Protocol analysis via probabilistic model checking

Marta Kwiatkowska
School of Computer Science
THE UNIVERSITY OF BIRMINGHAM

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

CMU, 12th Nov 2004

Overview

- Network protocols
  - Probability - why needed, challenges
- Probabilistic model checking
  - Some models and logics
  - What does it involve?
  - The PRISM model checker
- Case studies
  - Self-stabilising algorithms
  - IPv4 Zeroconf link-local addressing
  - Bluetooth device discovery
  - Contract signing
- Challenges for future

Ubiquitous computing: the trends…




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

Ubiquitous computing: users expect…



- …assurance of
  - safety
  - correctness
  - performance
  - reliability

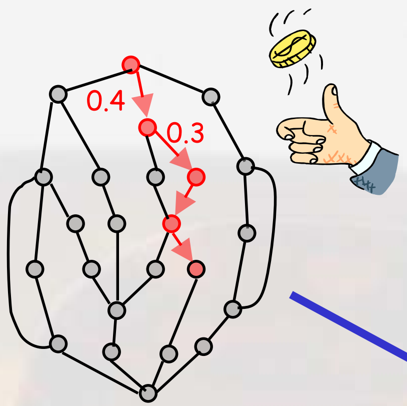
- For example:
  - Is my e-savings account secure?
  - Can someone bluesnarf from my phone?
  - How fast is the communication from my PDA to printer?
  - Is my mobile phone energy efficient?
  - Is the operating system reliable?
  - Can the laptop recover from faults with no effort on my part?

Probability helps

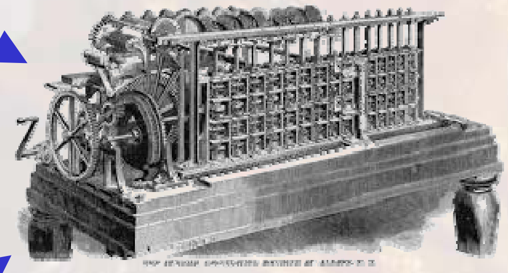
- In distributed co-ordination algorithms
  - As a symmetry breaker
    - "leader election is eventually resolved with probability 1"
  - In gossip-based routing and multicasting
    - "the message will be delivered to all nodes with high probability"
- When modelling uncertainty in the environment
  - To quantify failures, express soft deadlines, QoS
    - "the chance of shutdown is at most 0.1%"
    - "the probability of a frame delivered within 5ms is at least 0.91"
  - To quantify environmental factors in decision support
    - "the expected cost of reaching the goal is 100"
- When analysing system performance
  - To quantify arrivals, service, etc, characteristics
    - "in the long run, mean waiting time in a lift queue is 30 sec"

Probabilistic model checking...

in a nutshell



Probabilistic model



Probabilistic Model Checker

or

or

The probability

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
...
State 12: 0
State 13: 0.1245

send $\rightarrow$ $\mathbf{P}_{\geq 0.9}(\Diamond \text{deliver})$

Probabilistic temporal logic specification

Probability elsewhere

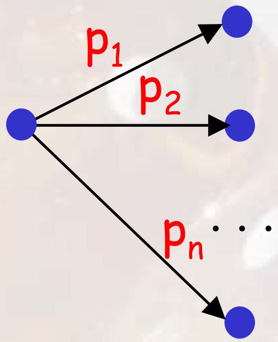
- In performance modelling
  - Pioneered by Erlang, in telecommunications, ca 1910
  - Models: typically continuous time Markov chains
  - Emphasis on steady-state and transient probabilities

- In stochastic planning
  - Cf Bellman equations, ca 1950s
  - Models: Markov decision processes
  - Emphasis on finding optimum policies

- Our focus, probabilistic model checking
  - Distinctive, on automated verification for probabilistic systems
  - Temporal logic specifications, automata-theoretic techniques
  - Shared models
  - Exchanging techniques with the other two areas

Probabilistic models: discrete time

- Labelled transition systems
  - Discrete time steps
  - Labelling with atomic propositions



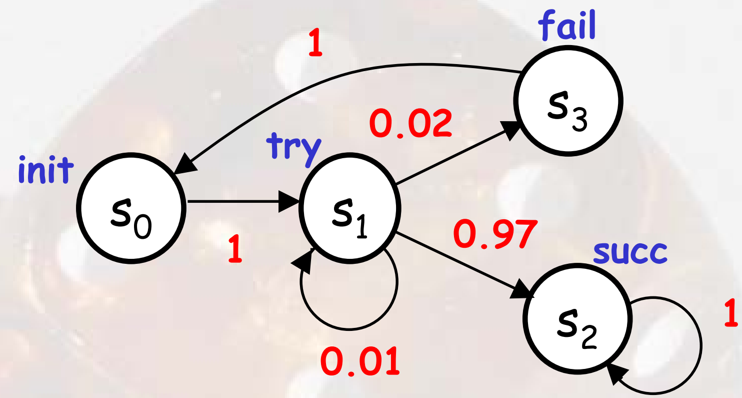
$\sum_i p_i = 1$

- Probabilistic transitions
  - Move to state with given probability
  - Represented as discrete probability distribution

- Model types
  - Discrete time Markov chains (DTMCs): probabilistic choice only
  - Markov decision processes (MDPs): probabilistic choice and nondeterminism

Discrete-Time Markov Chains (DTMCs)

- Features:
  - Only probabilistic choice in each state



- Formally, (S,s_0,P,L):
  - S finite set of states
  - s_0 initial state
  - P: S £ S ! [0,1] probability matrix, s.t. $\sum_{s'} P(s,s') = 1$, all s
  - L: S ! 2^{AP} atomic propositions

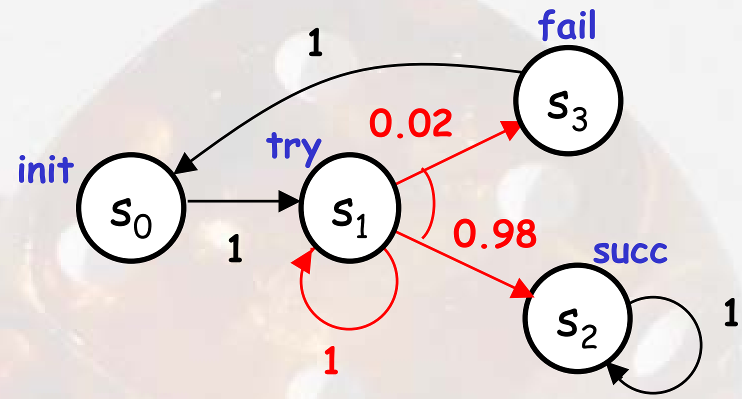
- Unfold into infinite paths $s_0 s_1 s_2 s_3 s_4 \ldots$ s.t. $P(s_i,s_{i+1}) > 0$, all i

- Probability for finite paths, multiply along path

  e.g. $s_0\, s_1\, s_1\, s_2$ is 1 ¢ 0.01 ¢ 0.97 = 0.0097

Markov Decision Processes (MDPs)

- Features:
  - Nondeterministic choice
  - Parallel composition of DTMCs



- Formally, $(S, s_0, Steps, L)$:
  - S finite set of states
  - s_0 initial state
  - Steps maps states s to sets of probability distributions μ over S
  - L: S ! 2^{AP} atomic propositions

- Unfold into infinite paths $s_0 \mu_0 s_1 \mu_1 s_2 \mu_2 s_3 \ldots$ 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 $s_0 \mu_0 s_1 \mu_1 \ldots s_n$ to a distribution from state s_n

The logic PCTL: syntax

- Probabilistic Computation Tree Logic [HJ94,BdA95,BK98]
  - For DTMCs/MDPs, based on CTL
  - New probabilistic operator, e.g. send $\rightarrow$ $\mathbf{P}_{\geq 0.9}(\Diamond \text{deliver})$

    "whenever a message is sent, the probability that it is eventually delivered is at least 0.9"

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

$$\phi ::= \text{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathbf{P}_{\bowtie p}(\alpha)$$

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

where $p \in [0,1]$ is a probability bound and $\bowtie\ \in \{ <, >, \ldots \}$

- Subsumes the qualitative variants [Var85,CY95] $\mathbf{P}_{=1}(\alpha)$, $\mathbf{P}_{>0}(\alpha)$

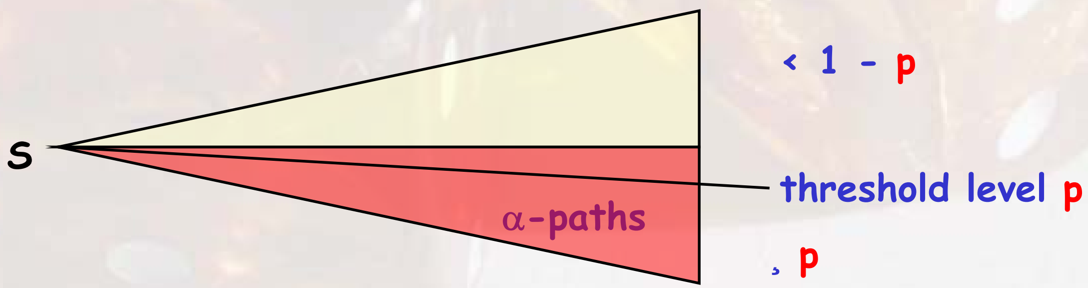
- Extension with cost/rewards and expectation operator $\mathbf{E}_{\bowtie c}(\phi)$

The logic PCTL: semantics

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

- The probabilistic operator is a quantitative analogue of 8, 9

$s \; {}^{2}{}_{Adv} \; \mathbf{P}_{» p}(\alpha)$, $Pr^{A} \{ \pi \; 2 \; Path^{A}_{s} \; j \; \pi \; {}^{2}{}_{Adv} \; \alpha \} \; » \; p$
for all A 2 Adv



The logic PCTL: model checking

- For the probabilistic operator and Until, solve
  - recursive linear equation for DTMCs
  - linear optimisation problem (form of Bellman equation) for MDPs
  - typically iterative solution methods

- Need to combine
  - conventional graph traversal
  - numerical linear algebra and linear optimisation (value iteration)

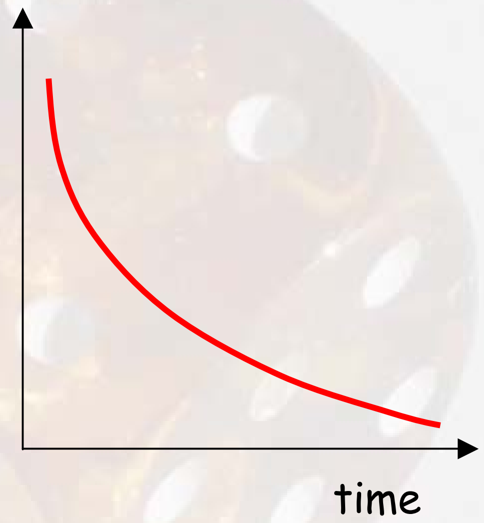
- Qualitative properties (probability 1, 0) proceed by graph traversal [Var85,dAKNP97], implemented in BDDs

- Expected min/max cost via stochastic shortest path problem, implement algorithms for MDPs from [dA97]

Probabilistic models: continuous

- Assumptions on time and probability
  - Continuous passage of time
  - Cont. randomly distributed delays
  - Continuous space

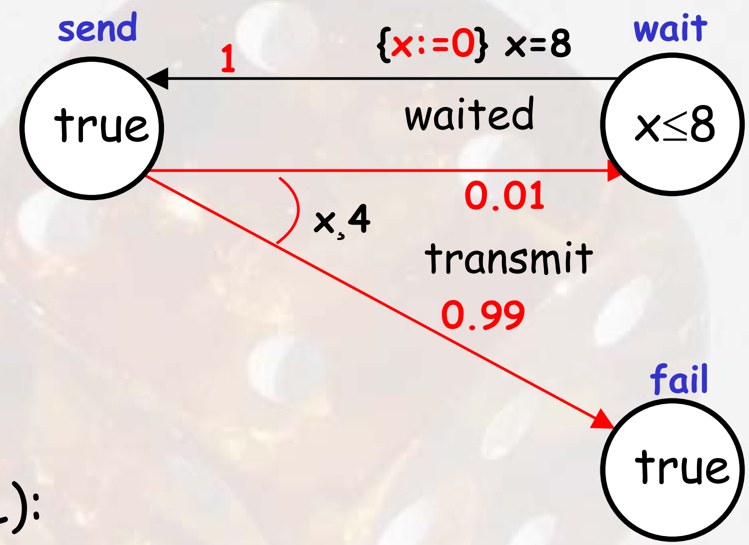
- Model types
  - (also supported) Continuous time Markov chains (CTMCs): exponentially distributed delays, discrete space, no nondeterminism
  - Probabilistic Timed Automata (PTAs): dense time, (usually) discrete probability, admit nondeterminism
  - (not considered) Labelled Markov Processes (LMPs): continuous space/time, no nondeterminism



$S_0^{+1} f(x)dx = 1$

Probabilistic Timed Automata: syntax

- Features:
  - Clocks, x, real-valued
  - Can be reset,
    e.g. {x:=0}
  - Invariants, e.g. x·8
  - Probabilistic transitions, guarded e.g. x¸4, x=8



- Formally, (Loc,s_0,Inv,prob,Act,L):
  - Loc finite set of locations
  - s_0 initial location
  - Inv maps locations s to invariant clock constraints
  - prob probabilistic edge relation that yields the probability of moving from s to s' if enabled at s, resetting specified clocks
  - Act action labelling of transitions μ (probability distribution)
  - L: S ! 2^{AP} atomic propositions

The logic PTCTL

- Probabilistic Timed CTL for PTAs
  - Based on TCTL [AD94]
  - Add probabilistic operator $\mathbf{P}_{» p}(¢)$ of PCTL

- Syntax

$$\phi ::= a \mid ζ \mid \phi \text{ Ç } \phi \mid :\phi \mid z.[\phi] \mid \mathbf{P}_{» p}(\phi \text{ U } \phi)$$

where z ranges over formula clocks, ζ are clock constraints over formula and system clocks

- Example: $z.[\ \mathbf{P}_{¸\ 0.98}\ (\}\ \text{delivered Æ } z < 5)]$

  "under any scheduling, with probability ¸ 0.85 the message is correctly delivered within 5 ms"

- Semantics derived from PCTL and TCTL

Model checking for PTAs

- Apply appropriate quotient, derive time-abstract MDP
  - Use standard TA constructs: regions, digitisation, zones
  - Model check the resulting MDP using standard methods

- This is possible since

$$\{s,E \text{ j } s,E \ {}^{2}\mathbf{P}_{» \, p}(\phi_1 \, U \, \phi_2)\} = \begin{cases} \{s,E \text{ j } s,E \ {}^{2} \ p^{max}(\phi_1 \, U \, \phi_2)\text{~}p\} \text{ if } \sim \ 2 \ \{\cdot, <\} \\ \{s,E \text{ j } s,E \ {}^{2} \ p^{min}(\phi_1 \, U \, \phi_2)\text{~}p\} \text{ if } \sim \ 2 \ \{¸, >\} \end{cases}$$

where for any PTCTL formula α, fixed s,E:

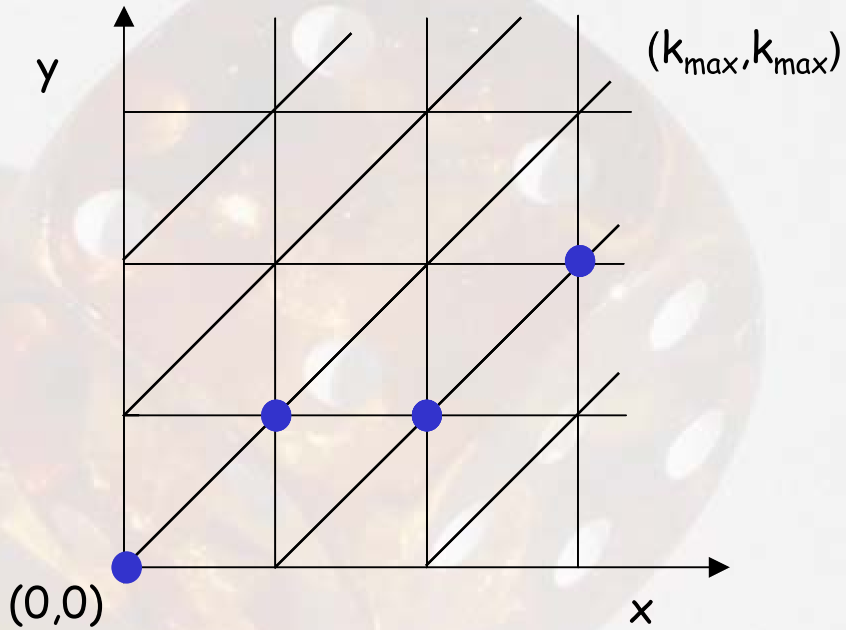
$$p^{max}(\alpha) = \sup_{A \ 2 \ Adv} Pr^{A} \{ \pi \ 2 \ Path^{A}_{s} \text{ j } \pi,E \ {}^{2} \ \alpha \}$$

$$p^{min}(\alpha) = \inf_{A \ 2 \ Adv} Pr^{A} \{ \pi \ 2 \ Path^{A}_{s} \text{ j } \pi,E \ {}^{2} \ \alpha \}$$

- Thus sufficient to compute maximum/minimum probability

Model checking for PTAs: digital clocks

- ε-digitisation [HMP92]
  - restrict to closed, diagonal-free Tas
  - Time domain N, with integer-valued clocks
  - Define time increment by min{v(x)+t, k_x +1)
  - Integer-valued time elapse



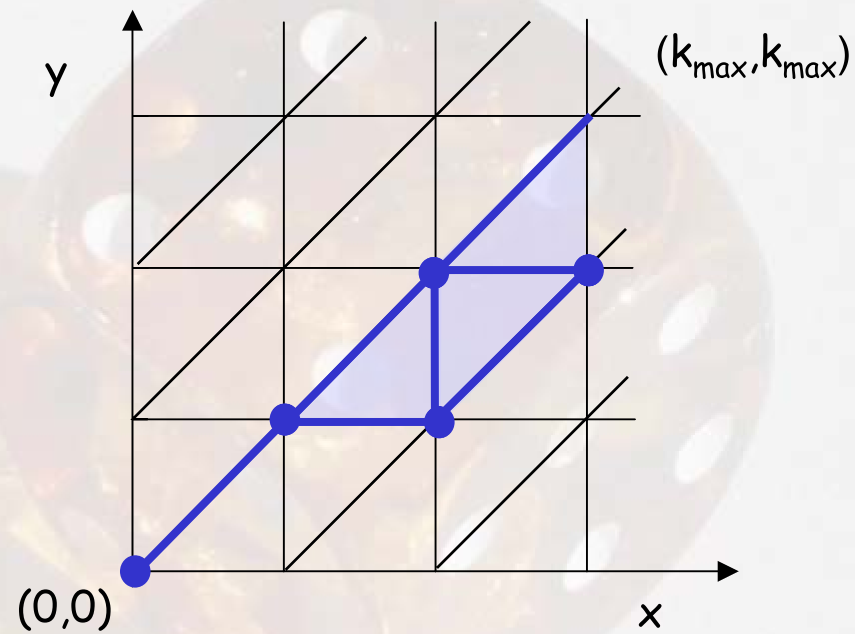
- Finiteness of state space immediate
- Preserves a subset of properties, cf reachability

Model checking: digital clocks

- Main result [KN'02,'03]: digitisation preserves
  - minimum/maximum reachability probability
  - minimum/maximum expected cost reachability
- Digitally-clocked PTAs (and variables representing cost) can be represented in the PRISM input language, and so can apply model checking directly on MDPs
- Restriction to closed, diagonal-free not important for many case studies
- Subset of PTCTL only, but expected costs possible
- Problem: inefficiency for some models, as large constants give rise to very large state spaces

Model checking for PTAs: symbolic

- Zones
  - usually convex conjunctions of atomic constraints,
  - e.g. $0<x<2$ Æ $0<y<1$
  - operations on zones: conjunctions, pre, post
- Can build time abstract zone graph - an MDP



- Methods (experimental implementation using DBMs)
  - Loss of on the fly (must first construct MDP over zones)
  - Forwards: max probabilities only, need not be preserved
  - Backwards: max/min probabilities, need non-convex zones, but full PTCTL

Probabilistic model checking in practice

- Model construction: probability matrices
  - Enumerative
    - Manipulation of individual states
    - Size of state space main limitation
  - Symbolic
    - Manipulation of sets of states
    - Compact representation possible in case of regularity

- Temporal logic model checking: currently limited to
  - discrete probability/space models
  - CTMCs
  - Simulation admits more general distributions

- Probabilistic Symbolic Model Checker PRISM

The PRISM tool: overview

- Functionality
  - Direct support for models: DTMCs, MDPs and CTMCs
  - Extension with costs/rewards, expectation operator
  - PTAs with digital clocks by manual translation
  - Connection from KRONOS to PRISM for PTAs
  - Experimental implementation using DBMs/DDDs for PTAs

- Input languages
  - System description
    - probabilistic extension of reactive modules [Alur and Henzinger]
  - Probabilistic temporal logics: PCTL and CSL

- Implementation
  - Symbolic model construction (MTBDDs), uses CUDD [Somenzi]
  - Three numerical computation engines
  - Written in Java and C++

The PRISM tool: implementation

- Numerical engines
  - Symbolic, MTBDD based
    - Fast construction, reachability analysis
    - Very large models if regularity
  - Enumerative, sparse-matrix based
    - Generally fast numerical computation
    - Model size up to millions
  - Hybrid
    - Speed comparable to sparse matrices for numerical calculations
    - Limited by size of vector
- Experimental results
  - Several large scale examples: 10^{10} - 10^{30} states
  - No engine wins overall
  - See www.cs.bham.ac.uk/~dxp/prism

PRISM real-world case studies

- MDPs/DTMCs
  - Bluetooth device discovery [ISOLA'04]
  - Crowds anonymity protocol (by Shmatikov) [JCS 2004]
  - Randomised consensus [CAV'01]
  - Randomised Byzantine Agreement [FORTE'02]
  - NAND multiplexing for nanotechnology (with Shukla) [VLSI'04]
  - Contract signing protocols [Shmatikov, Norman]

- CTMCs
  - Dynamic Power Management (with Shukla and Gupta) [HLDVT'02]
  - Dependability of embedded controller [INCOM'04]

- PTAs
  - IPv4 Zeroconf dynamic configuration [FORMATS'03]
  - Root contention in IEEE 1394 FireWire [FAC 2003, STTT 2004]
  - IEEE 802.11 (WiFi) Wireless LAN MAC protocol [PROBMIV'02]

Case Study: Self-Stabilization

- Self-stabilizing protocol for a network of processes
  - starts from possibly illegal start state
  - returns to a legal (stable) state
    - without any outside intervention
    - within some finite number of steps

- Network: synchronous or asynchronous ring of N processes
  - Illegal states: more than on process is privileged (has a token)
  - Stable states: exactly one process is privileged (has a token)
  - Properties
    - From any state, a stable state is reached with probability 1
    - Expected time to reach a stable state

Herman's self-stabilising protocol

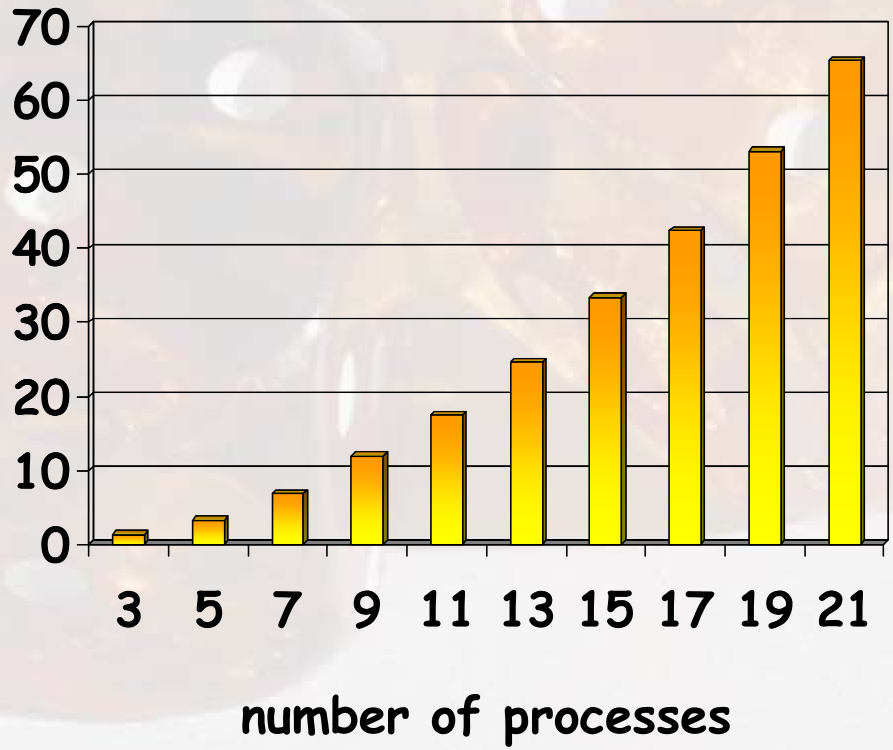
- Synchronous ring of N (N odd) processes (DTMC)
  - Each process has a local boolean variable x_i
  - Token in place i if $x_i=x_{i+1}$
  - Basic step of process i:
    - if $x_i = x_{i+1}$ make a uniform random choice as to the next value of x_i
    - otherwise set x_i to the current value of x_{i+1}
  - In the PRISM language:

```
module process1
    x1 : bool;
    [step]    x1=x2    -> 0.5 : x1'=0 + 0.5 : x1'=1;
    [step]  !(x1=x2)  -> x1'=x2;
endmodule

module process2  = process1 [x1=x2, x2=x3] endmodule
                ⋮                          ⋮
module processN = process1 [x1=xN, x2=x1] endmodule
```

Results: Herman's protocol

- P¸ 1 (◊stable): min probability of reaching a stable state is 1
- E. ? (stable): max expected time (number of steps) to reach a stable state, assuming the probability is 1, is:



Case study: IPv4 Zeroconf protocol

- IPv4 ZeroConf protocol [Cheshire,Adoba,Guttman'02]
  - New IETF standard for dynamic network self-configuration
  - Link-local (no routers within the interface)
  - No need for an active DHCP server
  - Aimed at home networks, wireless ad-hoc networks, hand-held devices
  - "Plug and play"
- Self-configuration
  - Performs assignment of IP addresses
  - Symmetric, distributed protocol
  - Uses random choice and timing delays

IPv4 Zeroconf Standard

- Select an IP address out of 65024 at random
- Send a probe querying if address in use, and listen for 2 seconds
  - If positive reply received, restart
  - Otherwise, continue sending probes and listening (2 seconds)
- If K probes sent with no reply, start using the IP number
  - Send 2 packets, at 2 second intervals, asserting IP address is being used
  - If a conflicting assertion received, either:
    - defend (send another asserting packet)
    - defer (stop using the IP address and restart)

Will it work?

- Possible problem…
  - IP number chosen may be already in use, but:
    - Probes or replies may get lost or delayed (host too busy)
- Issues:
  - Self-configuration delays may become unacceptable
    - Would you wait 8 seconds to self-configure your PDA?
  - No justification for parameters
    - for example K=4 in the standard
- Case studies:
  - DTMC and Markov reward models, analytical [BvdSHV03,AK03]
  - TA model using UPPAAL [ZV02]
  - PTA model with digital clocks using PRISM [KNS03]

The IPv4 Zeroconf protocol model

- Modelled using Probabilistic Timed Automata (with digital clocks)
- Parallel composition of two PTAs:
  - one (joining) host, modelled in detail
  - environment (communication medium + other hosts)
- Variables:
  - K (number of probes sent before the IP address is used)
  - the probability of message loss
  - the number of other hosts already in the network

Expected costs

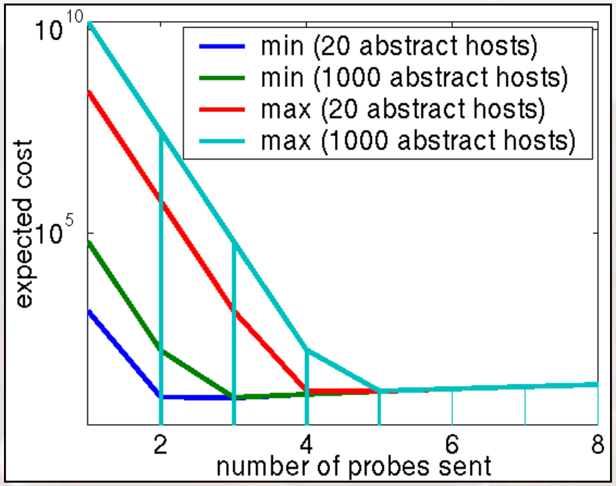
- Compute minimum/maximum expected cost accumulated before obtaining a valid IP address?
  - Implement algorithms of [de Alfaro97] (stochastic shortest path problems for finite-state MDPs)

- Costs:
  - Time should be costly: the host should obtain a valid IP address as soon as possible
  - Using an IP address that is already in use should be very costly: minimise probability of error

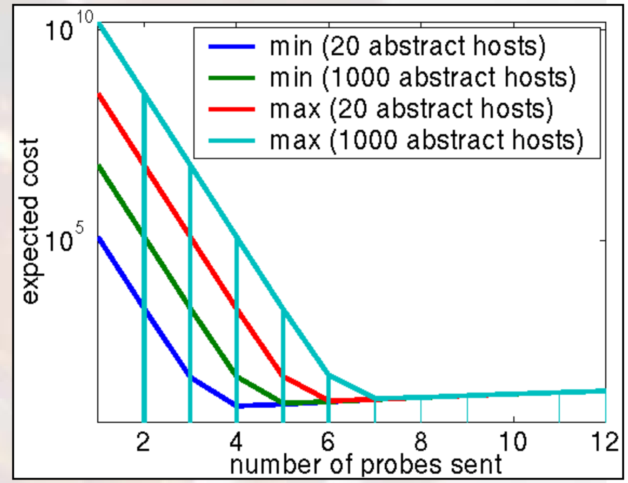
- Cost pair: (r,e)
  - r=1 (t time units elapsing corresponds to a cost of t)
  - $e=10^{12}$ for the event corresponding to using an address which is already in use
  - e=0 for all other events

Results for IPv4 Zeroconf

Prob. of message loss = 0.001



Prob. of message loss = 0.01

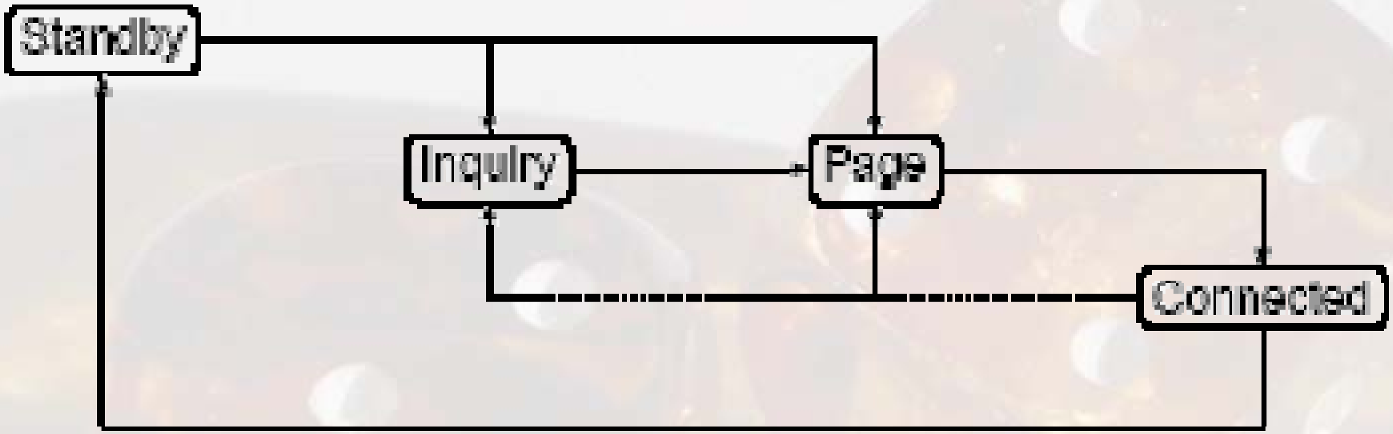


- Sending a high number of probes increases the cost
  - increases delay before a fresh IP address can be used
- Sending a low number of probes increases the cost
  - increases probability of using an IP address already in use
- Similar results to the simpler model of [BvdSHV03]

Case Study: Bluetooth protocol

- Short-range low-power wireless protocol
  - Personal Area Networks (PANs)
  - Open standard, versions 1.1 and 1.2
  - Widely available in phones, PDAs, laptops, …
- Uses frequency hopping scheme
  - To avoid interference (uses unregulated 2.4GHz band)
  - Pseudo-random frequency selection over 32 of 79 frequencies
  - Inquirer hops faster
  - Must synchronise hopping frequencies
- Network formation
  - Piconets (1 master, up to 7 slaves)
  - Self-configuring: devices discover themselves
  - Master-slave roles

States of a Bluetooth device



- Master looks for device, slave listens for master
- Standby: default operational state
- Inquiry: device discovery
- Page: establishes connection
- Connected: device ready to communicate in a piconet

Frequency hopping

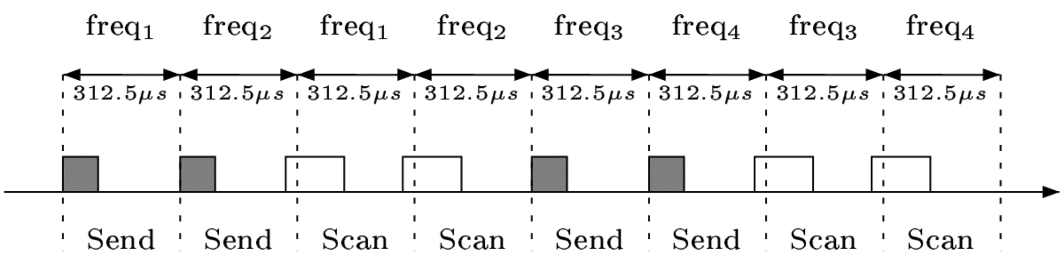


Fig. 1. Timing of the inquiring device's behaviour

- Clock CLK, 28 bit free-running, ticks every 312.5µs
- Inquiring device (master) broadcasts inquiry packets on two consecutive frequencies, then listens on the same two (plus margin)
- Potential slaves want to be discovered, scan for messages
- Frequency sequence determined by formula, dependent on bits of clock CLK (k defined on next slide):

$$freq = [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \bmod 16] \bmod 32$$

Frequency hopping sequence

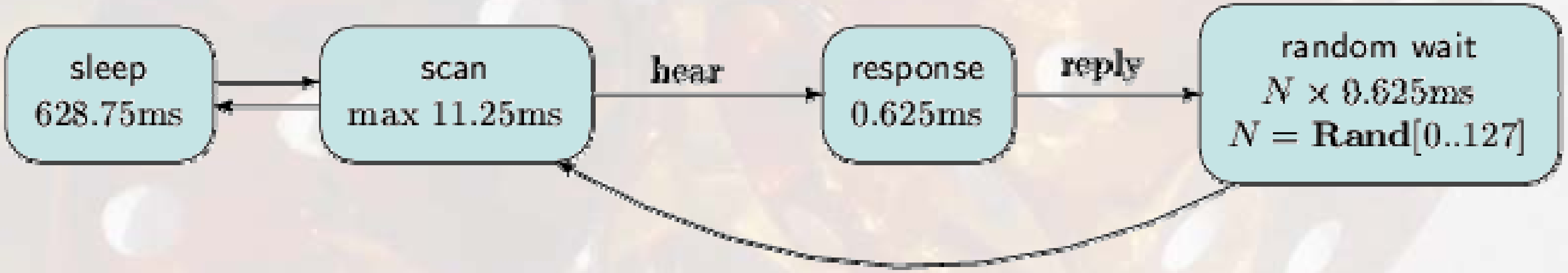
$freq = [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \bmod 16] \bmod 32$

- Two trains (=lines)
- k is offset that determines which train
- Swaps between trains every 2.56 sec
- Each line repeated 128 times

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

Sending and receiving in Bluetooth

- Sender: broadcasts inquiry packets, sending according to the frequency hopping sequence, then listens, and repeats
- Receiver: follows the frequency hopping sequence, own clock



- Listens continuously on one frequency
- If hears message sent by the sender, then replies on the same frequency
- Random wait to avoid collision if two receivers hear on same frequency

Bluetooth modelling

- Very complex interaction
  - Genuine randomness, probabilistic modelling essential
  - Devices make contact only if listen on the right frequency at the right time!
  - Sleep/scan periods unbreakable, much longer than listening
  - Cannot scale constants (approximate results)
  - Cannot omit subactivities, otherwise oversimplification
- Huge model, even for one sender and one receiver!
  - Initial configurations dependent on 28 bit clock
  - Cannot fix start state of receiver, clock value could be arbitrary
  - 17,179,869,184 possible initial states
- But is a realistic future ubiquitous computing scenario!

What about other approaches?

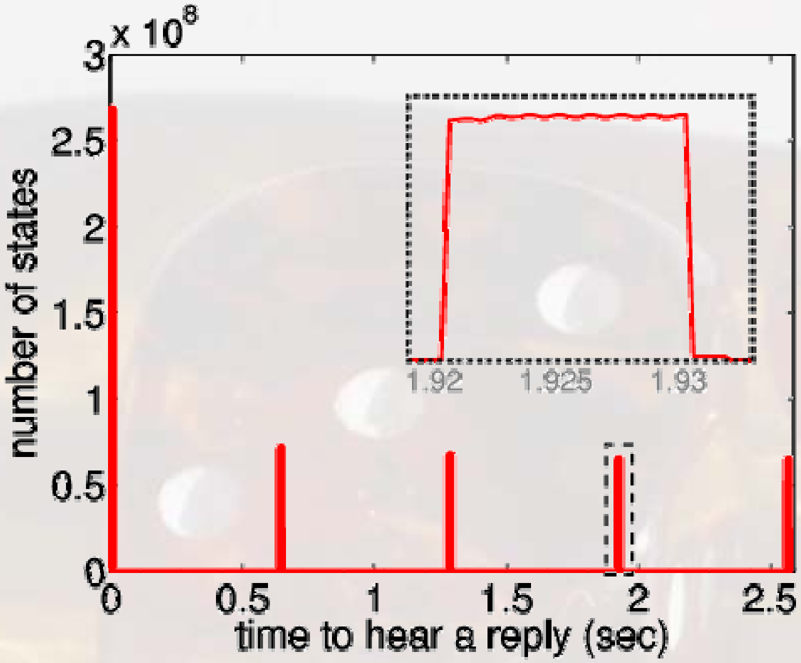
- Indeed, others have tried…
  - network simulation tools (BlueHoc)
  - analytical approaches
- But
  - simulations obtain averaged results, in contrast to best/worst case analysis performed here
  - analytical approaches require simplifications to the model
  - it is easy to make incorrect probabilistic assumptions, as we can demonstrate
- There is a case for all types of analyses, or their combinations…

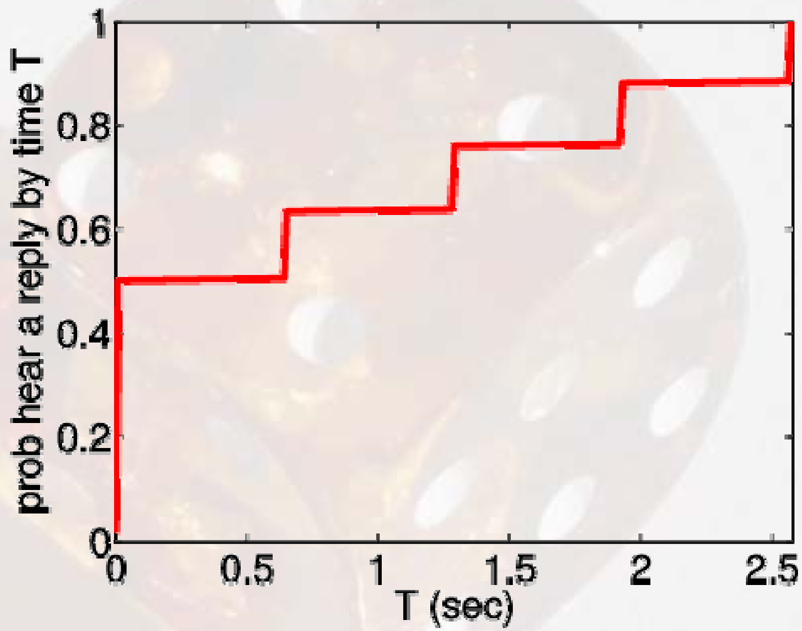
Lessons learnt…

- Must optimise/reduce model
  - Assume negligible clock drift
  - Discrete time, obtain a DTMC
  - Manual abstractions, combine transitions, etc
  - Divide into 32 separate cases
  - Success (exhaustive analysis) with one/two replies

- Observations
  - Work with realistic constants, as in the standard
  - Analyse v1.2 and 1.1, confirm 1.1 slower
  - Show best/worst case values, can pinpoint scenarios which give rise to them
  - Also obtain power consumption analysis

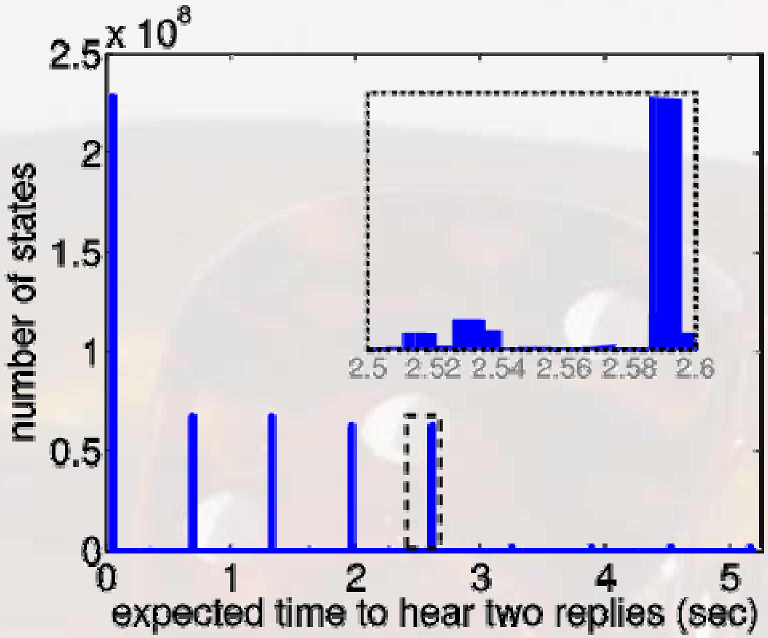
Time to hear 1 reply

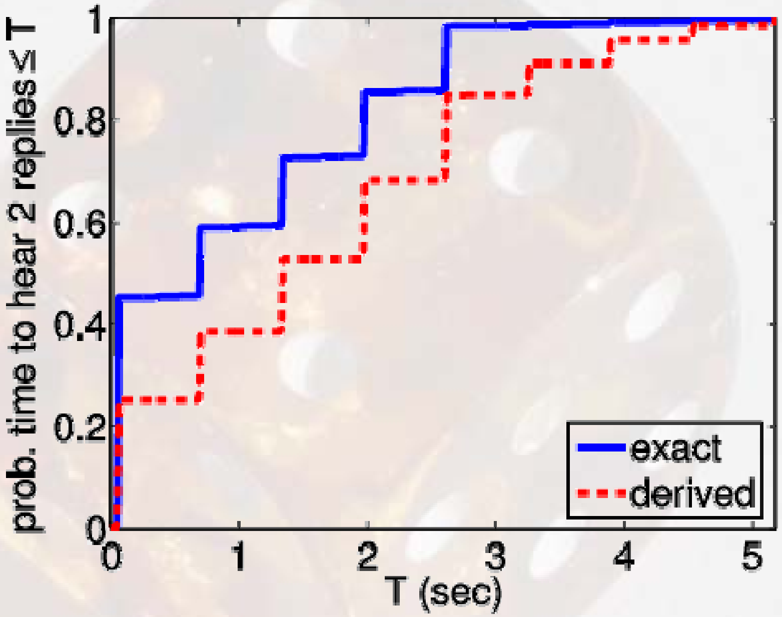




- Max time to hear is 2.5716sec, in 921,600 possible initial states, (Min 635μs)
- Cumulative: assume uniform distribution on states when receiver first starts to listen

Time to hear 2 replies





- Max time to hear is 5.177sec (16,565 slots), in 444 possible initial states
- Cumulative (derived): assumes time to reply to 2nd message is independent of time to reply to 1st (incorrect, compare with exact curve obtained from model checking)

Case Study: Contract Signing

- Two parties want to agree on a contract
- Each will sign if the other will sign
  - Cannot trust other party in the protocol
  - There may be a trusted third party (judge), but it should only be used if something goes wrong
- Contract signing with pen and paper
  - Sit down and write signatures simultaneously
- Contract signing on the Internet
  - Challenge: how to exchange commitments on an asynchronous network?

Contract Signing [Norman,Shmatikov]

Partial secret exchange protocol of Even, Goldreich and Lempel (1985) for two parties (**A** and **B**)

- **A** (**B**) holds secrets $a_1,\ldots,a_{2n}$ ($b_1,\ldots,b_{2n}$)
  - Secret is a binary string of length l
  - Secrets partitioned into pairs:
    $\{(a_i, a_{n+i}) \mid i=1,\ldots,n\}$ and $\{(b_i, b_{n+i}) \mid i=1,\ldots,n\}$
  - **A** (**B**) committed if **B** (**A**) knows one of **A**'s (**B**'s) pairs
- Uses **1-out-of-2 oblivious transfer protocol**: **OT(S,R,x,y)**
  - **S** sends **x** and **y** to **R**
  - **R** receives **x** with probability ½ otherwise receives **y**
  - **S** does not know which one **R** receives
  - if **S** cheats then **R** can detect this with probability ½

Contract Signing

(step 1)

for i=1,…,n

    OT(A,B, a_i, a_{n+i})

    OT(B,A b_i, b_{n+i})

end

(step 2)

for i=1,…,l (l is the bit length of the secrets)

    for j=1,…,2n

        A transmits bit i of secret a_j to B

    end

    for j=1,…,2n

        B transmits bit i of secret b_j to A

    end

end

Results: Contract Signing

- Discovered a weakness in the protocol when party **B** is allowed to act maliciously by quitting the protocol early
  - this behaviour not considered in the original analysis
- **PRISM** analysis shows:
  - if **B** stops participating in the protocol as soon as he/she has obtained at least one of **A** pairs, then, with probability 1, at this point:
    - **B** possesses a pair of **A**'s secrets
    - **A** does not have complete knowledge of any pair of **B**'s secrets
- Protocol is therefore not fair under this attack:
  - **B** has a distinct advantage over **A**

Results: Contract Signing

- The protocol is unfair because in step 2: **A** sends a bit for each of its secret before **B** does.
- Can we make this protocol fair by changing the message sequence scheme?
- Since the protocol is asynchronous the best we can hope for is with probability $\frac{1}{2}$ **B** (or **A**) gains this advantage
- We consider 3 possible alternate message sequence schemes…

Contract Signing: EGL2

(step1)

…

(step2)

for i=1,…,l

  for j=1,…,n A transmits bit i of secret a_j to B

  for j=1,…,n B transmits bit i of secret b_j to A

end

for i=1,…,l

  for j=n+1,…,2n A transmits bit i of secret a_j to B

  for j=n+1,…,2n B transmits bit i of secret b_j to A

end

Contract Signing: EGL3

(step1)

…

(step2)

for i=1,…,l for j=1,…,n

A transmits bit i of secret a_j to B

B transmits bit i of secret b_j to A

end

for i=1,…,l for j=n+1,…,2n

A transmits bit i of secret a_j to B

B transmits bit i of secret b_j to A

end

Contract Signing: EGL4

(step1)

…

(step2)

for i=1,…,l

A transmits bit **i** of secret a_1 to **B**

for j=1,…,n B transmits bit **i** of secret b_j to A

for j=2,…,n A transmits bit **i** of secret a_j to **B**

end

for i=1,…,l

A transmits bit **i** of secret a_{n+1} to **B**

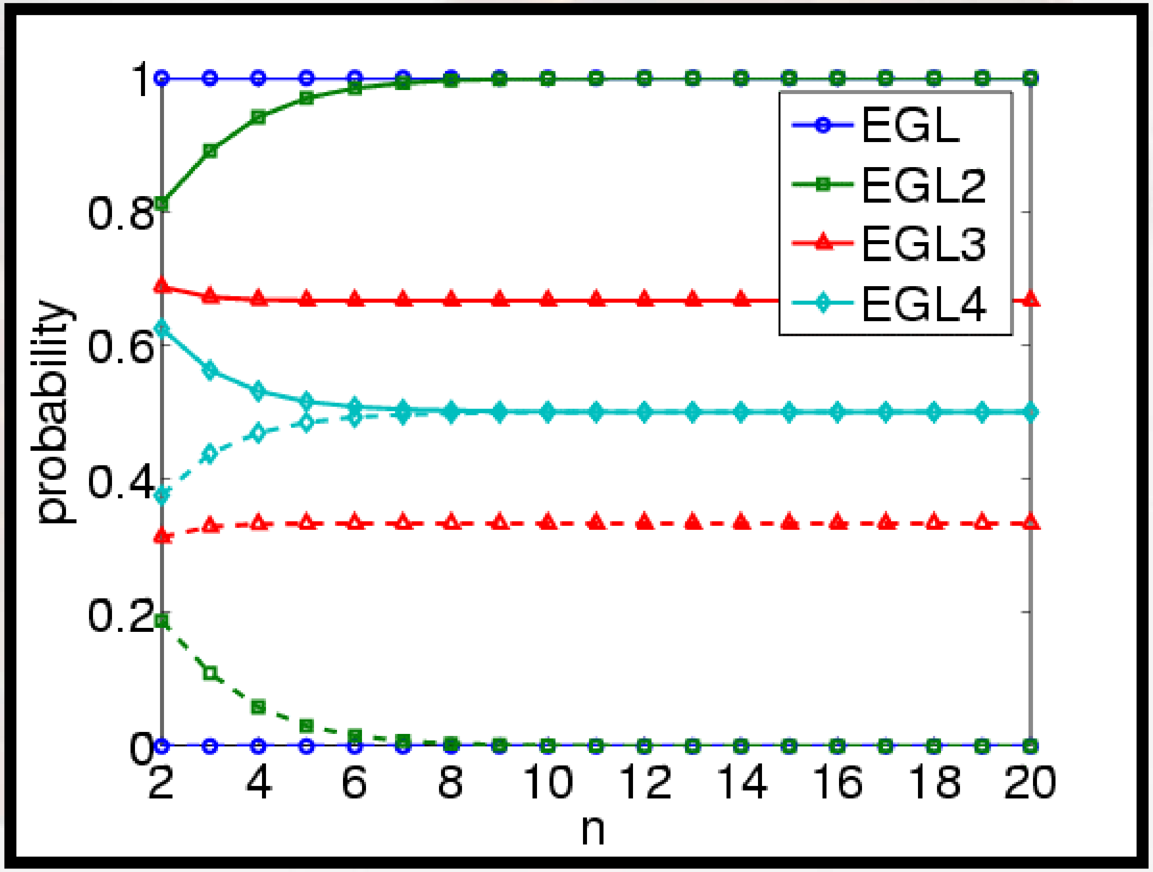
for j=n+1,…,2n B transmits bit **i** of secret b_j to A

for j=n+2,…,2n A transmits bit **i** of secret a_j to **B**

end

Results: Contract Signing

- **Probability the other party gains knowledge first**
  - The chance that the protocol is unfair

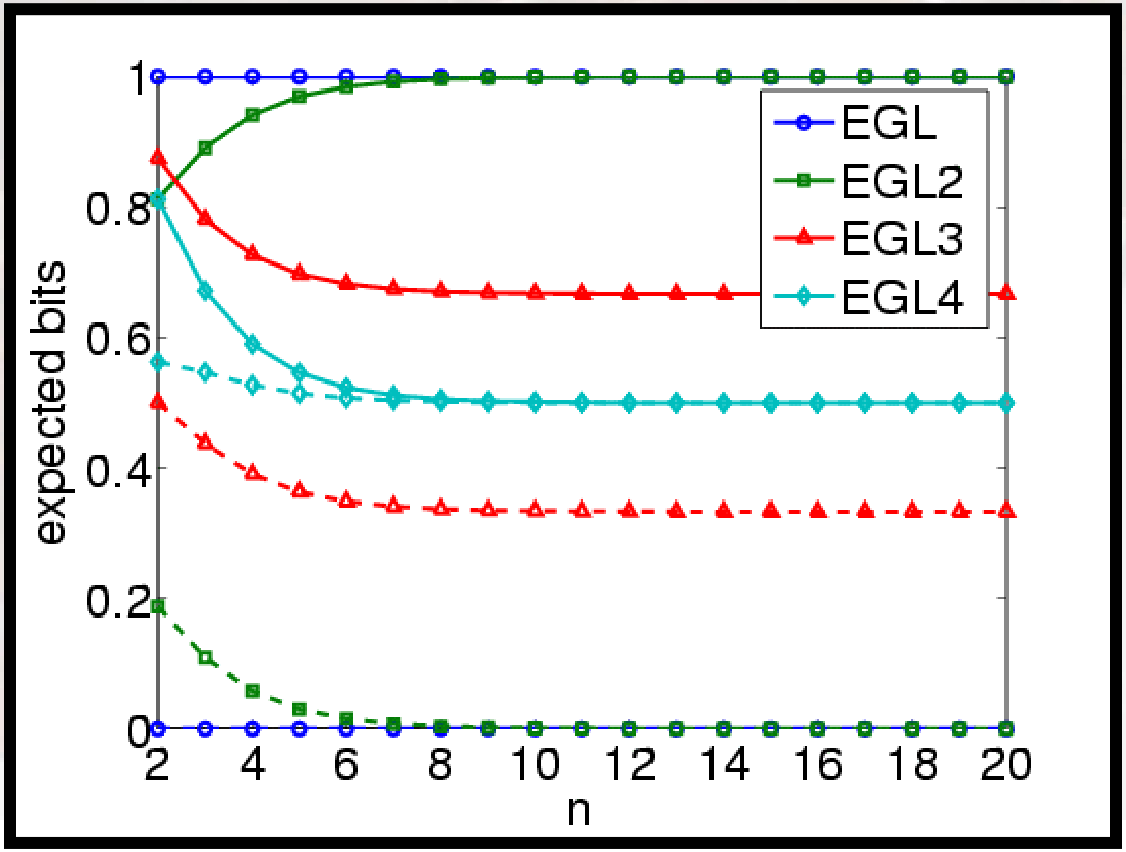


A solid line

B dashed line

Results: Contract Signing

- **Expected bits a party requires to know a pair once the other knows a pair**
  - quantifies how unfair the protocol is

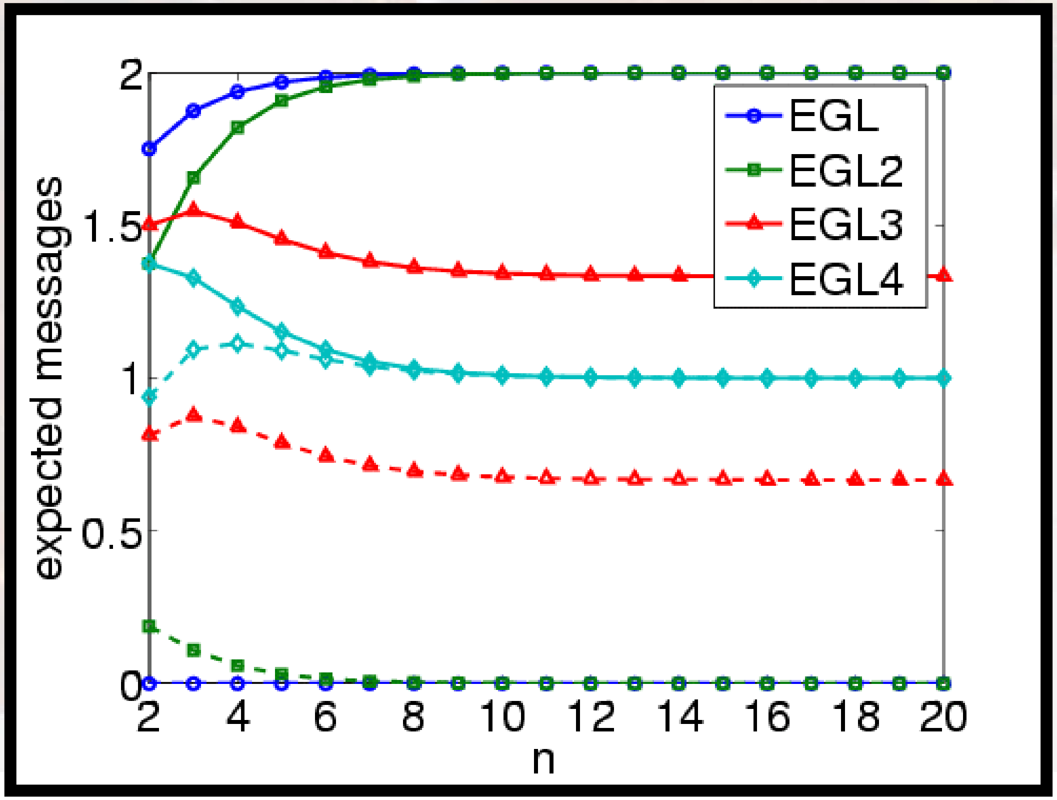


A solid line
B dashed line

Results: Contract Signing

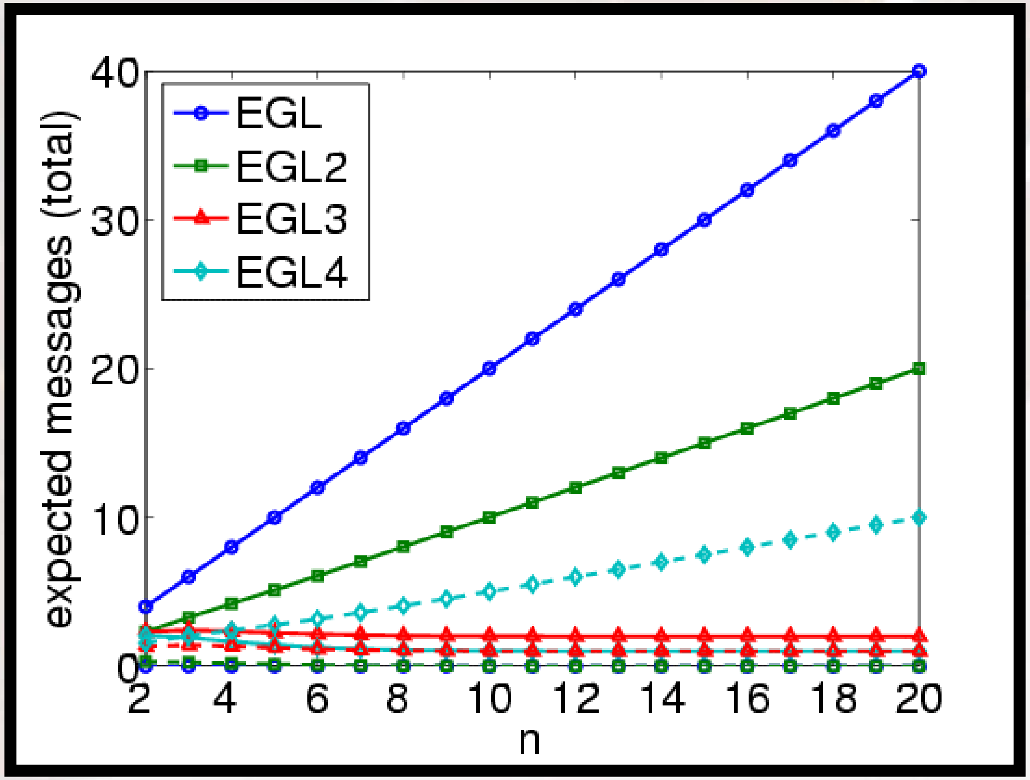
- **Expected messages a party must receive to know a pair once the other knows a pair**
  - measures the influence the other party has on the fairness, since it can try and delay these messages

A solid line
B dashed line



Results: Contract Signing

- **Expected messages that need to be sent for a party to know a pair once the other party knows a pair**
  - measures the duration of unfairness



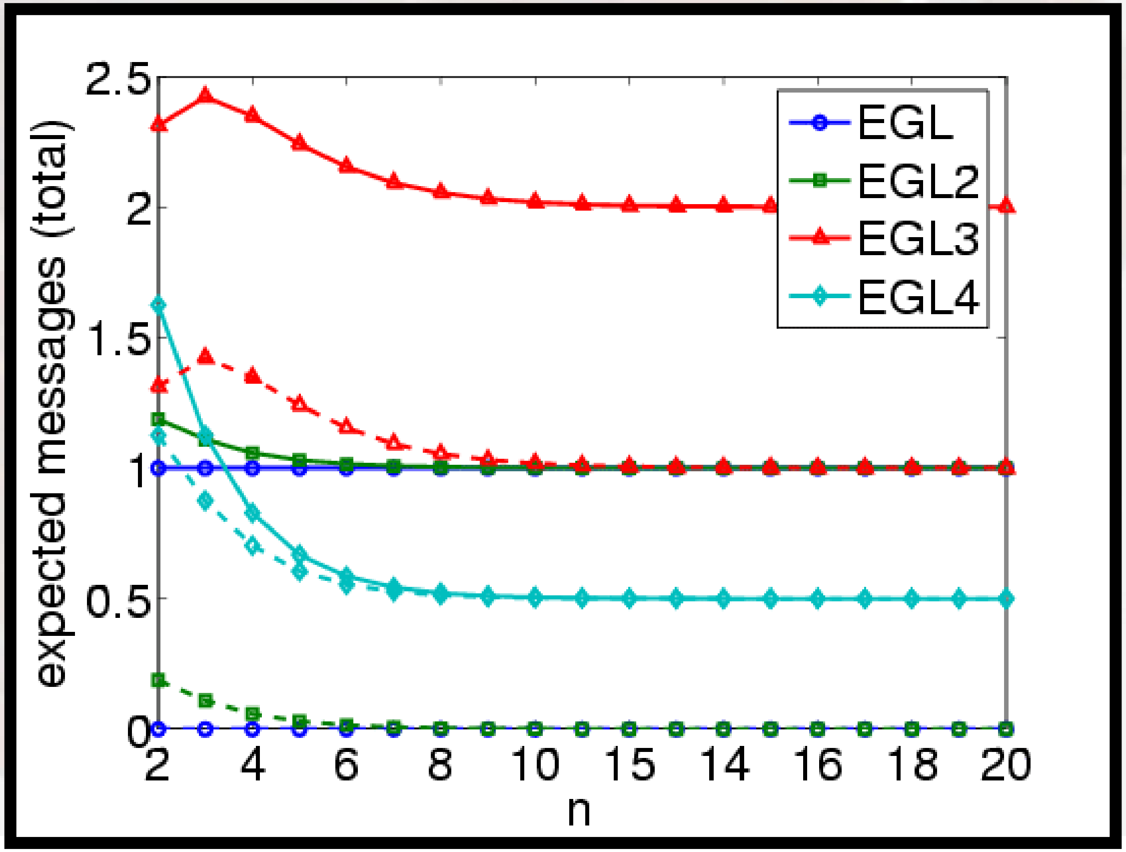
A solid line
B dashed line

Results: Contract Signing

- Results show EGL4 is the 'fairest' protocol
- Except for duration of fairness measure:
  Expected messages that need to be sent for a party to know a pair once the other party knows a pair
  - this value is larger for **B** than for **A**
  - and, in fact, as **n** increases, this measure:
    - increases for **B**
    - decreases for **A**
- Solution: if a party sends a sequence of bits in a row (without the other party sending messages in between), require that the party send these bits as as a single message

Results: Contract Signing

- **Expected messages that need to be sent for a party to know a pair once the other party knows a pair**
  - measures the duration of unfairness



A solid line
B dashed line

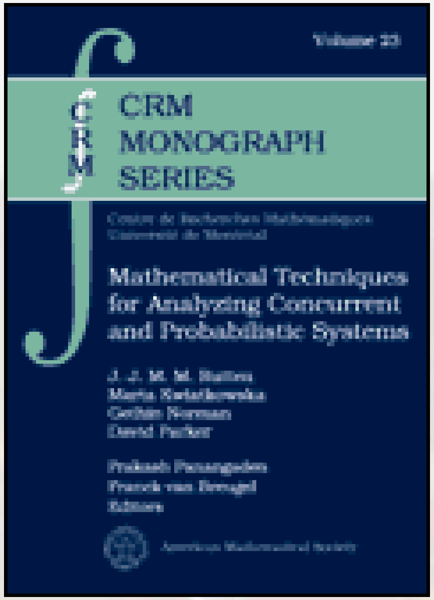
Related projects

- FORWARD (this case study, see ISOLA'04)
  - Performance modelling of MAC layer of Bluetooth
  - Security analysis of Bluetooth
- Modelling and verification of mobile ad hoc network protocols
  - Modelling language with mobility and randomisation
  - Model checking algorithms & techniques
  - Tool development & implementation
  - Modelling timing properties of AODV
- Focus on properties
  - "probability of delivery within time deadline is …"
  - "expected time to device discovery is …"
  - "expected power consumption is …"

Challenges for future

- Exploiting structure
  - Abstraction, data reduction, compositionality…
  - Parametric probabilistic verification?
- Proof assistant for probabilistic verification
- Extension for mobility
- Extension for hybrid systems
- Simulation, statistical testing [Younes]
- Approximation methods
- Continuous PTAs
  - Efficient model checking methods?
- More expressive specifications
  - Probabilistic LTL/PCTL*/mu-calculus?
- Real software, not models!

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 2.0 (approx. 1000 users)
- Publications by others and courses that feature PRISM…

PRISM Contributors