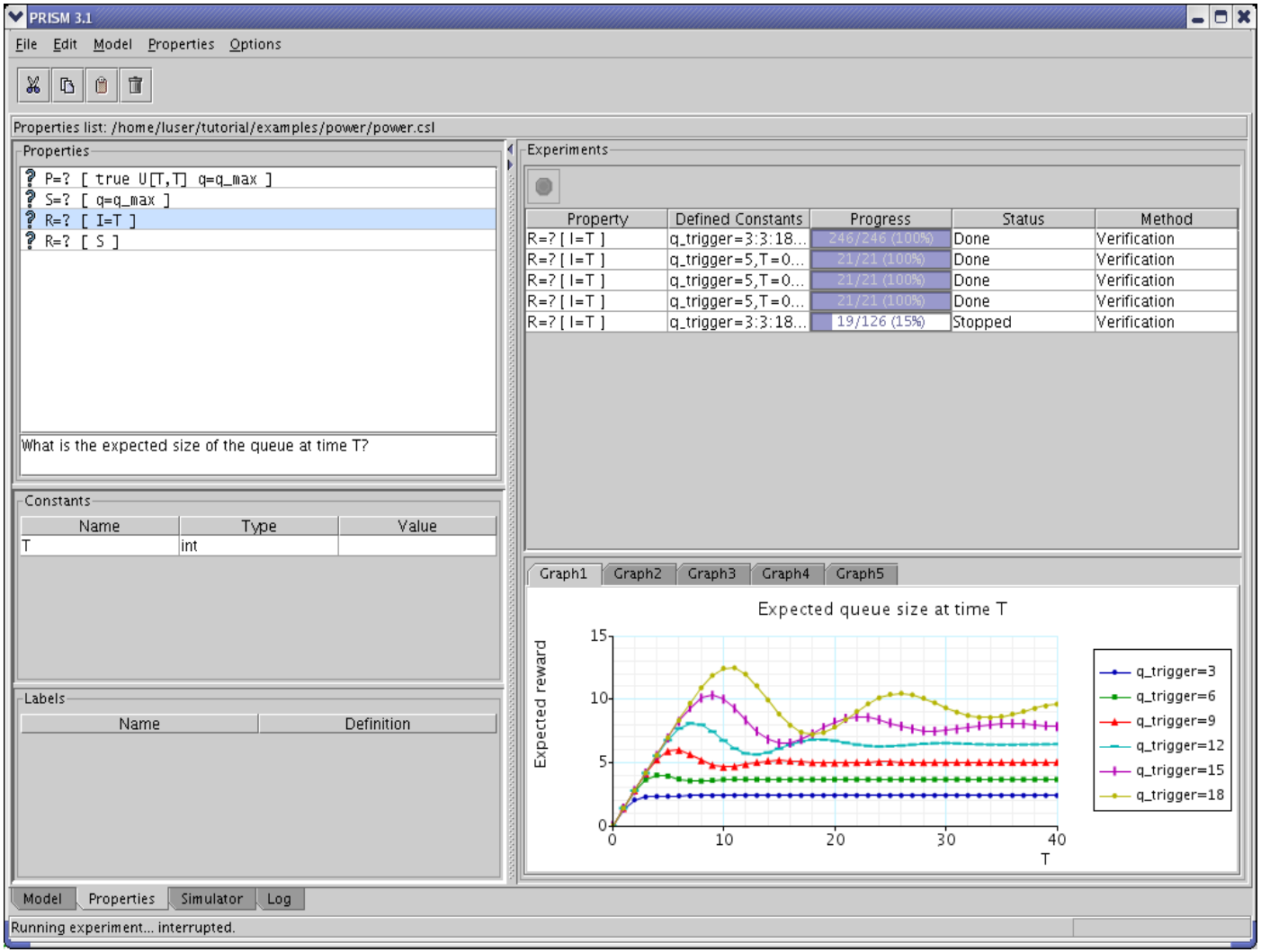
- Graphical user interface
  - model/property editor
  - discrete-event simulator - model traces for debugging, etc.
  - verification of PCTL, CSL + costs/rewards, etc.
  - approximate verification using simulation + sampling
  - easy automation of verification experiments
  - graphical visualisation of results
- Command-line version
  - same underlying verification engines
  - useful for scripting, batch jobs

PRISM demo

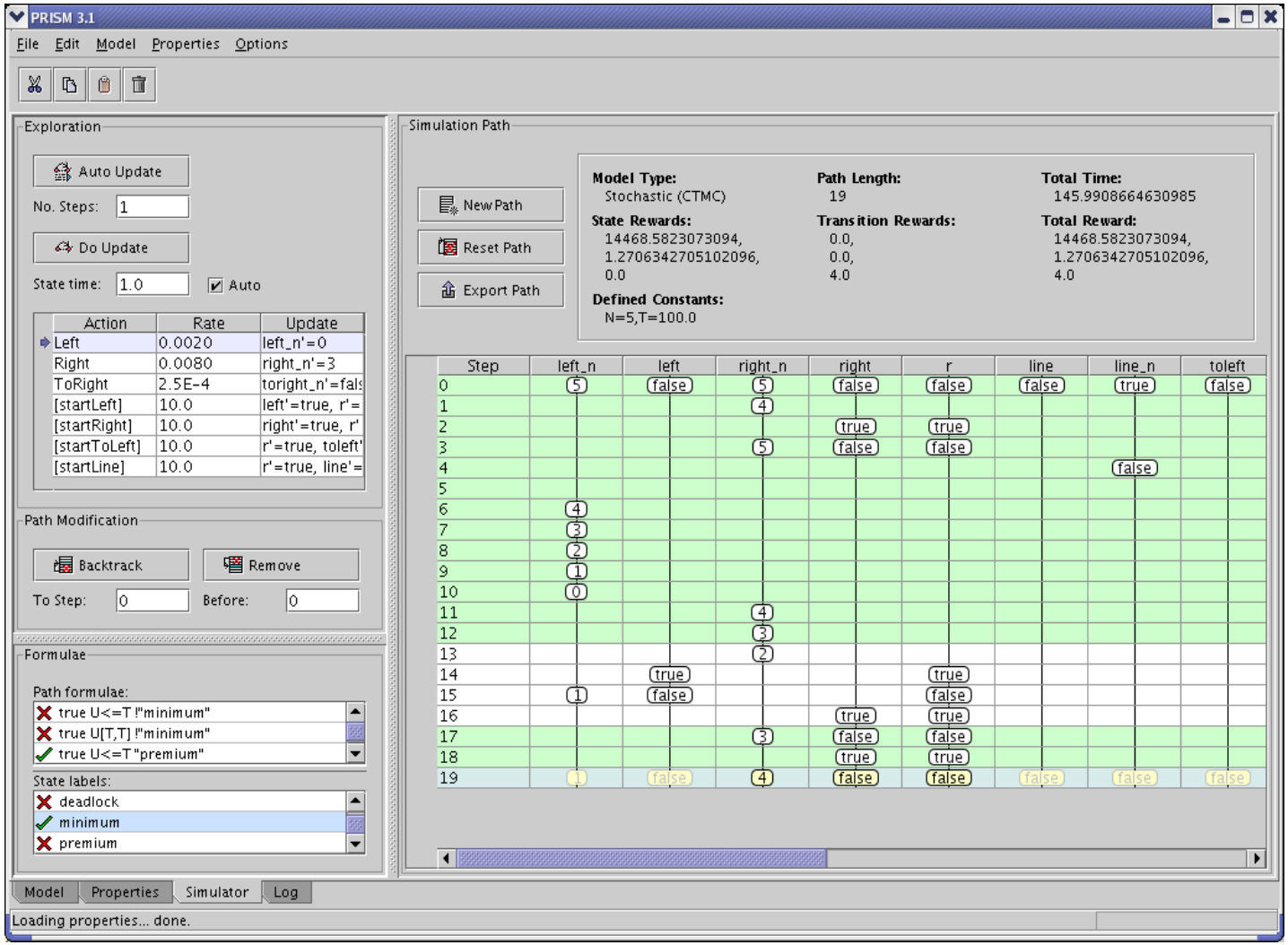
PRISM screenshots



PRISM screenshots



PRISM screenshots



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 numerical 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 ($\sim 10^7$ states)

Efficiency – Approximate verification

- Approximate probabilistic model checking
  - sampling using Monte Carlo discrete-event simulation
  - performed at modelling language level – better scalability
  - more easily extended to a wider range of properties
  - potentially huge number of samples for accurate answers
- Tool support:
  - APMC [LHP06] – PCTL/LTL for D/CTMCs, distributed version
  - Also supported in PRISM (distributed version coming soon)
- Statistical hypothesis testing, acceptance sampling
  - "bounded" properties, e.g. P<p[...]. See e.g. Ymer [YS02]

Efficiency - Parellelisation

- Parallelisation of probabilistic model checking
  - distribution of storage/computation costs
  - ease of distribution depends on particular computation task
    - reachability? numerical computation?
  - potentially promising for symbolic approaches – reduced I/O
    - steady-state/transient using Kronecker [Kemper et al.]
    - MTBDD implementations on multi-processor machines [KPZM04], networked clusters [ZPK05], grid-based architectures
- Parallelisation of approximate probabilistic model checking
  - simulation-based computations much easier to distribute

Some 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

References

- **[BGC04]** C. Baier, M. Grosser, and F. Ciesinski. Partial order reduction for probabilistic systems. In *Proc. QEST'04*, pages 230–239. IEEE Computer Society Press, 2004.
- **[Che06]** L. Cheung. Reconciling Nondeterministic and Probabilistic Choices. Ph.D. thesis, Radboud University of Nijmegen. 2006.
- **[DJJL01]** P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proc. PAPM/PROBMIV'01*, volume 2165 of *LNCS*, pages 39–56, Springer, 2001.
- **[DN04]** P. D'Argenio and P. Niebert. Partial order reduction on concurrent probabilistic programs. In *Proc. QEST'04*, pages 240–249. IEEE Computer Society Press, 2004.

References

- **[dAHJ01]** L. de Alfaro, T. Henzinger and R. Jhala. Compositional Methods for Probabilistic Systems. In *CONCUR 01: Concurrency Theory, 12th International Conference,* LNCS, Springer-Verlag, 2001.
- **[DM06]** A. Donaldson and A. Miller. Symmetry Reduction for Probabilistic Model Checking using Generic Representatives. In *Proc. 4th International Symposium on Automated Technology for Verification and Analysis (ATVA'06)*, Springer. October 2006.
- **[KNP06a]** M. Kwiatkowska, G. Norman and D. Parker. Game-based Abstraction for Markov Decision Processes. In *Proc. 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pages 157-166, IEEE CS Press. 2006.
- **[KNP06b]** M. Kwiatkowska, G. Norman and D. Parker. Symmetry Reduction for Probabilistic Model Checking. In *Proc. 18th International Conference on Computer Aided Verification (CAV'06)*, volume 4144 of LNCS, pages 234-248, Springer, 2006.

References

- **[KPZM04]** M. Kwiatkowska, D. Parker, Y. Zhang and R. Mehmood. Dual-Processor Parallelisation of Symbolic Probabilistic Model Checking. In Proc. MASCOTS'04, pages 123-130, IEEE CS Press. 2004.
- [**LHP06]** R. Lassaigne, T. Hérault and S. Peyronnet. APMC 3.0: Approximate verification of Discrete and Continuous Time Markov Chains. In *Proc. 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, 2006.
- **[YS02]** H. Younes and R. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *Proceedings of the 14th International Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 223-235, Copenhagen, Denmark, July 2002.
- **[ZPK05]** Y. Zhang, D. Parker and M. Kwiatkowska. A Wavefront Parallelisation of CTMC Solution using MTBDDs. In *Proc. International Conference on Dependable Systems and Networks (DSN'05)*, pages 732-742, IEEE Computer Society Press. 2005.

Part III

PRISM Case Studies

PRISM case studies

- Communication and multimedia protocols
  - Bluetooth device discovery [DKNP06]
  - IEEE 1394 FireWire root contention [KNS03]
  - IPv4 Zeroconf protocol [KNPS06]
  - IEEE 802.3 CSMA/CD protocol [DFH+04]
  - IEEE 802.11 WiFi wireless LANs [KNS02]
  - Zigbee (IEEE 802.15.4) protocol [Fru06]

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

PRISM case studies

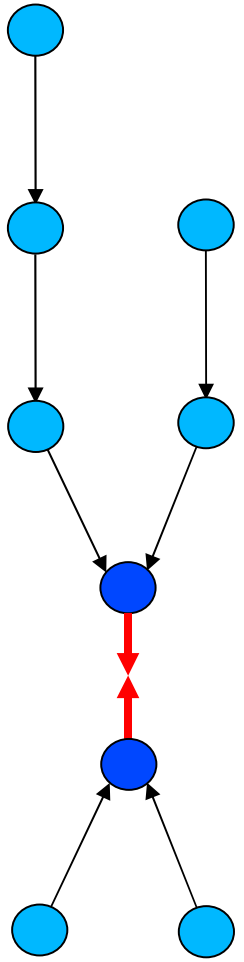
- Security systems/protocols
  - Probabilistic Contract Signing [NS06]
  - Crowds Protocol (anonymity) [Shm04]
  - Probabilistic Fair Exchange [NS06]
  - PIN Cracking Schemes [Ste06]
  - Negotation frameworks [BFW06]
  - Quantum cryptography [NPBG05]

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

PRISM case studies

- Randomised distributed algorithms for:
  - Byzantine Agreement [KN02]
  - Consensus [KNS01]
  - Self-stabilisation
  - Leader election
  - Mutual exclusion
  - Two Process Wait-Free Test-and-Set

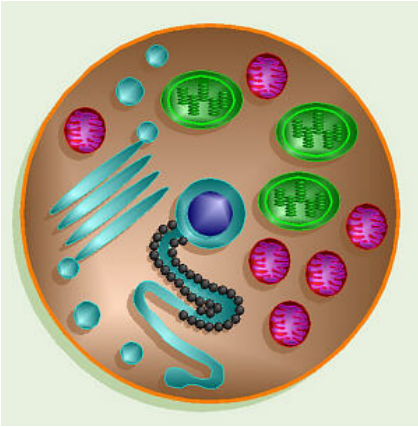
www.cs.bham.ac.uk/~dxp/prism/casestudies



PRISM case studies

- Analysis of behaviour/performance/reliability of:
  - Biological processes – signalling/cell cycle pathways [HKN+06]
  - Dynamic power management systems [NPK+05]
  - Dynamic voltage scaling algorithms [KNP05]
  - Manufacturing/control systems [KNP06,GF06]
  - Nanotechnology - NAND multiplexing [NPKS05]
  - Groupware protocols ("thinkteam") [BML05]

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



Bluetooth device discovery

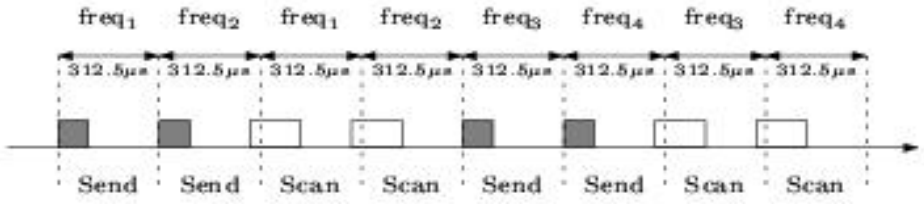
- Bluetooth: short-range low-power wireless protocol
  - widely available in phones, PDAs, laptops, ...
  - personal area networks (PANs)
  - open standard, specification freely available
- Uses frequency hopping scheme
  - to avoid interference (uses unregulated 2.4GHz band)
  - pseudo-random selection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

Bluetooth device discovery

- States of a Bluetooth device:
  - Standby: default operational state
  - Inquiry: device discovery
    - master looks for devices, slaves listens for master
  - Page: establish connection - synchronise clocks, etc.
  - Connected: device ready to communicate in a piconet
- Device discovery
  - manadatory first step before any communication possible
  - "page" reuses information from "inquiry" so is much faster
  - power consumption much higher for "page"
  - performance crucial

Master (sender) behaviour

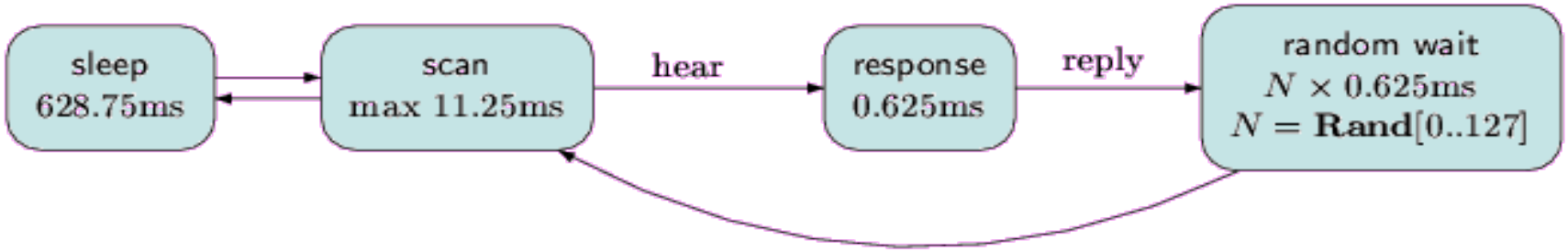
- 28 bit free-running clock CLK, ticks every 312.5µs
- Frequency hopping sequence determined by clock:
  - freq = [CLK_{16-12}+k+ ($CLK_{4-2,0}$- CLK_{16-12}) mod 16] mod 32
  - 2 trains of 16 frequencies (determined by offset k), 128 times each, swap between every 2.56s
- Broadcasts inquiry packets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

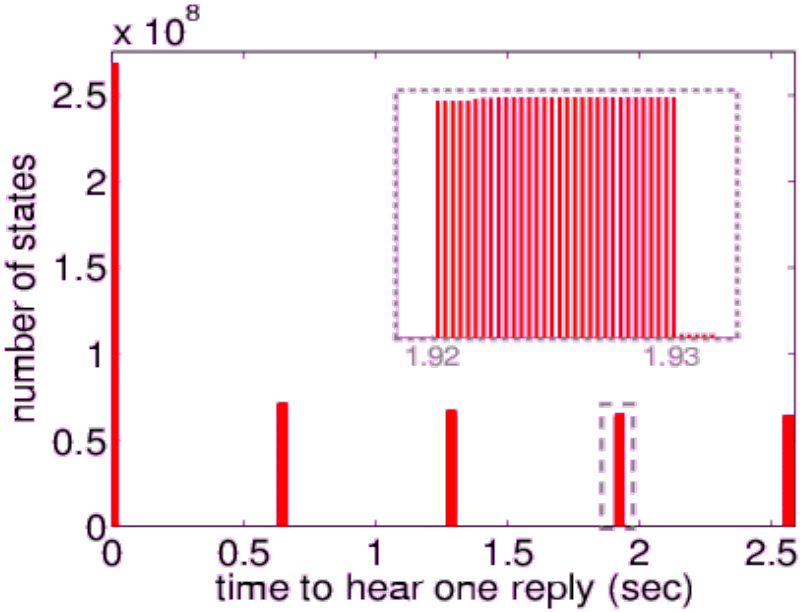
Bluetooth device discovery

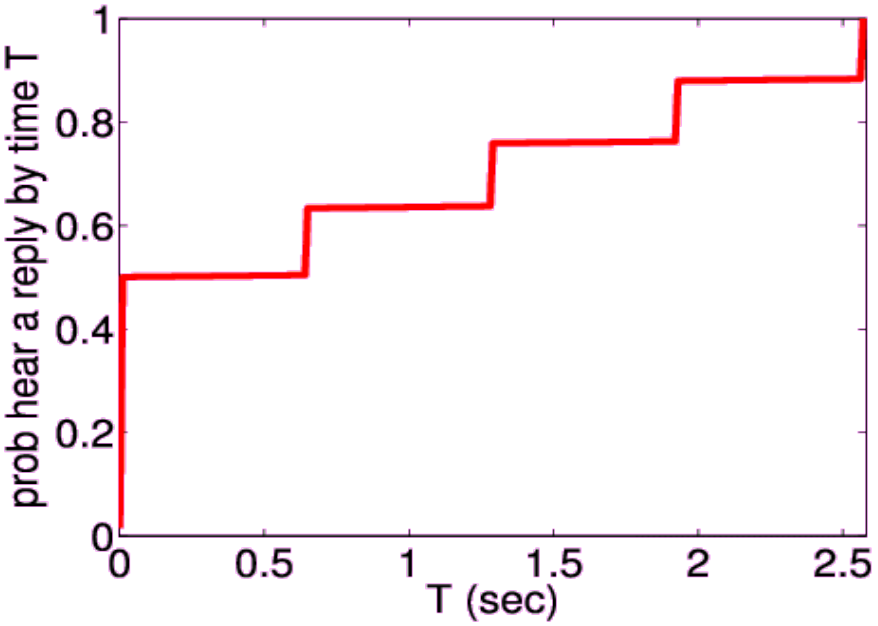
- Modelling in PRISM
  - model one sender and one receiver
  - synchronous (clock speed defined by Bluetooth spec)
  - randomised behaviour – use DTMC
  - model at lowest-level (one clock-tick = one transition)
  - use real values for delays, etc. from Bluetooth spec
- Modelling challenges
  - complex interaction between sender/receiver
  - combination of short/long time-scales – cannot scale down
  - sender/receiver not initially synchronised, state determined by 28 bit clock – 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:
  - 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

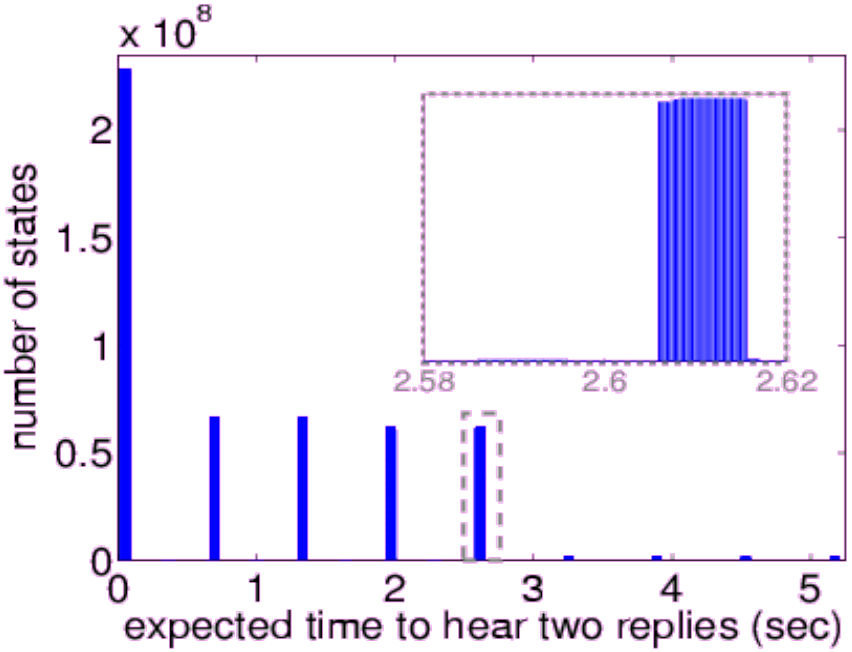
Bluetooth: Time to hear 1 reply

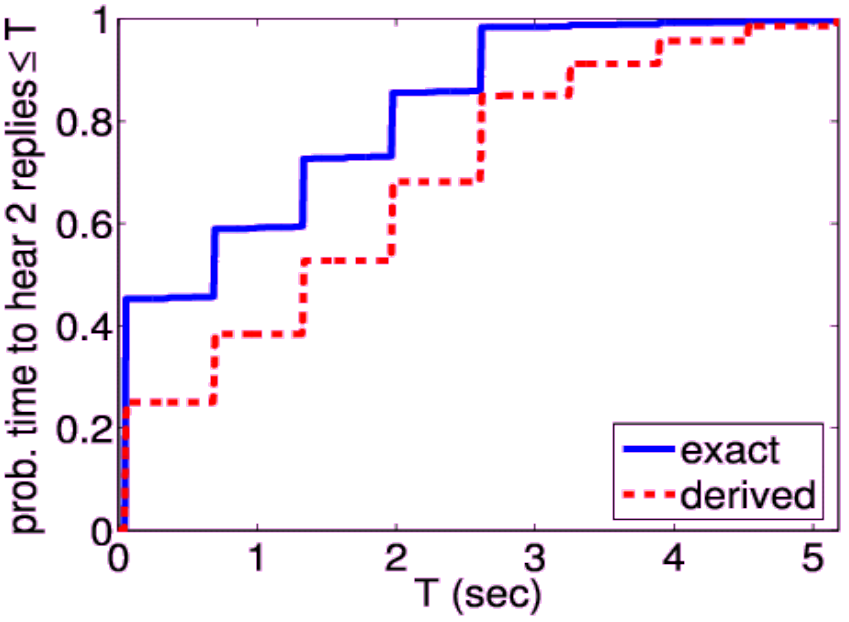




- worst-case expected time = 2.5716 sec
- in 921,600 possible initial states
- best-case = 635 μs

Bluetooth: Time to hear 2 replies





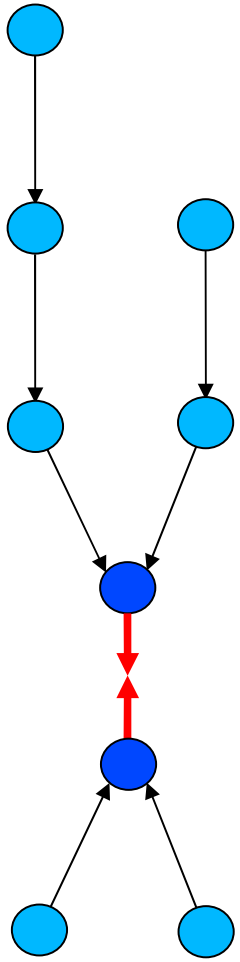
- worst-case expected time = 5.177 sec
- in 444 possible initial states
- compare actual CDF with derived version which assumes times to reply to first/second messages are independent

Bluetooth: Results

- Other results:
  - compare versions 1.2 and 1.1 of Bluetooth, confirm 1.1 slower
  - power consumption analysis
- Conclusions:
  - successful analysis of complex real-life model, actual parameters
  - exhaustive analysis: best-/worst-case values
    - can pinpoint scenarios which give rise to them
    - not possible with simulation approaches
  - model still realtively simple
    - consider multiple receivers?
    - combine with simulation?

IEEE 1394 (FireWire) root contention

- Serial bus for networking multimedia devices
  - "hot-pluggable" - add/remove devices (nodes) at any time
- Root contention protocol
  - leader election algorithm, when nodes join/leave
  - nodes send messages: "be my parent"
  - root contention: when nodes contend leadership
  - random choice: "fast"/"slow" delay before retry
- Properties of interest
  - time taken for leader election
  - effect of using biased coin - conjecture [Stoelinga]



FireWire - PRISM model

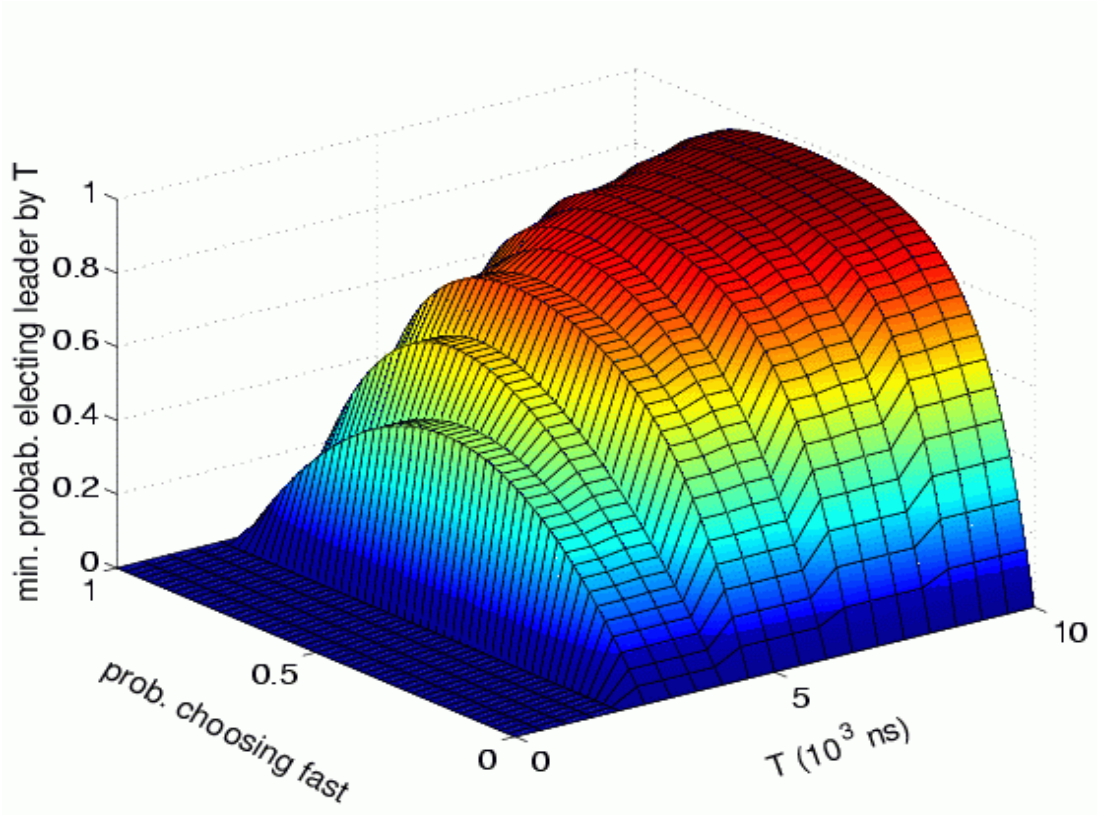
- Based on probabilistic timed automata (PTA) model
  - by Stoelinga et al. [SV99], [SS01]
  - infinite state (real-time)
  - digital clocks approach [KNS03] reduces to…

- PRISM model: finite-state MDP
  - concurrency: messages between nodes and wires
  - underspecification of delays (upper/lower bounds)
  - probability: coin toss
  - max. model size: 170 million states
  - analysed using PRISM's MTBDD engine

FireWire - Properties

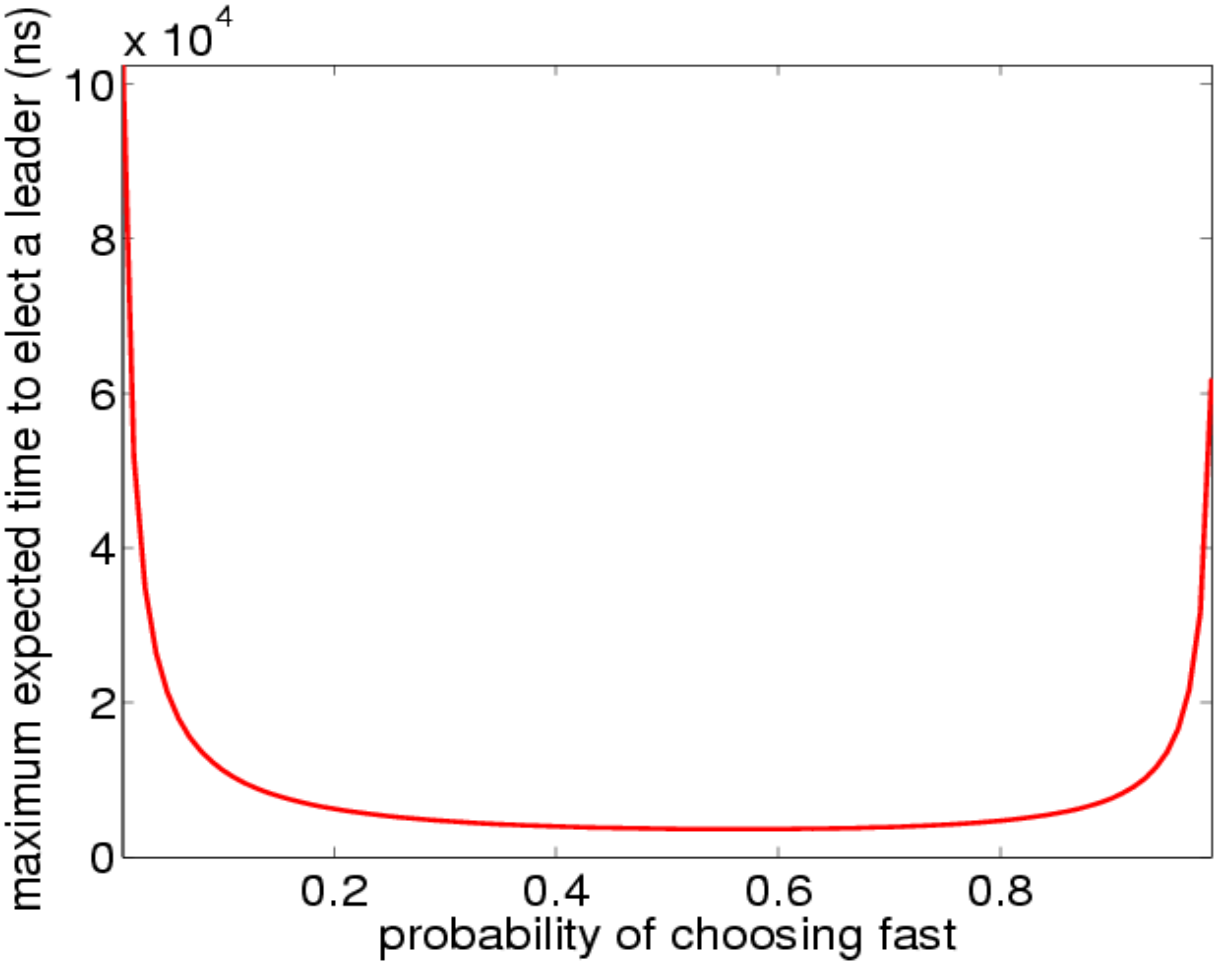
- "minimum probability that a leader is elected by time T"
  - add variable t to count elapsed time
  - Pmin=? [t≤T U “elected”]
  - vary: T, coin bias: probability of choosing "fast"
- "maximum expected time to elect a leader"
  - add timing costs
  - Rmax=? [F “elected”]
  - vary: coin bias

FireWire - Results



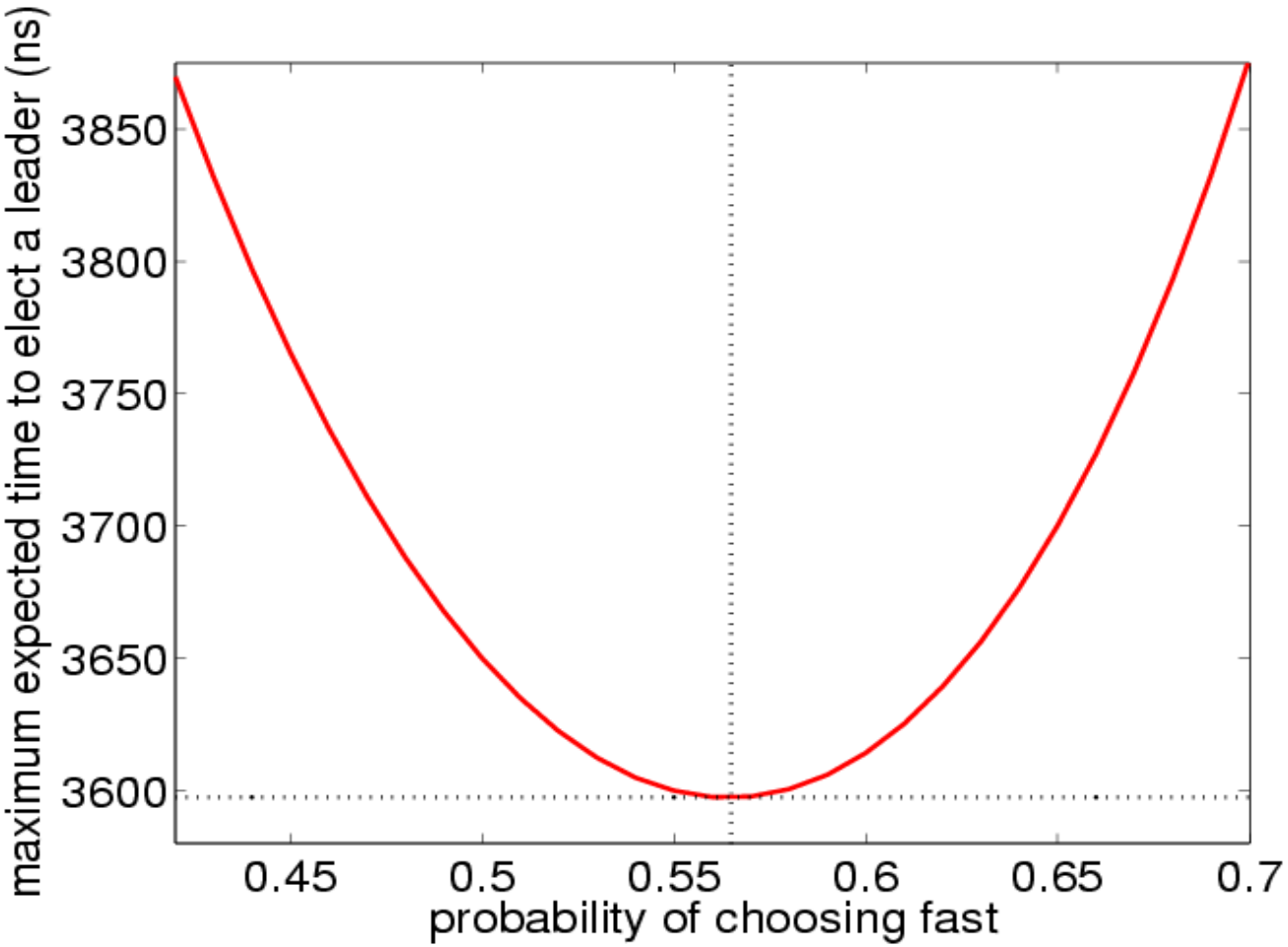
"minimum probability of electing leader by time T"

FireWire - Results



"maximum expected time to elect a leader"

FireWire - Results



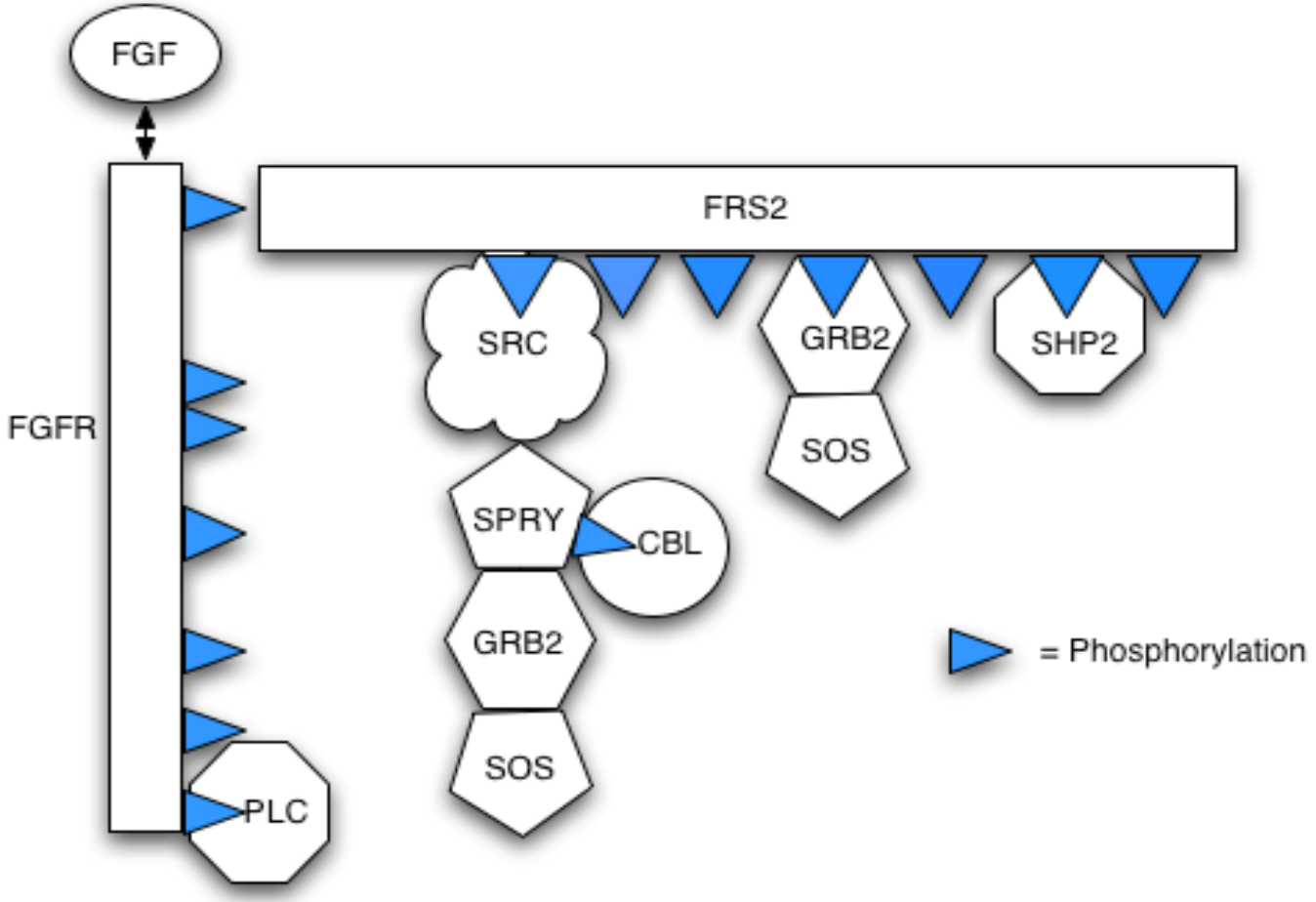
"maximum expected time to elect a leader"

Biased coin is beneficial

Case study: FGF pathway

- Fibroblast Growth Factors (FGF)
  - biological cell signalling pathway
  - key role in several contexts, e.g. wound healing, regulation of skeletal development (e.g. number of digits)
  - complex, not fully understood
- Removal experiments ("in silico genetics")
  - remove key model components, obtain testable predictions
  - compare with with real experimental data
  - use to prioritise (expensive) real experiments

The mechanism



Components of the model

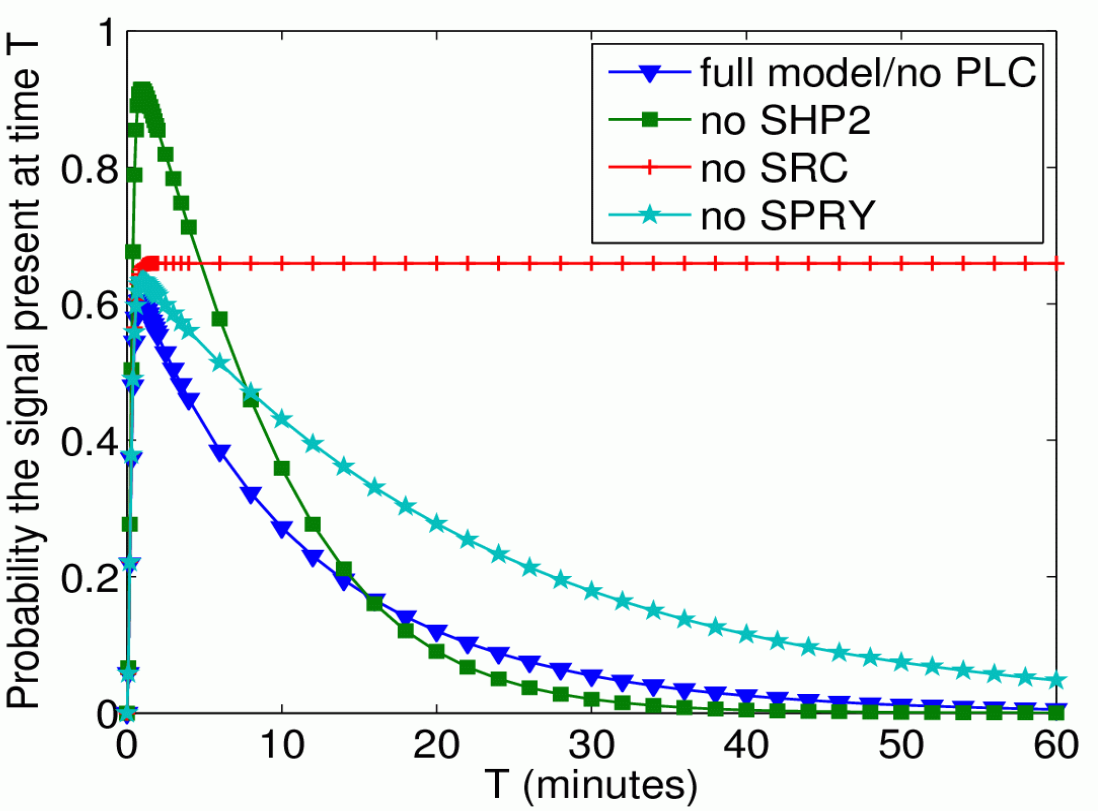
- FGF: the ligand
- FGFR: the receptor kinase, is activated by binding ligand and phosphorylates itself and FRS2
- FRS2: an adaptor, (phosphorylation allows effectors to bind)
- Plc: an enzyme, which induces proteolytic degradation of FGFR receptor (binds to phospho-FGFR)
- Shp2: a phosphatase, removes phosphate groups added by FGR (binds to phospho-FRS2)
- Grb2: connection to MAPK pathway (bind to phospho-FGFR)
- Src: a kinase, directs removal of FRS2 to another compartment. Binds to phospho-FRS2
- Spry: an attenuator, binds to and is phosphorylated by Src phospho-Spry binds GRB2 (competition with FRS2) and induces proteolytic degradation of FRS2

PRISM model of FGF pathway

- Biological Model
  - 12 elements
  - 14 phosphorylation sites
  - 14 sets of reaction rules (38 rules)
- PRISM model
  - continuous-time Markov chain (CTMC)
  - Suppose one element of each type
  - 10 modules and 26 variables
  - 80,616 states and 560,520 transition
  - relatively small state space
  - but highly complex: large number of interactions

Model checking results

Probability signal present at time T
P=? [true U[T,T] a_grb2]



No SRC: no relocation of FRS2, and hence the signal can remain active

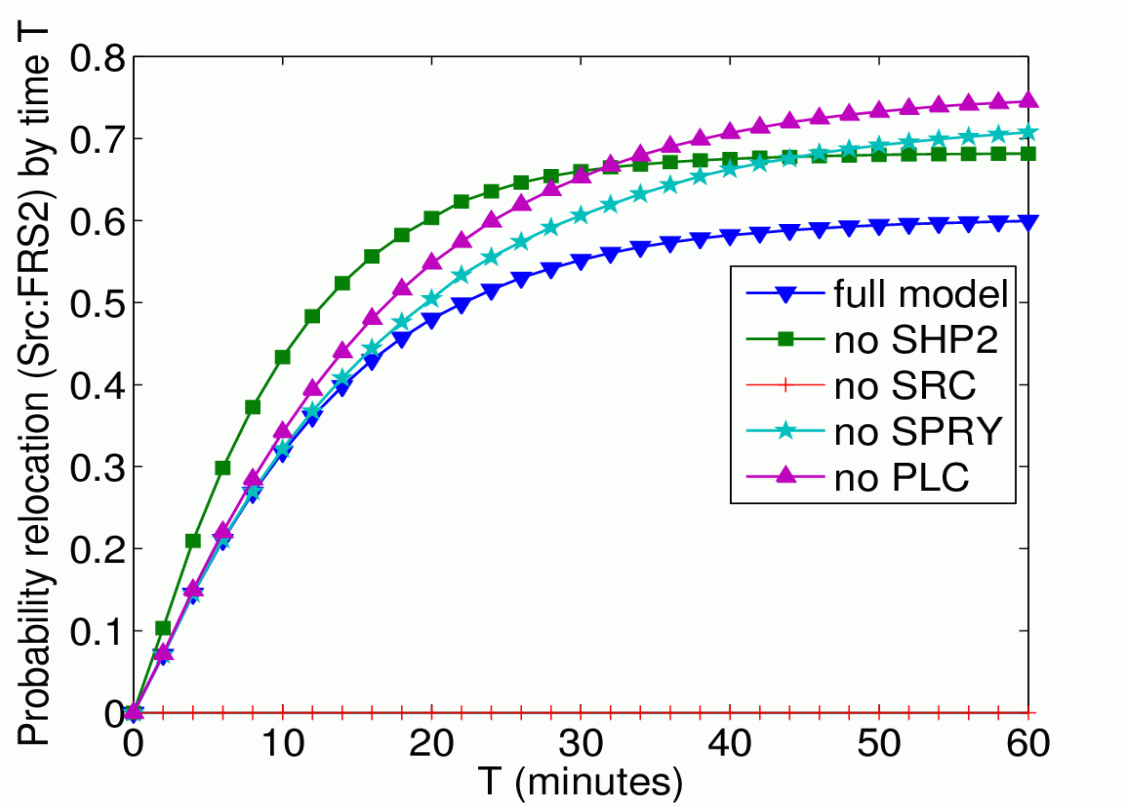
No SHP2: main cause of FRS2 dephosphorylation lost increasing the chance that:

- Grb2 bound to FRS
  faster increase in signal
- SRC bound to FRS2
  faster degradation in signal

Model checking results

Probability SRC causes degradation/relocation by time T

P=?[!(a_src | a_spry | a_plc) U≤T a_src]



SRC removed - SRC cannot cause degradation

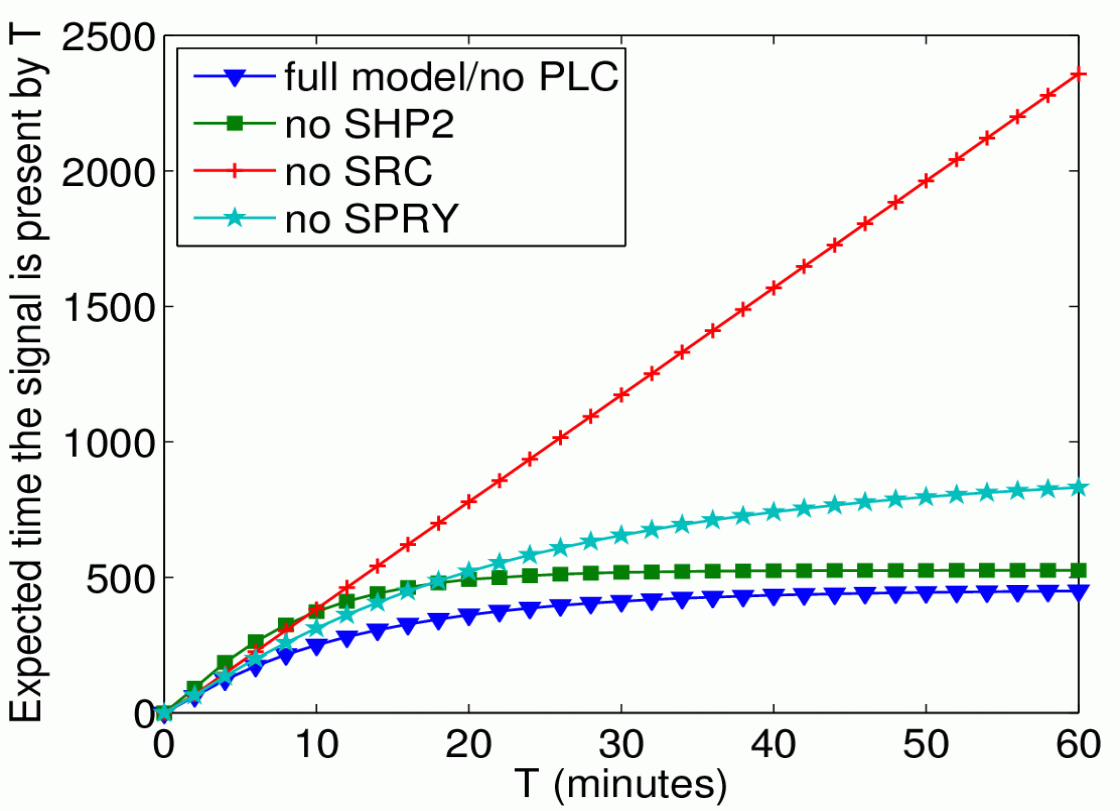
Removal of SPRY/PLC remove alternative cause of degradation/relocation, hence SRC has greater influence

Removal SHP2 increases chance that SRC is bound: SRC has greater influence

Model checking results

Expected time GRB2 bound to FRS2 within time T

R=? [C≤T] (assign reward 1 to states where Grb2:FRS2)



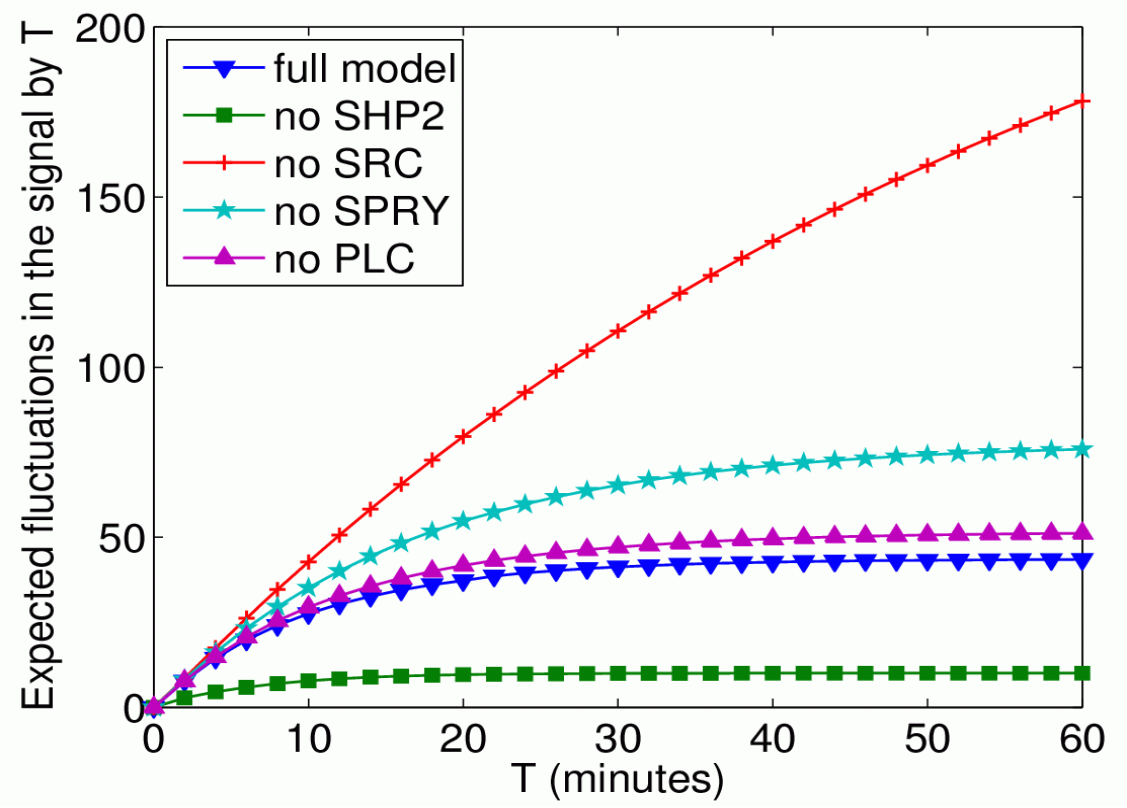
No SRC: no relocation of FRS2 and greater chance FRS2 remains active for longer, hence GRB2 and FRS2 spend more time bound

SPRY: no degradation of FRS2, again GRB2 and FRS2 spend more time bound (but SPRY has smaller influence than SRC)

Model checking results

Expected number of times GRB2 & FRS2 bind by T

R=? [C≤T] (assign reward 1 to transitions binding Grb2 & FRS2)



Cases when SRC and SPRY removed: increased chance that FRS2 remains active, and hence GRB2 and FRS2 can bind more often

No SHP2: decrease in the chance that GRB2:FRS2 unbind, therefore the chance that GRB2 and FRS2 are in a position to (re)bind decreases

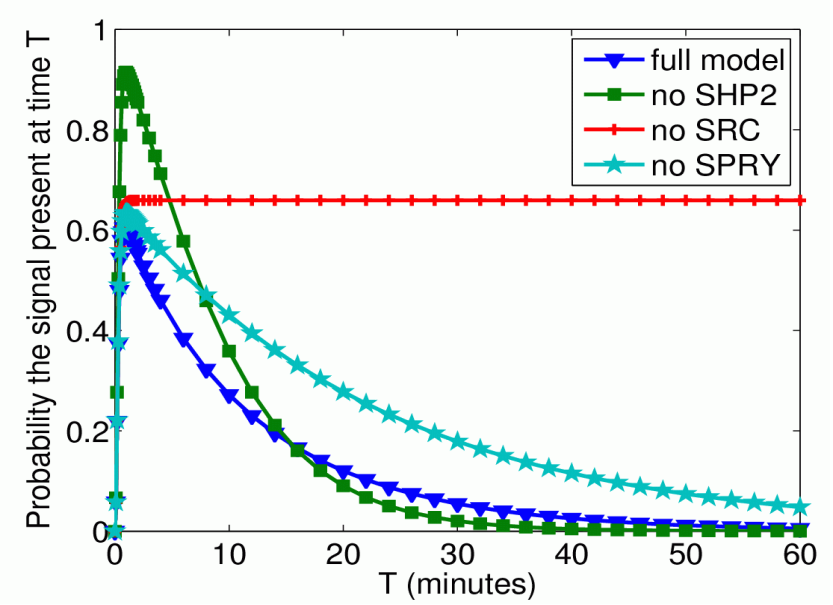
Model checking results

- Long run probability GRB2 is bound to FRS2: S=? [a_grb2]
- Expected bindings of GRB2 & FRS2 before degradation:
  - R=? [F (a_src | a_spry | a_plc)] (reward 1 for binding transitions)
- Expected time GRB2 & FRS2 spend bound before degradation:
  - R=? [F (a_src | a_spry | a_plc)] (reward 1 for bound states)

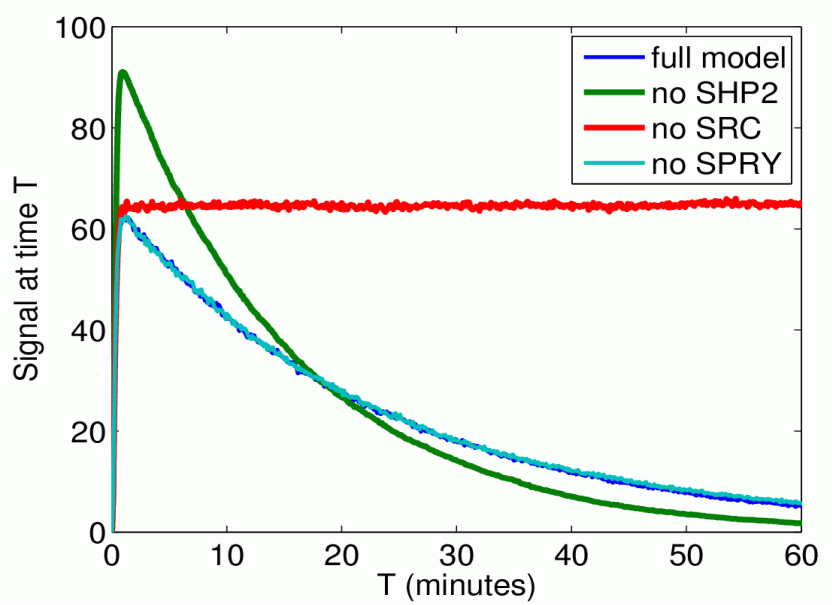
	probability bound	expected Bindings	expected time bound
full model	7.54e-7	43.10	6.270
no **SHP2**	3.29e-9	10.05	7.789
no **SRC**	0.65946	283.2	39.61
no **SPRY**	4.46e-6	78.33	10.88
no **PLC**	0.0	51.54	7.562

Comparison with simulation model

Probabilistic model checking
(1 element of each type)



Simulation
average of 100 runs
(100 elements of each type)



References

- **[BFW06]** P. Ballarini, M. Fisher and M. Wooldridge. Automated Game Analysis via Probabilistic Model Checking: A Case Study. In Proc. 3rd Workshop on Model Checking and Artificial Intelligence (MoChArt'05), volume 149 of ENTCS, pages 125-137, Elsevier. 2006.
- **[BML05]** M. ter Beek, M. Massink and D. Latella. Towards Model Checking Stochastic Aspects of the thinkteam User Interface. In M. Harrison (editor) *Proc. 12th International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'05)*, volume 3941 of Lecture Notes in Computer Science, pages 39-50, Springer. 2005.
- **[DFH+04]** M. Duflot, L. Fribourg, T. Hérault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet and C. Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. In Proc. 4th Workshop on Automated Verification of Critical Systems (AVoCS'04), volume 128(6) of ENTCS, pages 195-214, Elsevier Science. 2004.
- **[DKNP06]** M. Duflot, M. Kwiatkowska, G. Norman and D. Parker. A Formal Analysis of Bluetooth Device Discovery. *International Journal on Software Tools for Technology Transfer (STTT)*. To appear. 2006.

References

- **[EGL85]** S. Even, O. Goldreich and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637-647, 1985
- **[Fru06]** M. Fruth. Probabilistic Model Checking of Contention Resolution in the IEEE 802.15.4 Low-Rate Wireless Personal Area Network Protocol. In Proc. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'06). To appear. 2006.
- **[GF06]** J. Greifeneder and J. Frey. Dependability analysis of networked automation systems by probabilistic delay time analysis. In *Proc. 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'06),* pages 269-274. May 2006.
- **[HKN+06]** J. Heath, M. Kwiatkowska, G. Norman, D. Parker and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In C. Priami (editor) *Proc. Computational Methods in Systems Biology (CMSB'06)*, volume 4210 of Lecture Notes in Lecture Notes in Bioinformatics, pages 32-47, Springer Verlag. 2006.

References

- **[KN02]** M. Kwiatkowska and G. Norman. Verifying Randomized Byzantine Agreement. In *Proc. Formal Techniques for Networked and Distributed Systems (FORTE'02)*, volume 2529 of LNCS, pages 194-209, Springer-Verlag. November 2002.
- **[KNP05]** M. Kwiatkowska, G. Norman and D. Parker. Probabilistic Model Checking and Power-Aware Computing. In *In Proc. 7th International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS)*, pages 6-9. September 2005.
- **[KNP06]** M. Kwiatkowska, G. Norman and D. Parker. Controller Dependability Analysis By Probabilistic Model Checking. *Control Engineering Practice*, Elsevier. To appear. 2006.
- **[KNPS06]** M. Kwiatkowska, G. Norman, D. Parker and J. Sproston. Performance Analysis of Probabilistic Timed Automata using Digital Clocks. *Formal Methods in System Design*, 29, pages 33-78, Springer. August 2006.

References

- **[KNS01]** M. Kwiatkowska, G. Norman and R. Segala. Automated Verification of a Randomised Distributed Consensus Protocol Using Cadence SMV and PRISM. In *Proc. CAV'01*, volume 2102 of LNCS, pages 194-206, Springer-Verlag. 2001.
- **[KNS02]** M. Kwiatkowska, G. Norman and J. Sproston. Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol. In Proc. PAPM/PROBMIV'02, volume 2399 of LNCS, pages 169-187. 2002.
- **[KNS03]** M. Kwiatkowska, G. Norman and J. Sproston. Probabilistic Model Checking of Deadline Properties in the IEEE1394 FireWire Root Contention Protocol. *Formal Aspects of Computing*, 14(3), pages 295-318. April 2003.

References

- **[NPBG05]** R. Nagarajan, N. Papanikolaou, G. Bowen and S. Gay. An Automated Analysis of the Security of Quantum Key Distribution. In *Proc. 3rd International Workshop on Security Issues in Concurrency (SecCo'05)*. 2005.
- **[NPKS05]** G. Norman, D. Parker, M. Kwiatkowska and S. Shukla. Evaluating the Reliability of NAND Multiplexing with PRISM. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(10), pages 1629-1637. October 2005.
- **[NPK+05]** G. Norman, D. Parker, M. Kwiatkowska, S. Shukla and R. Gupta. Using Probabilistic Model Checking for Dynamic Power Management. *Formal Aspects of Computing*, 17(2), pages 160-176, Springer-Verlag. August 2005.
- **[NS06]** G. Norman and V. Shmatikov. Analysis of Probabilistic Contract Signing. *Journal of Computer Security*. To appear. 2006.

References

- **[Shm04]** V. Shmatikov. Probabilistic Model Checking of an Anonymity System. *Journal of Computer Security*, 12(3/4), pages 355-377. 2004.
- **[SS01]** D. Simons. and M. Stoelinga. Mechanical verification of the IEEE1394a root contention protocol using Uppaal2k. *International Journal on Software Tools for Technology Transfer* 3(4):469-485, 2001.
- **[Ste06]** G. Steel. Formal Analysis of PIN Block Attacks. *Theoretical Computer Science*. To appear. 2006.
- **[SV99]** M. Stoelinga and F. Vaandrager. Root contention in IEEE 1394. In *Proc. 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)*, pp. 53-74, 1999.

Further Reading

- **[DKN+06]** M. Duflot, M. Kwiatkowska, G. Norman, D. Parker, S. Peyronnet, C. Picaronny and J. Sproston. Practical Applications of Probabilistic Model Checking to Communication Protocols. In *Handbook of Formal Methods in Industrial Critical Systems (FMICS)*. To appear. 2006.
- **[Nor04]** G. Norman. Analysing Randomized Distributed Algorithms. In *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925 of Lecture Notes in Computer Science. August 2004.
- **[KNP05d]** M. Kwiatkowska, G. Norman and D. Parker. Quantitative analysis with the probabilistic model checker PRISM. *Electronic Notes in Theoretical Computer Science*, 153(2), pages 5-31, Elsevier. May 2005.
- **[KNP05b]** M. Kwiatkowska, G. Norman and D. Parker. Probabilistic model checking in practice: Case studies with PRISM. *ACM Performance Evaluation Review*, 32(4), pages 16-21. March 2005.