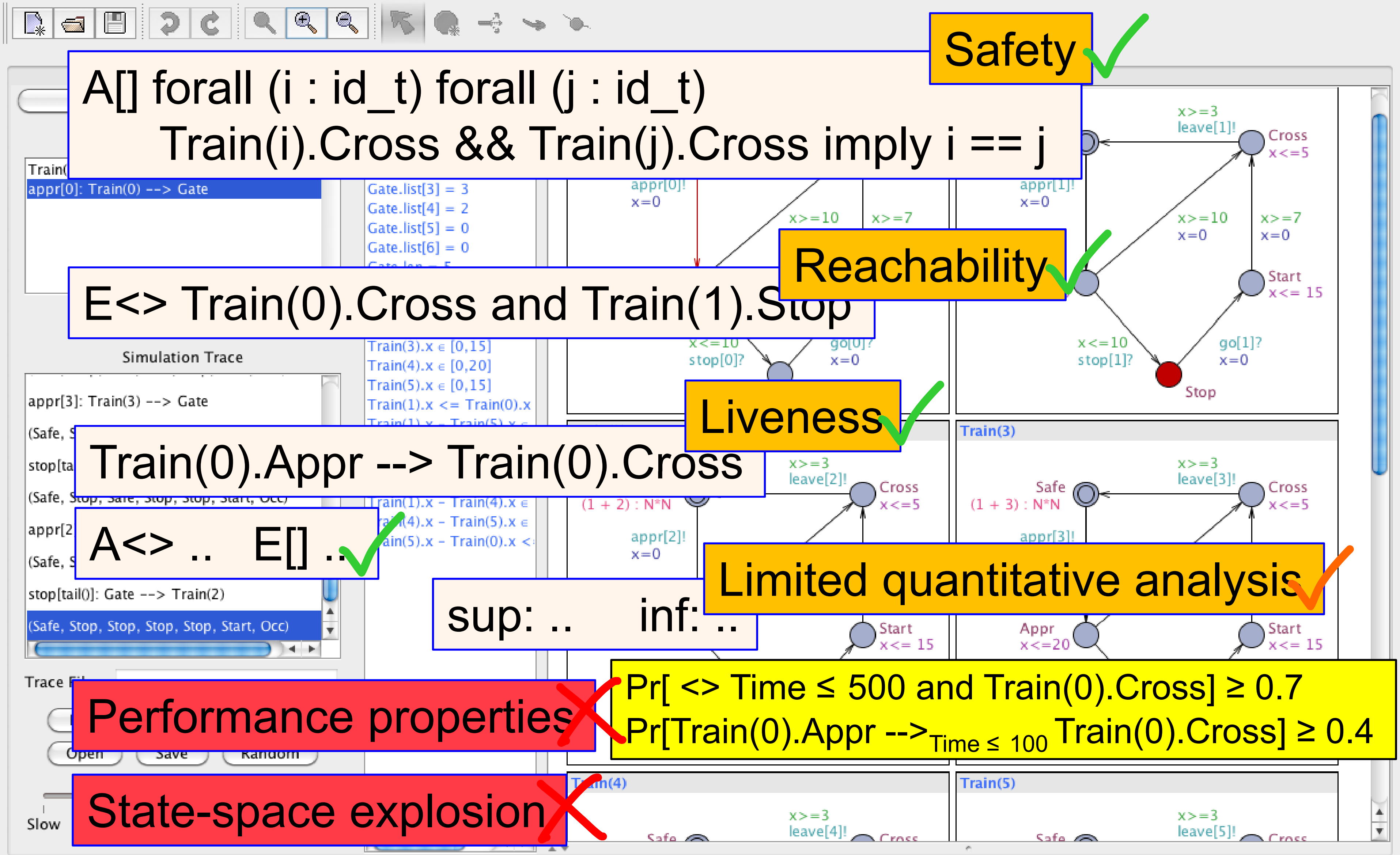
Statistical Model Checking for Timed Automata

Collaborators: Peter Bulychev,
Alexandre David
Axel Legay, Marius Mikucionis
Wang Zheng
Jonas van Vliet, Danny Poulsen

CAV 2011, PDMC 2011,
FORMATS 2011

UPPAAL



A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j

Safety ✓

E<> Train(0).Cross and Train(1).Stop

Reachability ✓

Train(0).Appr --> Train(0).Cross

Liveness ✓

A<> ..   E[] .. ✓

sup: ..   inf: ..

Limited quantitative analysis ✓

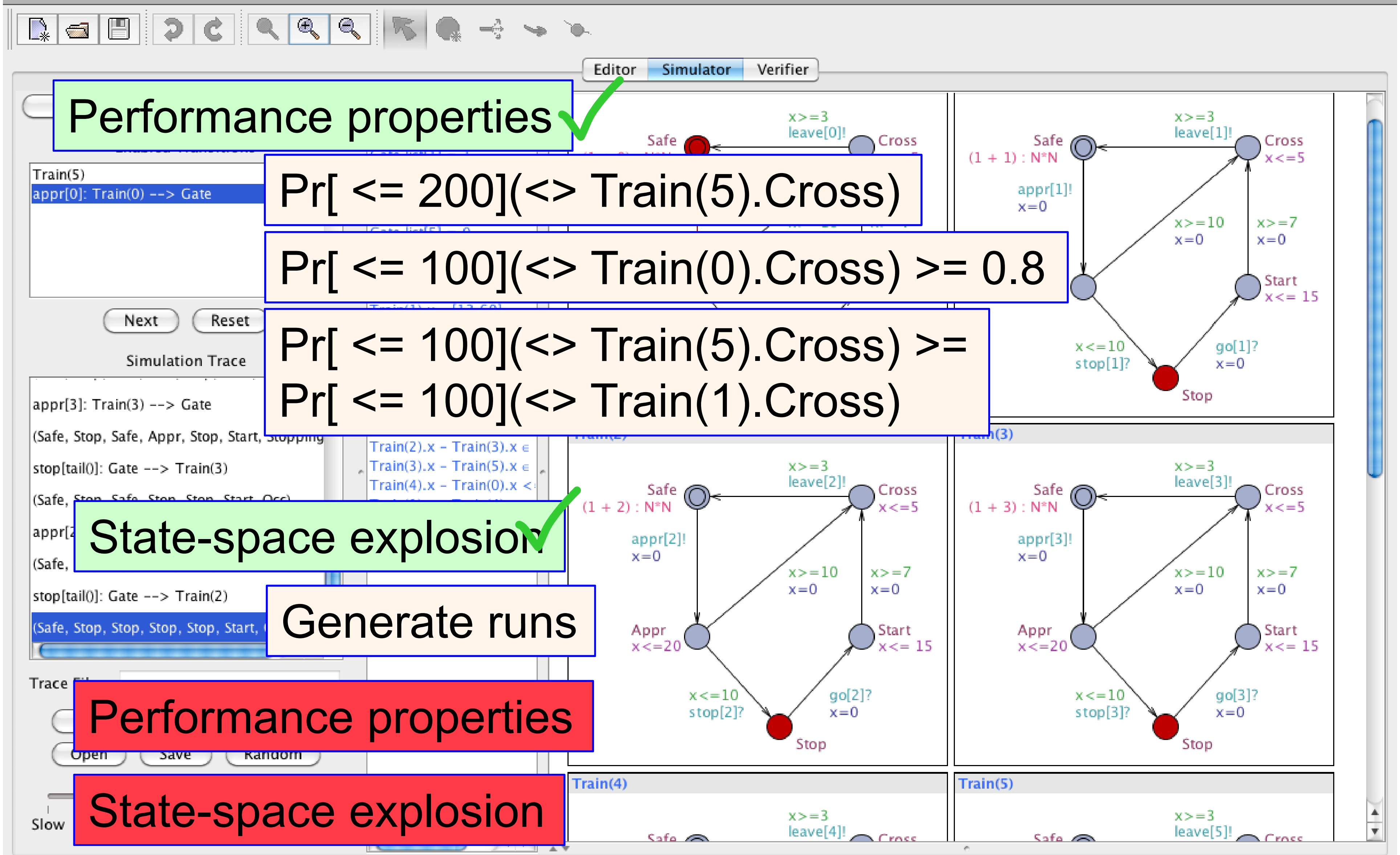
Pr[<> Time ≤ 500 and Train(0).Cross] ≥ 0.7

Pr[Train(0).Appr -->$_{Time \leq 100}$ Train(0).Cross] ≥ 0.4

Performance properties ✗

State-space explosion ✗

UPPAAL SMC



Performance properties ✓

Pr[<= 200](<> Train(5).Cross)

Pr[<= 100](<> Train(0).Cross) >= 0.8

Pr[<= 100](<> Train(5).Cross) >= Pr[<= 100](<> Train(1).Cross)

State-space explosion ✓

Generate runs

Performance properties

State-space explosion

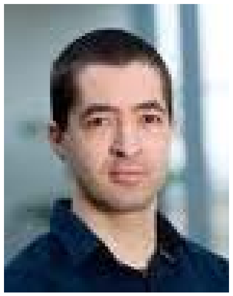
Overview

- Statistical Model Checking in UPPAAL
  - Estimation
  - Testing
- Distributed SMC for Parameterized Models
  - Parameter Sweeps
  - Optimization
  - Nash Equilibria
- Distributing Statistical Model Checking
  - Estimation
  - Testing
- Parameter Analysis of DSMC
- Conclusion

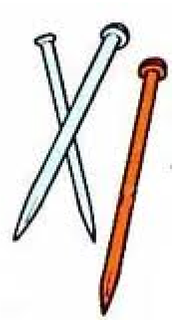
Overview

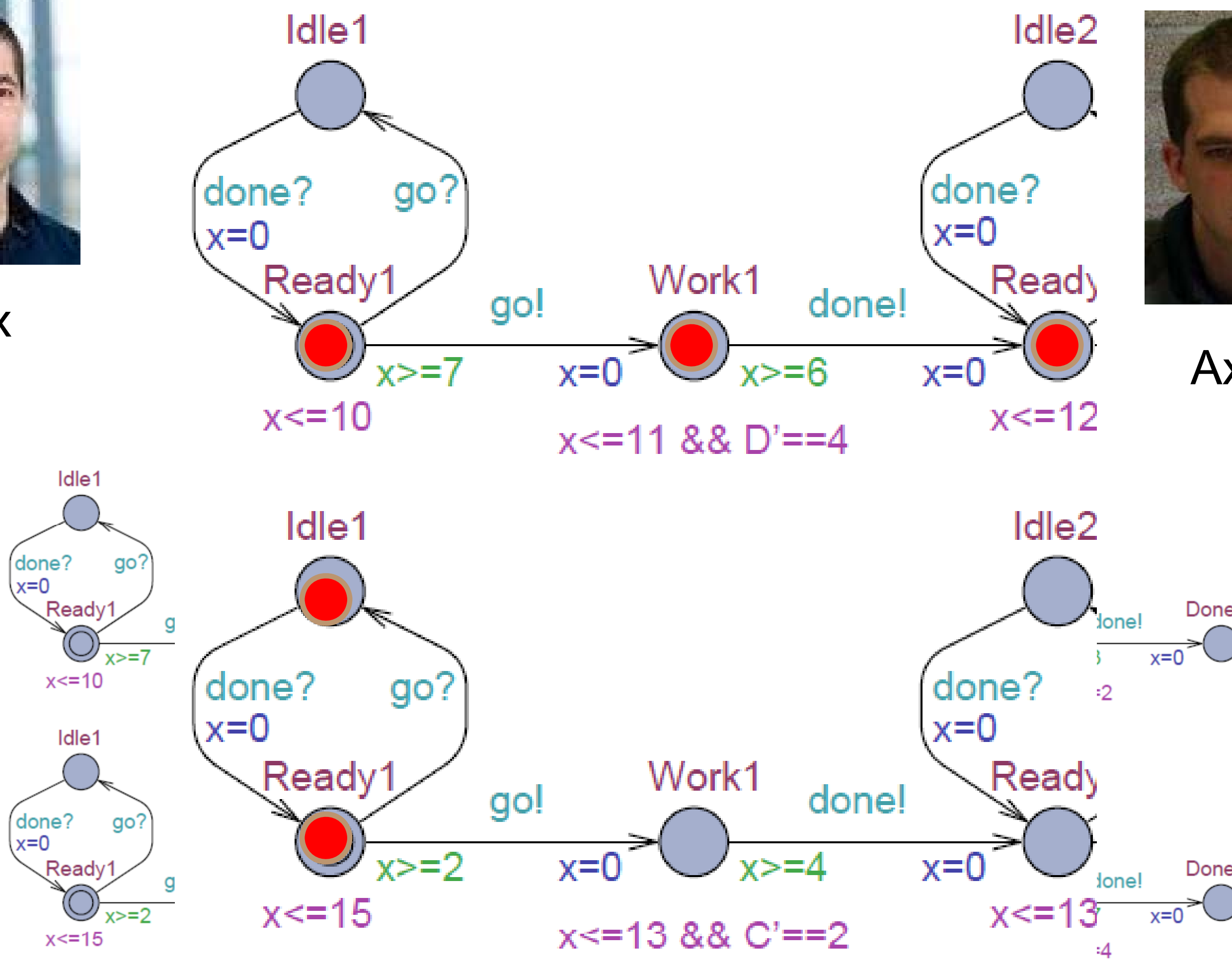
- Statistical Model Checking in UPPAAL
  - Estimation
  - Testing
- Distributed SMC for Parameterized Models
  - Parameter Sweeps
  - Optimization
  - Nash Equilibria
- Distributing Statistical Model Checking
  - Estimation
  - Testing
- Parameter Analysis of DSMC
- Conclusion

The Hammer Game



Alex



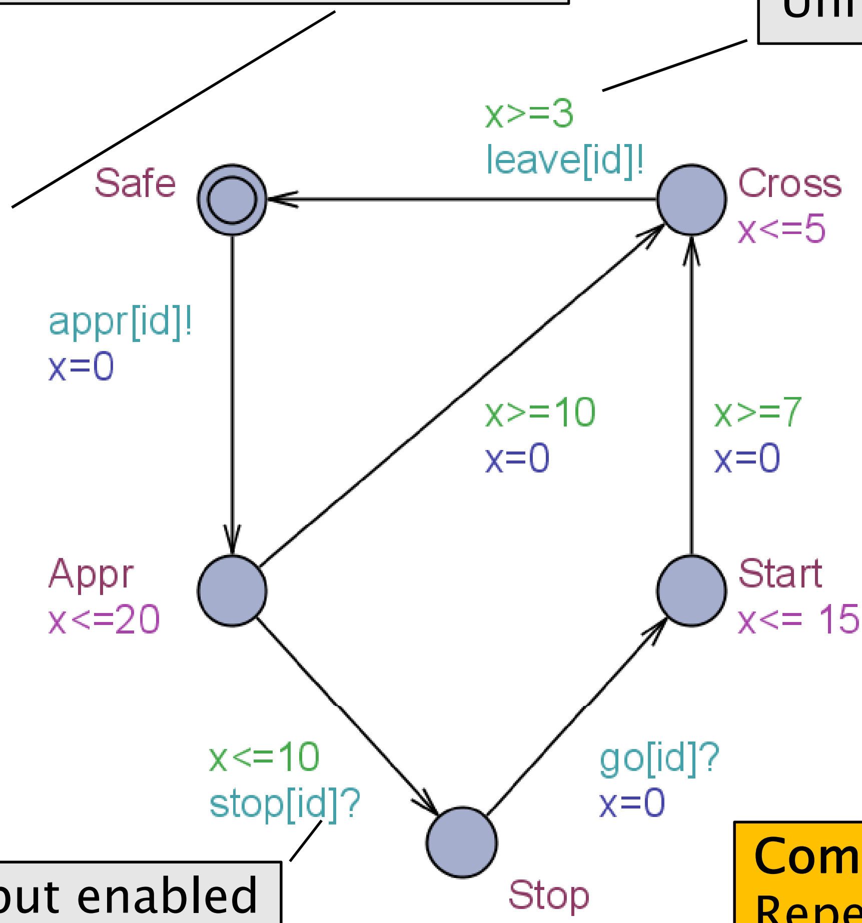


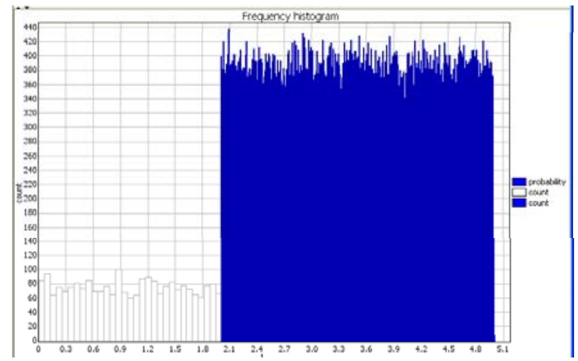
Axel

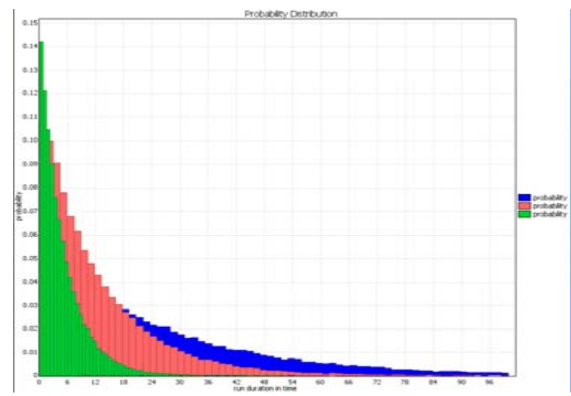
Stochastic Semantics of TA

Exponential Distribution

Uniform Distribution



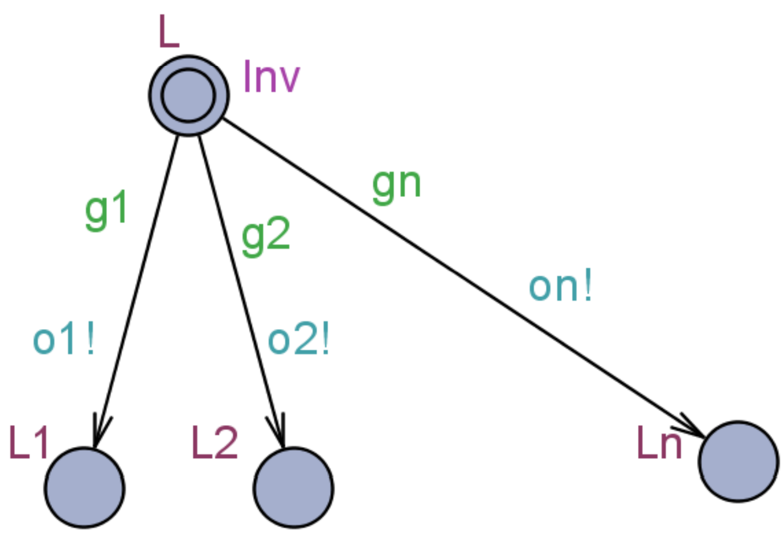


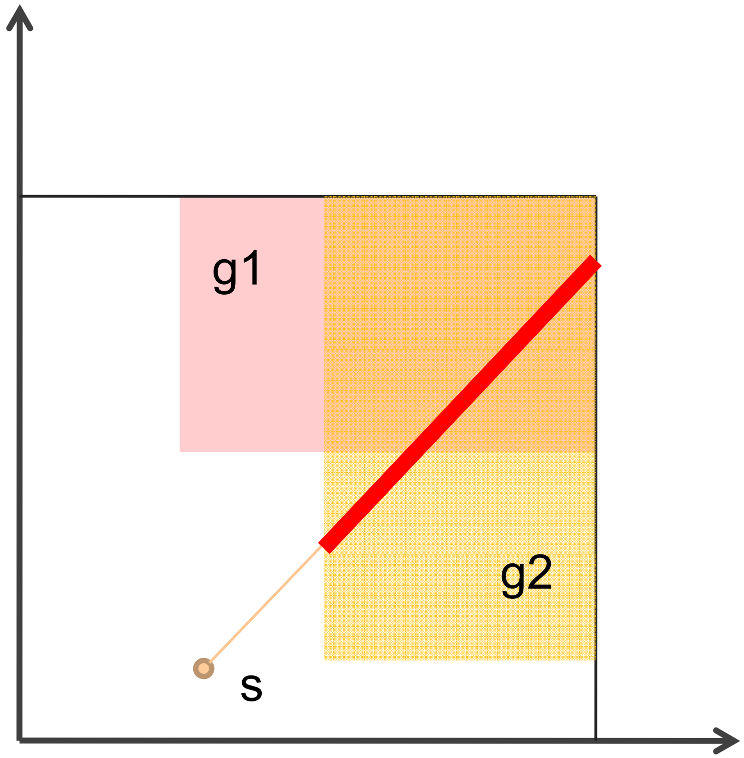


Input enabled

Composition =
Repeated races between components

Stochastic Semantics of Timed Automata





Delay Density Function

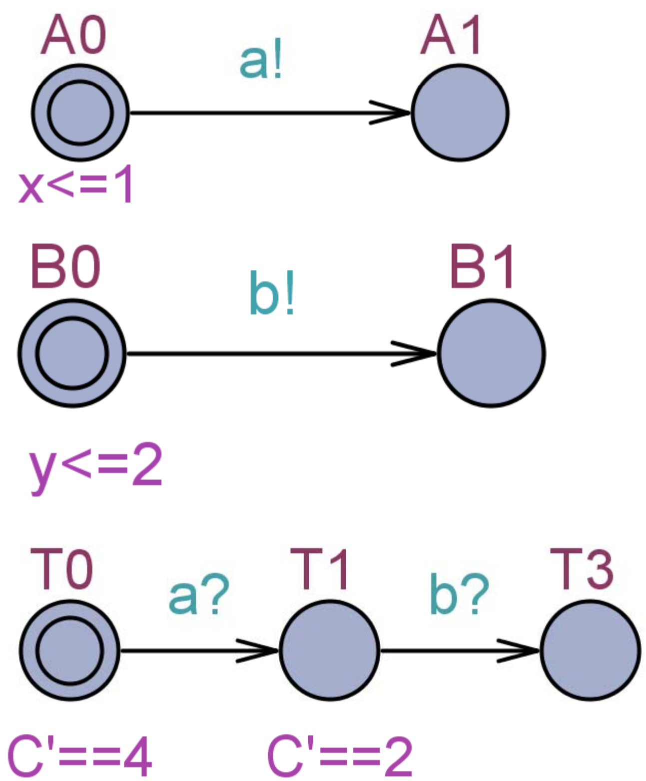
μ_s: **R** → **R**

Output Probability Function

$\gamma_s: \Sigma_o \rightarrow [0,1]$

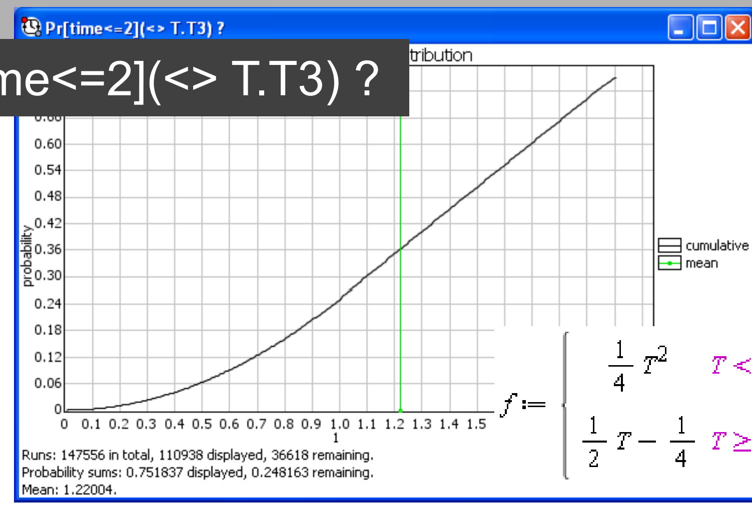
- μ_s uniform on [d_{min}, d_{max}]
- γ_s uniform over enabled outputs

Stochastic Semantics of Timed Automata



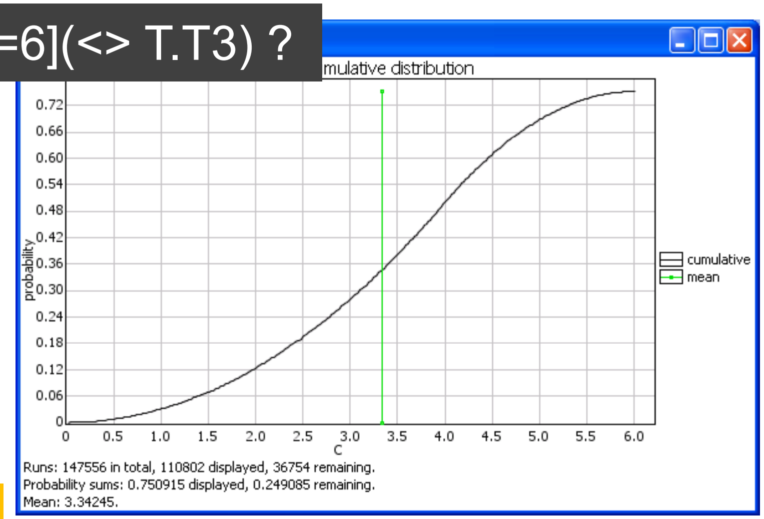
Composition = Race between components for outputting

Pr[time<=2](<> T.T3) ?



$$f := \begin{cases} \frac{1}{4}T^2 & T < 1 \\ \frac{1}{2}T - \frac{1}{4} & T \geq 1 \end{cases}$$

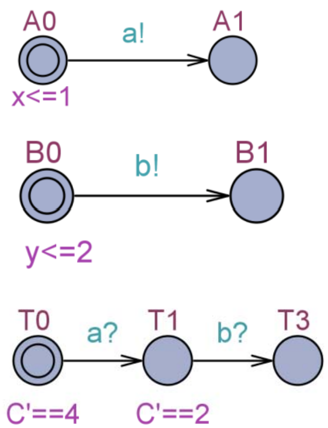
Pr[C<=6](<> T.T3) ?



$$f := c \rightarrow \begin{cases} \frac{1}{32}c^2 & c < 4 \\ \frac{1}{4}c - \frac{1}{2} - \frac{1}{4}\left(\frac{1}{2}c - 2\right)^2 & c \geq 4 \end{cases}$$

Stochastic Semantics of Timed Automata

$\mathcal{A}$



Assumptions:
Component TAs are:

- Input enabled
- Deterministic
- Disjoint set of output actions

π (**s** , a_1 a_2 .... a_n) :
the set of maximal runs from **s** with a prefix
t_1 a_1 t_2 a_2 … t_n a_k
for some $t_1,\ldots,t_n \in$ **R**.

$$\mathbb{P}_{\mathcal{A}}\big(\pi(\mathbf{s},\, a_1 a_2 \ldots a_n)\big) =$$

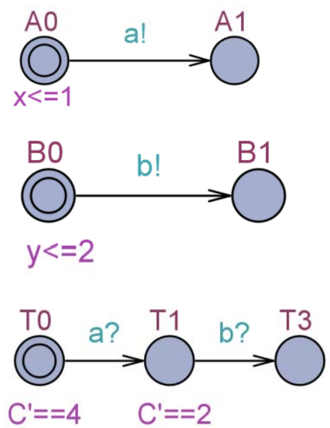
$$\int_{t \geq 0} \mu_{s_c}(t) \cdot \Big(\prod_{j \neq c} \int_{\tau > t} \mu_{s_j}(\tau)\, d\tau\Big) \cdot \gamma_{s_c{}^t}(a_1) \cdot \mathbb{P}_{\mathcal{A}}\big(\pi(\mathbf{s}^t)^{a_1},\, a_2 \ldots a_n)\big)\; dt$$

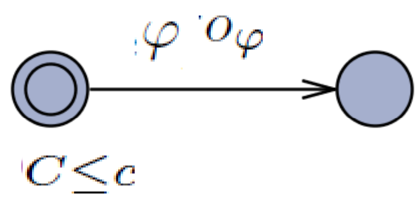
where $c = c(a_1)$, and as base case we take $P_{\mathcal{A}}(\pi(\mathbf{s}), \varepsilon) = 1$.

Logical Properties

$$\psi \ ::= \ \mathbb{P}(\Diamond_{C \leq c} \varphi) \sim p \ \mid \ \mathbb{P}(\Box_{C \leq c} \varphi) \sim p$$

$\mathcal{A}^*$





$$(A|B|T) \models \mathbb{P}(\Diamond_{C \leq 6} T_3) = 0.75$$

$$(A|B|T) \models \mathbb{P}(\Diamond_{t \leq 2} T_3) = 0.75$$

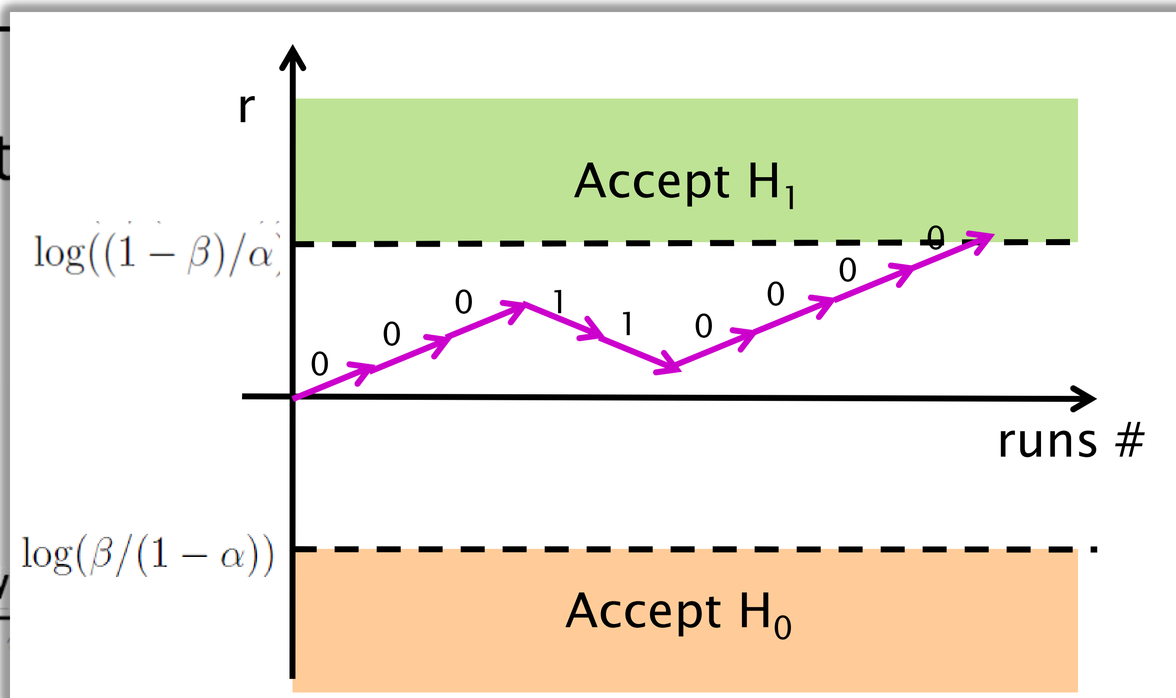
$$\mathcal{A} \models \mathbb{P}(\Diamond_{C \leq c} \varphi) \sim p \ \text{ iff } \ \mathbb{P}_{\mathcal{A}^*}\Big(\bigcup_{\sigma \in \Sigma^*} \pi(s_0, \sigma o_\varphi) \Big) \sim p$$

SMC Algorithms in UPPAAL

Qualitative (Hypot

$$p = \mathbb{P}_{\mathcal{A}}(\Diamond_{C \leq c} \varphi)$$

α: prob of acc H_0 when H_1

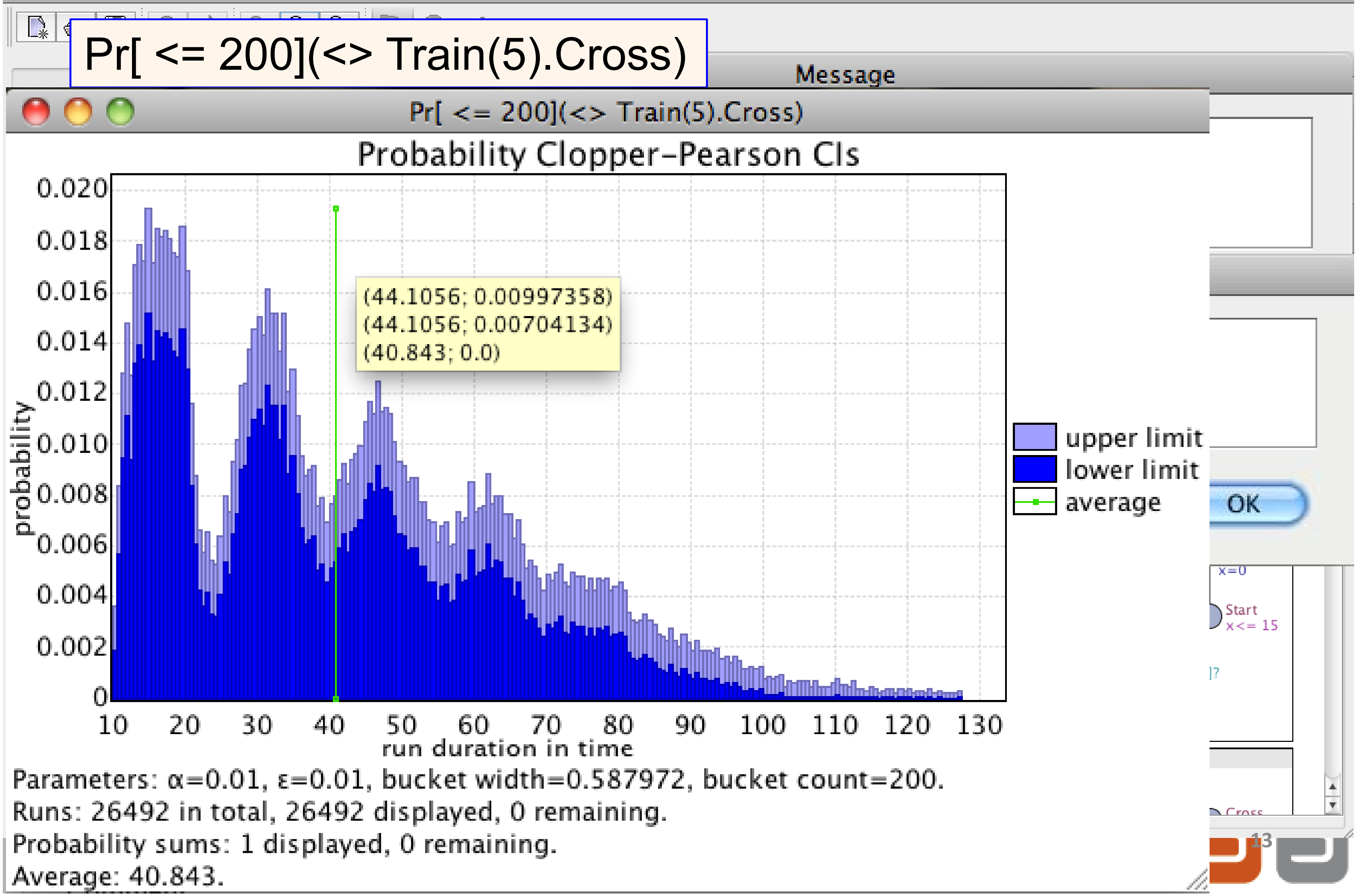


Algorithm II: Sequential Probability

function hypothesis(S:model ,
r:=0
while *true* **do**
  Observe the random variable x corresponding to $\Diamond_{C \leq c} \varphi$ for a run.
  $r := r + x * \log(p_1/p_0) + (1-x) * \log((1-p_1)/(1-p_0))$
  if $r \leq \log(\beta/(1-\alpha))$ **then** accept H_0
  if $r \geq \log((1-\beta)/\alpha)$ **then** accept H_1

11

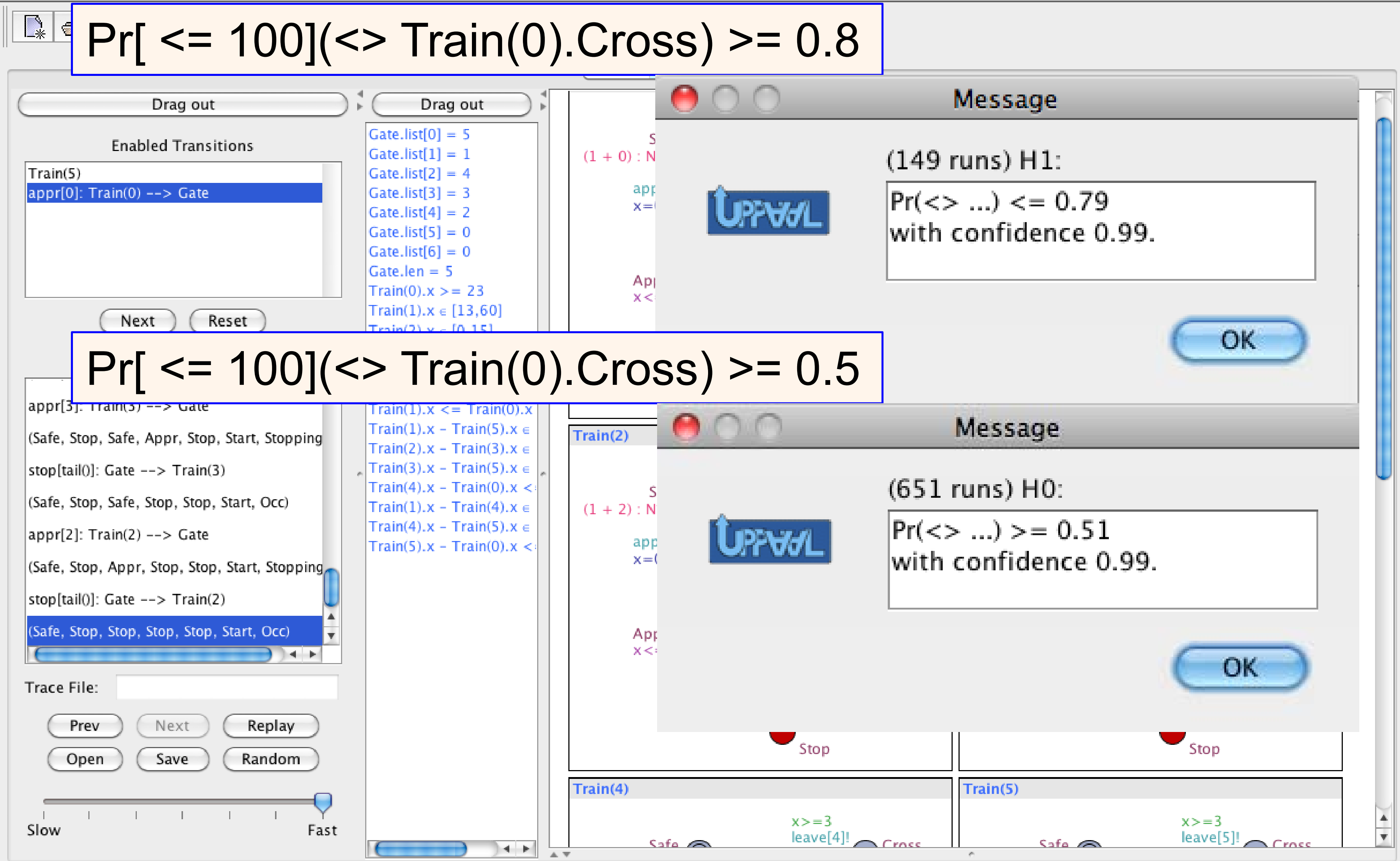
Queries in UPPAAL SMC



Queries in UPPAAL SMC

Pr[<= 100](<> Train(0).Cross) >= 0.8

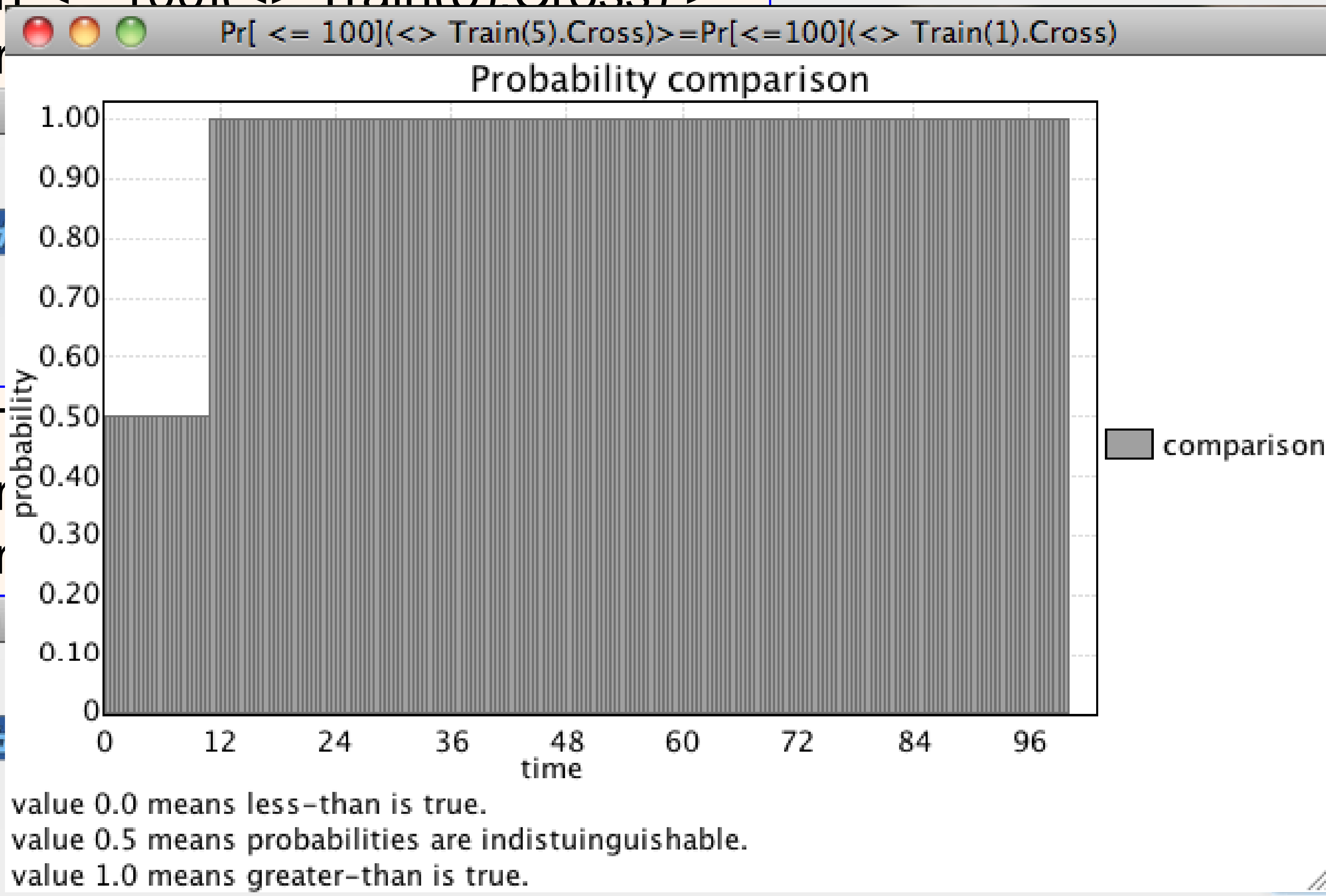
Pr[<= 100](<> Train(0).Cross) >= 0.5



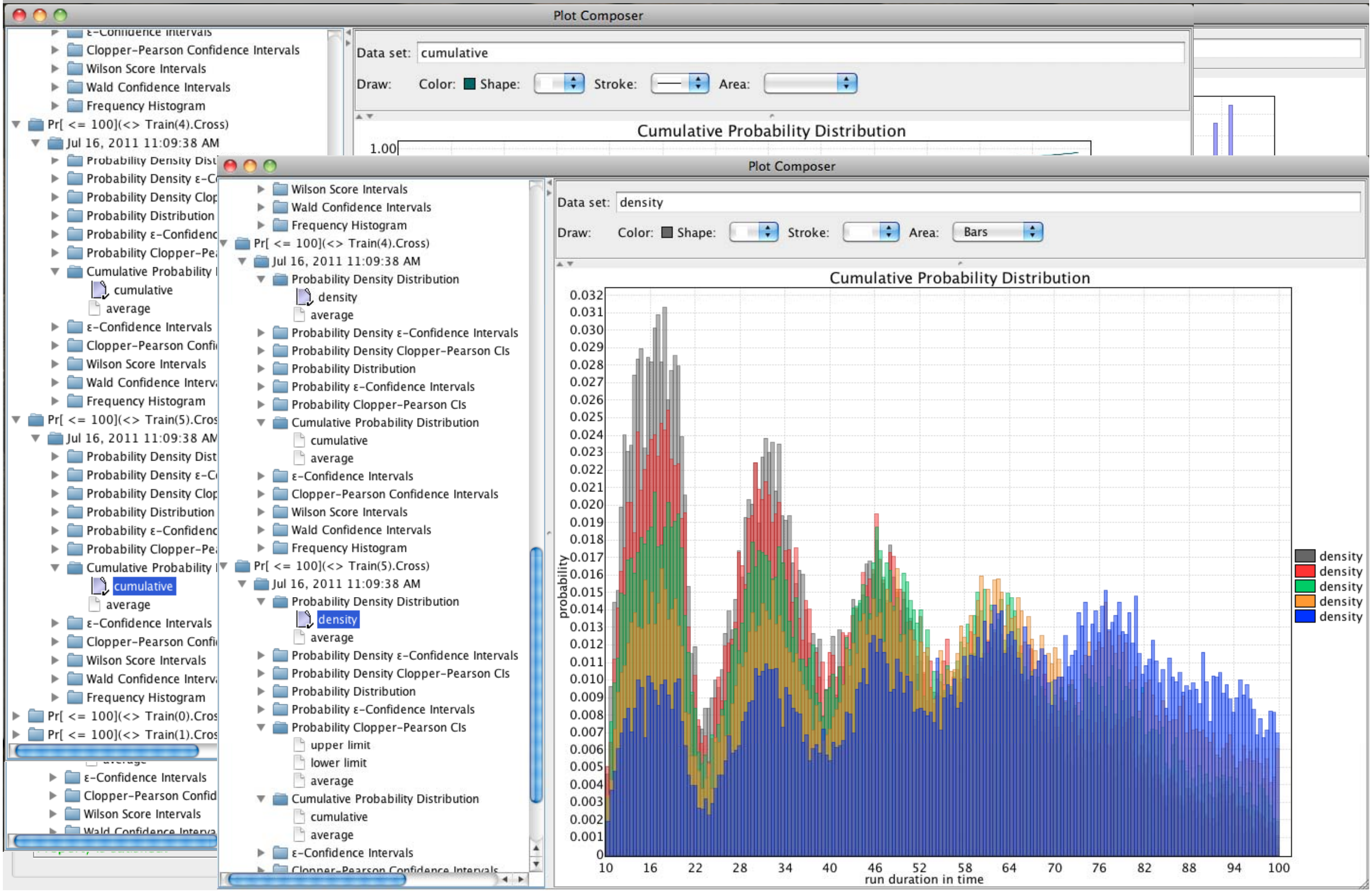
Queries in UPPAAL SMC

Pr[<= 100](<> Train(5).Cross) >=
Pr

∀
Pr
Pr



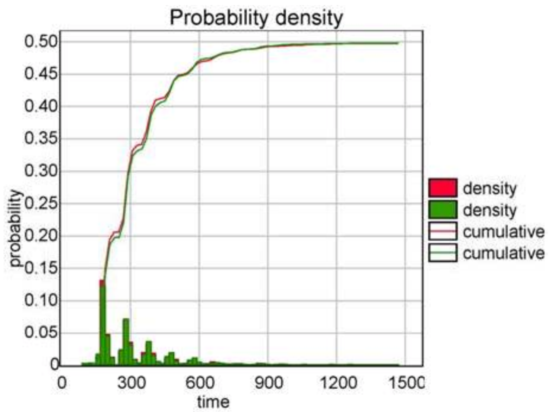
Analysis Tool: Plot Composer

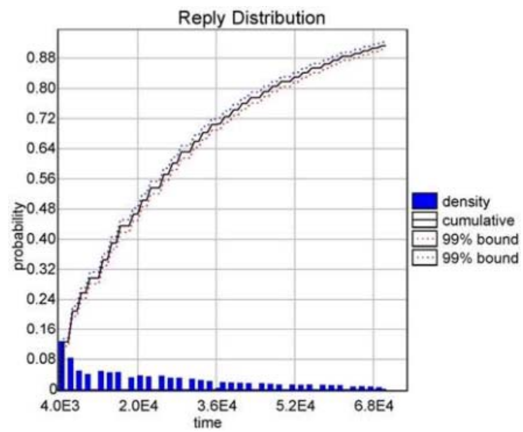


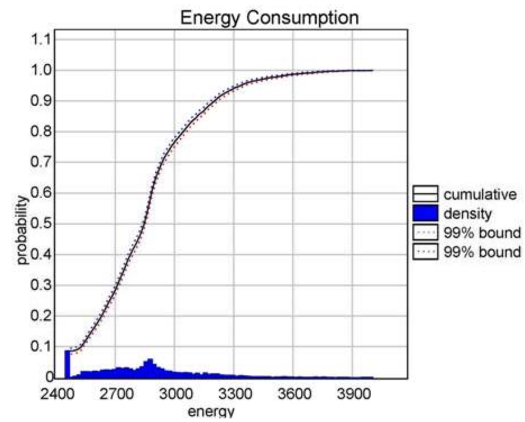
SMC in UPPAAL

- Constant Slope Timed Automata
  - Clocks may have different (integer) slope in different locations.
  - Branching edges with discrete probabilities (weights).
  - Beyond Priced TA, Energy TA. Equal LHA in (non-stochastic) expressive power.
  - Beyond DTMC, beyond CTMC (with multiple rewards)
- All features of UPPAAL supported
  - User defined functions and types
  - Expressions in guards, invariants, clock-rates, delay-rates (rationals), and weights.
- New GUI for plot-composing and exporting.

Case Studies

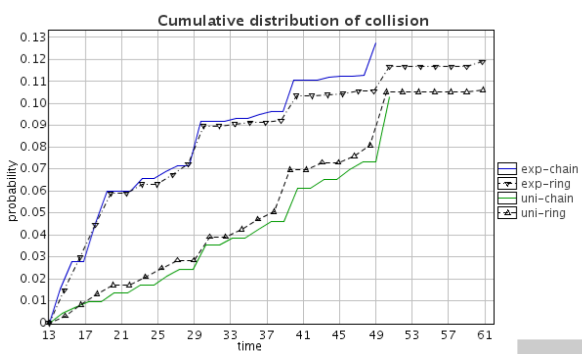


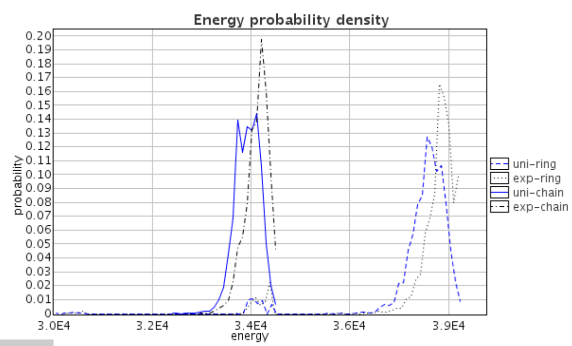




FIREWIRE

BLUETOOTH





DPA

LMAC

Benchmarking
Duration Probabilistic Automata

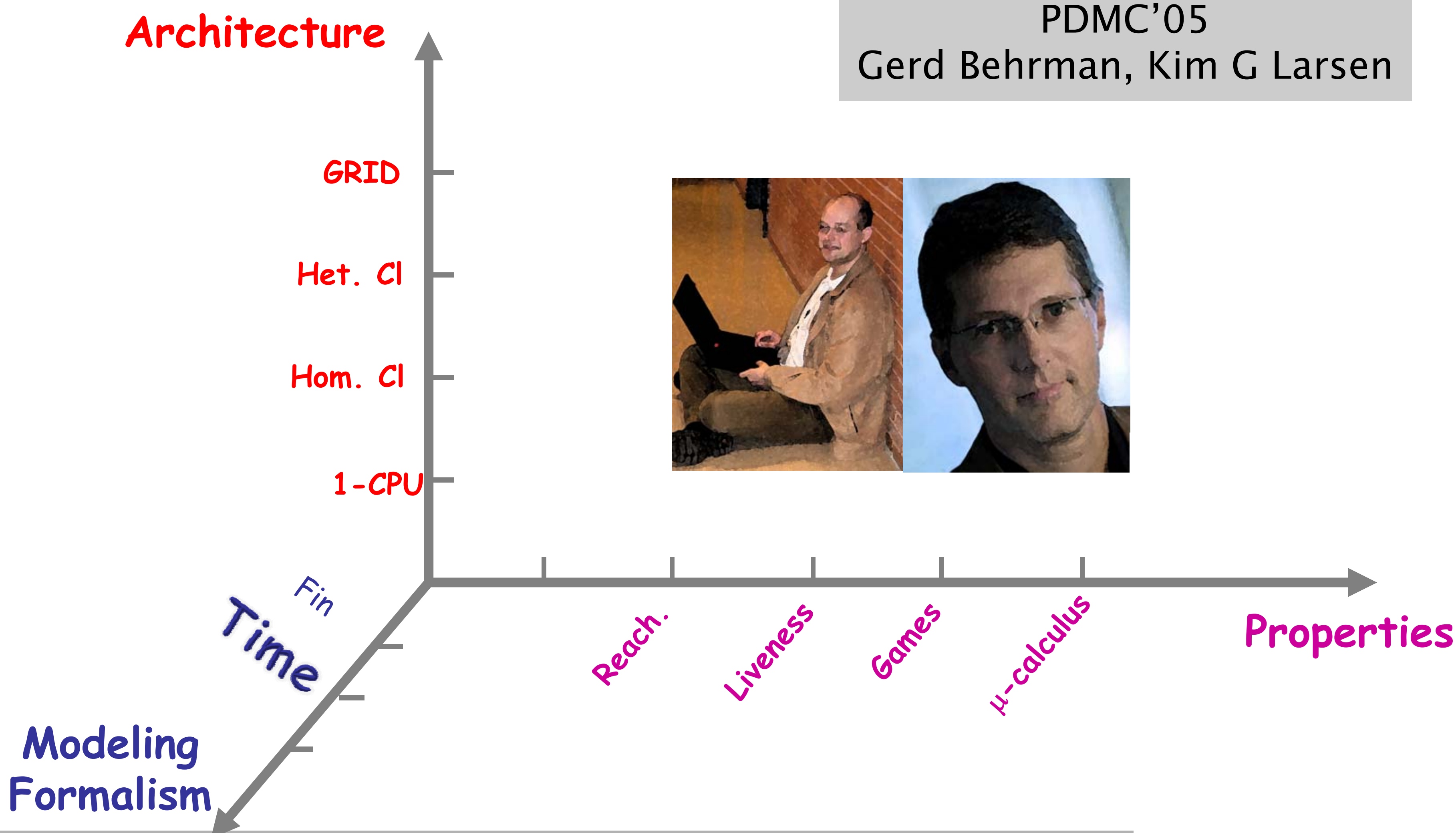
Param.			Estim.				Hyp. Testing			
n	k	m	PRISM	Up$_p$	Up$_d$	Up$_c$	PRISM	Up$_p$	Up$_d$	Up$_c$
4	4	3	2.7	0.3	0.2	0.2	2.0	0.1	0.1	0.1
6	6	3	7.7	0.6	0.5	0.4	3.9	0.2	0.2	0.3
8	8	3	26.5	1.2	0.9	0.7	16.4	0.5	0.4	0.3
20	40	20	>300				>300	35.5	26.2	20.7
30	40	20	>300				>300	61.2	41.8	33.2
40	40	20	>300				>300	92.2	56.9	59.5
40	20	20	>300				>300	41.1	31.2	26.5
40	30	20	>300				>300	68.8	46.7	46.1
40	55	40	>300				>300			219.5

Overview

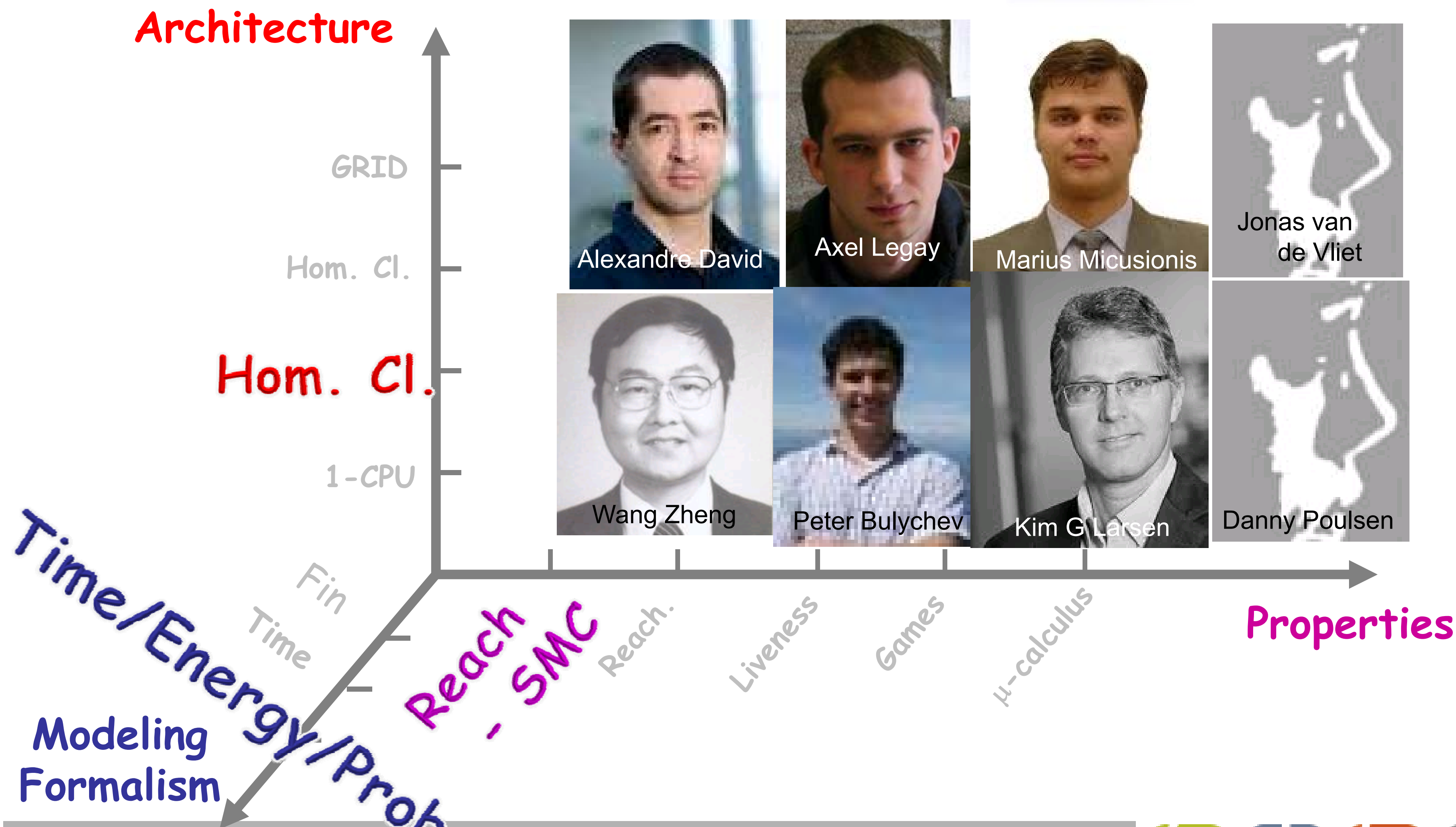
- Statistical Model Checking in UPPAAL
  - Estimation
  - Testing
- Distributed SMC for Parameterized Models
  - Parameter Sweeps
  - Optimization
  - Nash Equilibria
- Distributing Statistical Model Checking
  - Estimation
  - Testing
- Parameter Analysis of DSMC
- Conclusion

UPPAAL & PDMC'05

PDMC'05
Gerd Behrman, Kim G Larsen

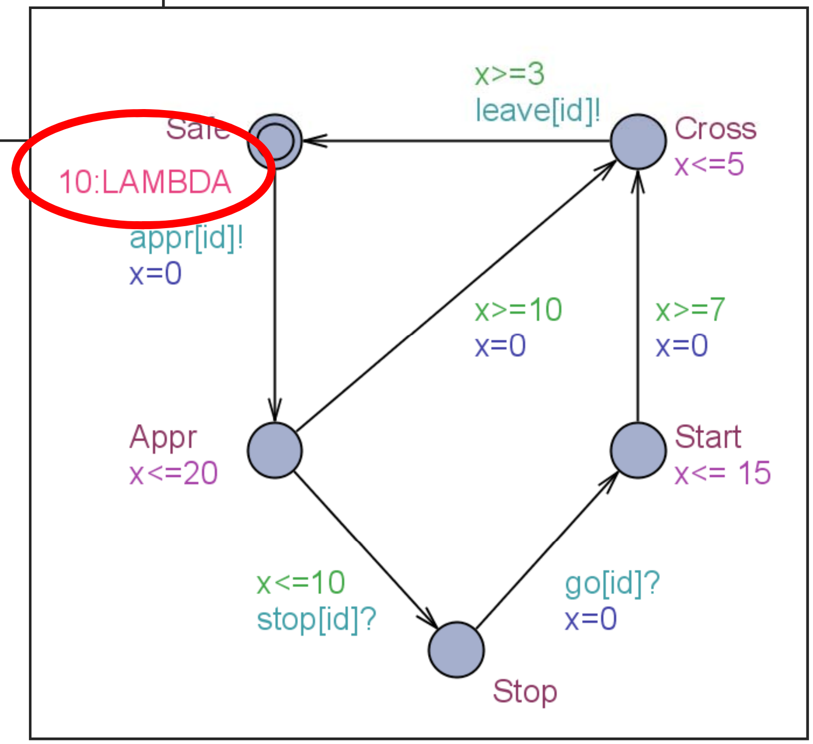


UPPAAL & PDMC'11



Parameterized Models in UPPAAL

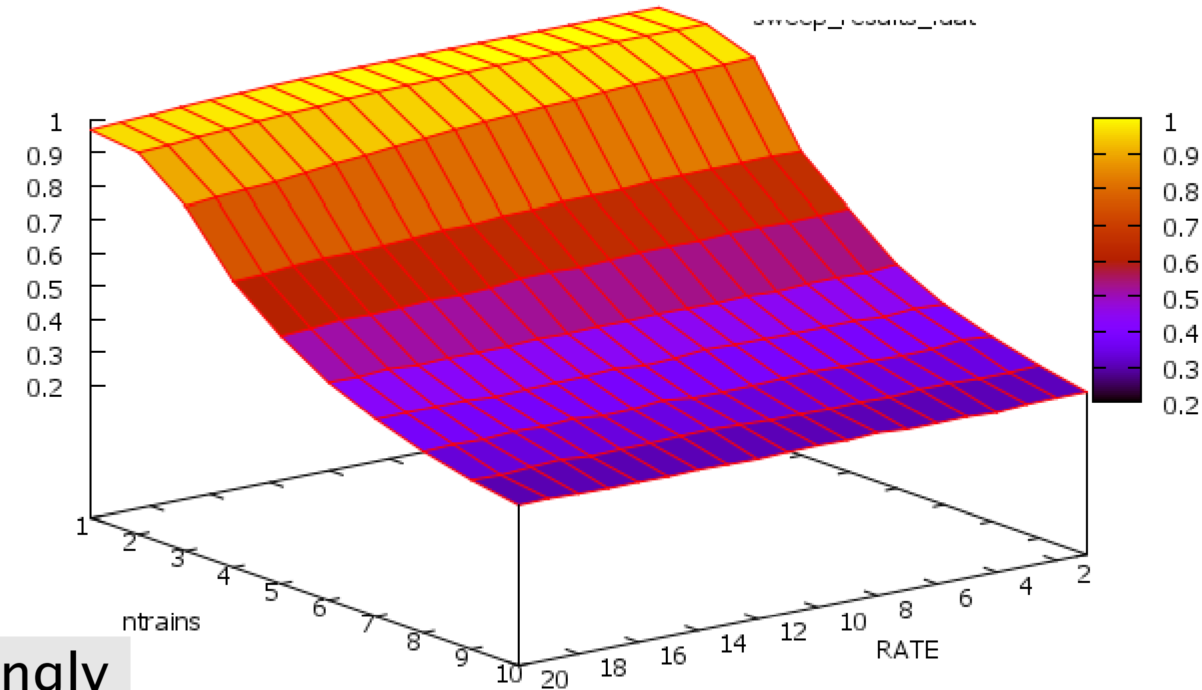
```
const int N = #range(1, 10, ntrains);          // # trains
const int LAMBDA = 5*#range(1, 20, RATE);
typedef int[0,N-1] id_t;

broadcast chan         appr[N], stop[N], leave[N];
urgent broadcast chan go[N];
clock time;
```



Extended Syntax
constants declared with a range
are treated as parameter

Parameterized Analysis of Trains

Pr[time<=100](<>Train(0).Cross)



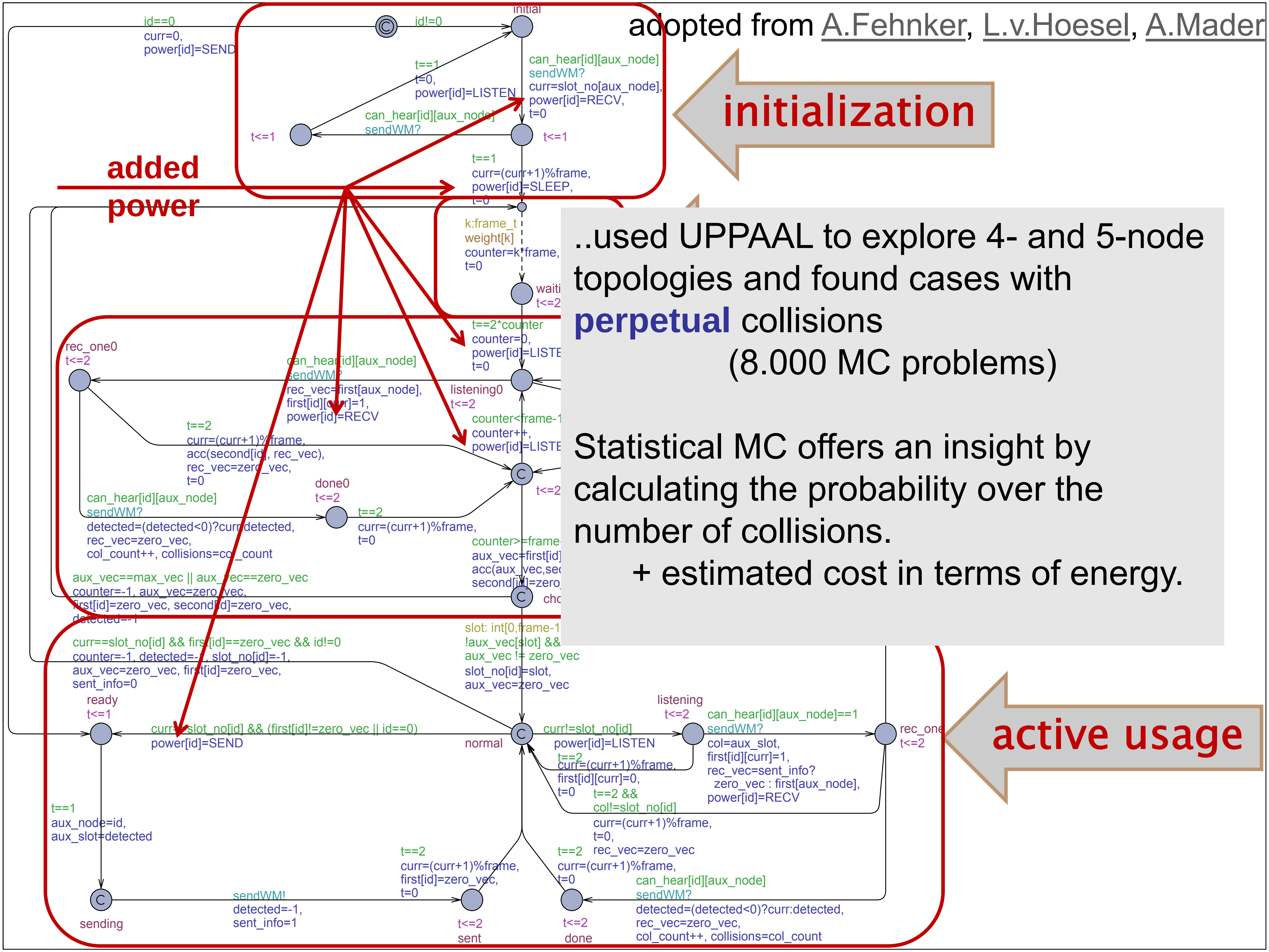
"Embarrassingly Parallelizable"

Lightweight Media Access Control

- Problem domain:
  - communication scheduling
- Targeted for:
  - self-configuring networks,
  - collision avoidance,
  - low power consumption
- Application domain:
  - wireless sensor networks

- Initialization (listen until a neighbor is heard)
- Waiting (delay a random amount of time frames)
- Discovery (wait for entire frame and note used slots)
- Active
  - choose free slot,
  - use it to transmit, including info about detected collisions
  - listen on other slots
  - fallback to Discovery if collision is detected
- Only neighbors can detect collision and tell the user-node that its slot is used by others

adopted from A.Fehnker, L.v.Hoesel, A.Mader

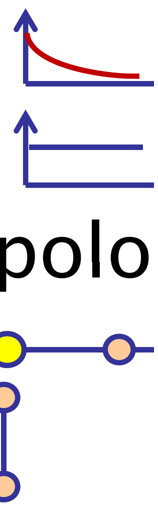


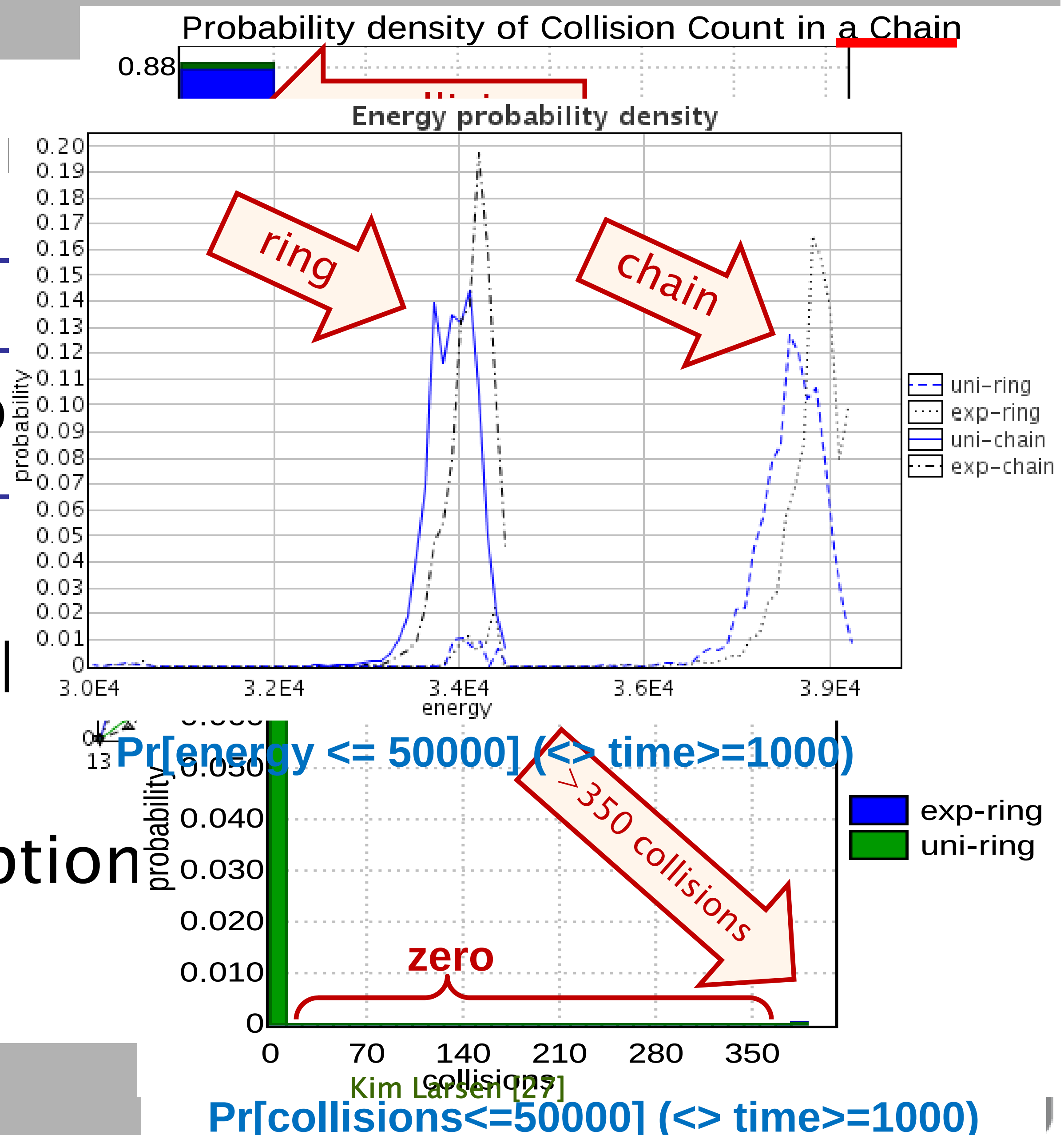
..used UPPAAL to explore 4- and 5-node topologies and found cases with **perpetual** collisions (8.000 MC problems)

Statistical MC offers an insight by calculating the probability over the number of collisions.
+ estimated cost in terms of energy.

SMC of LMAC with 4 Nodes

- Wait distributio
  - geometric
  - uniform
- Network topolo
  - chain
  - ring
- Collision probal
- Collision count
- Power consumption

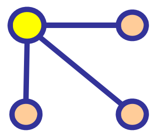
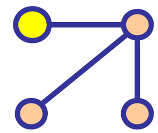
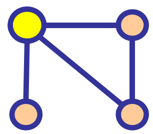
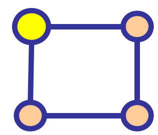
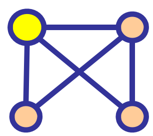
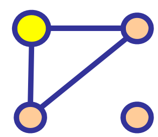
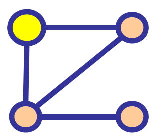
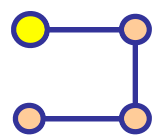




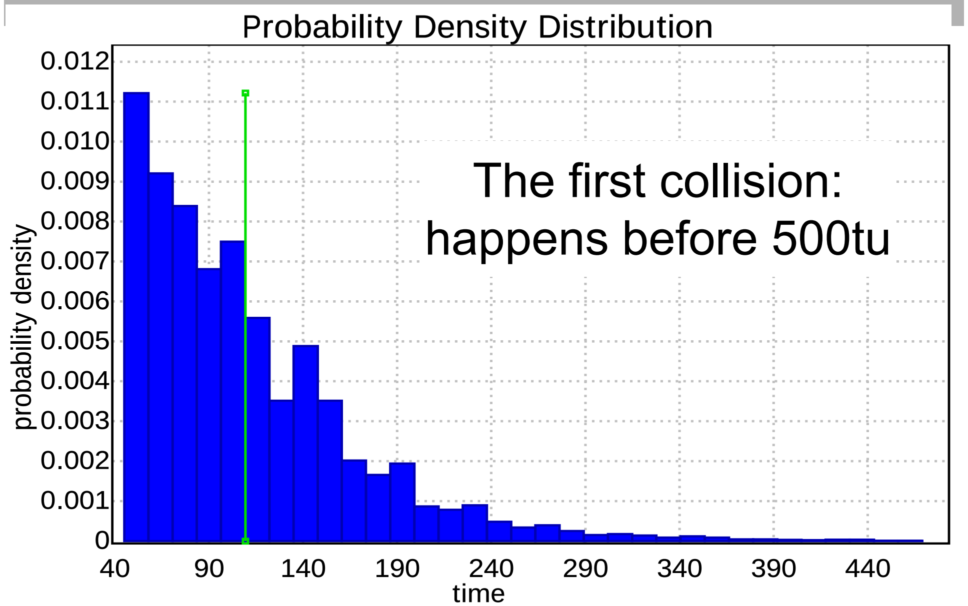
LMAC with Parameterized Topology
Distributed SMC

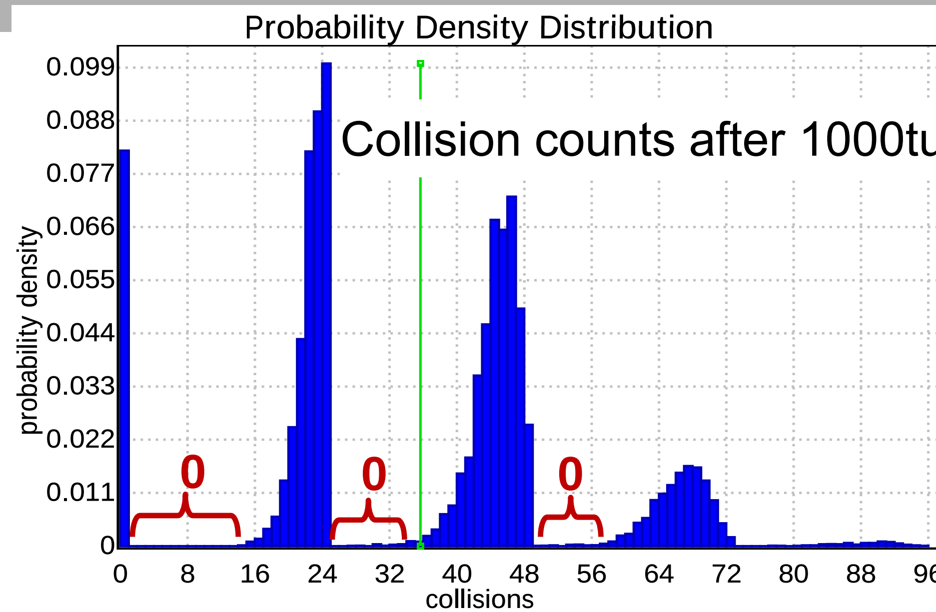
Collision probability in a 4 node network: sweep over all topologies.
32 core cluster: - 8xIntel Core2 2.66GHz CPU

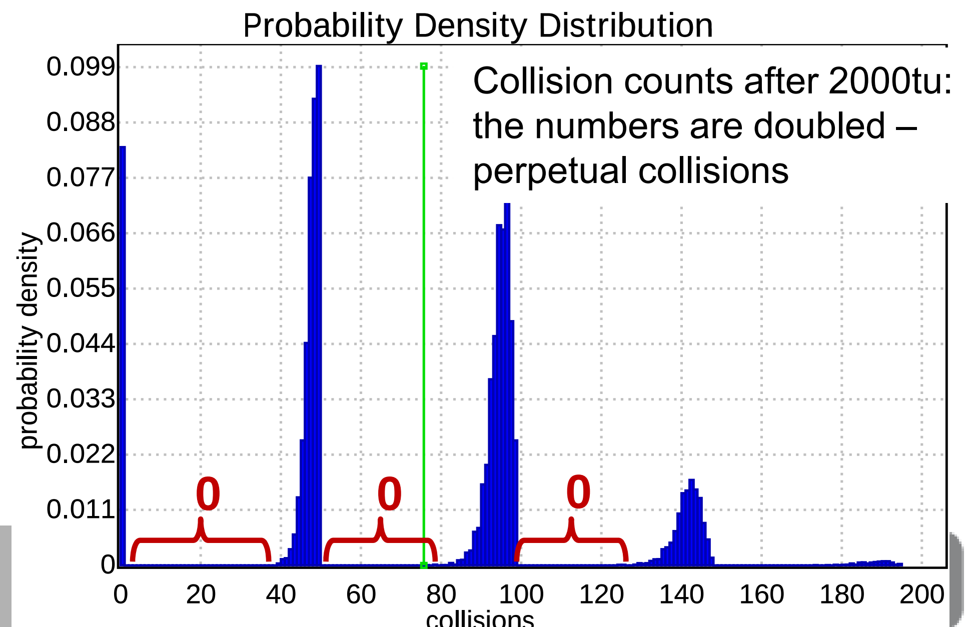
Pr[time<=200] (<> col_count>0)

	topology	collision probability		topology	collision probability
(star)		**[0.36; 0.39]**			**[0.08; 0.19]**
		**[0.29; 0.36]**	(ring)		**[0.11; 0.13]**
		**[0.26; 0.30]**			**[0.08; 0.15]**
		**[0.19; 0.21]**	(chain)		**[0.049; 0.050]**

10-Node Star

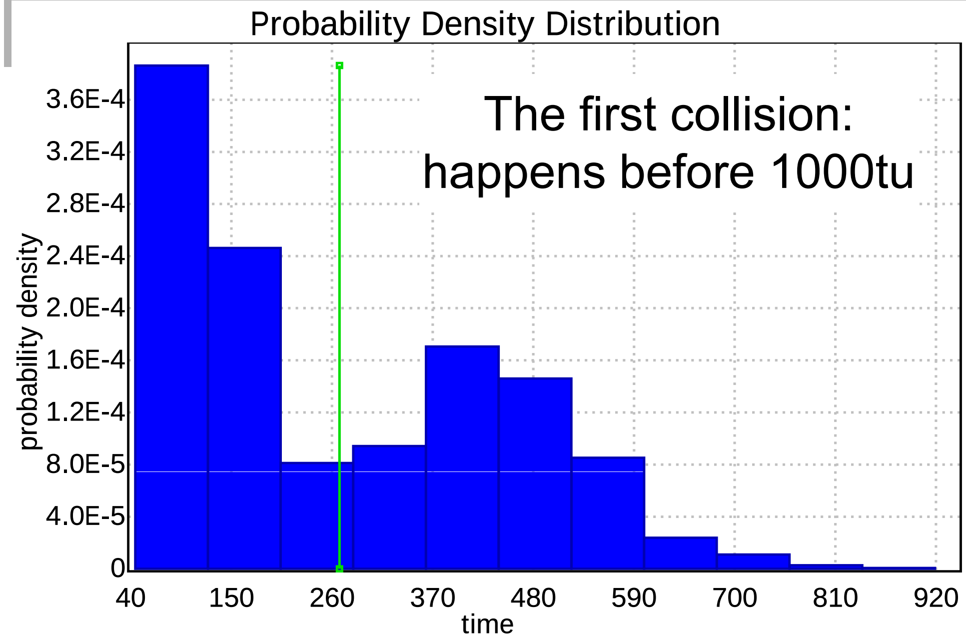


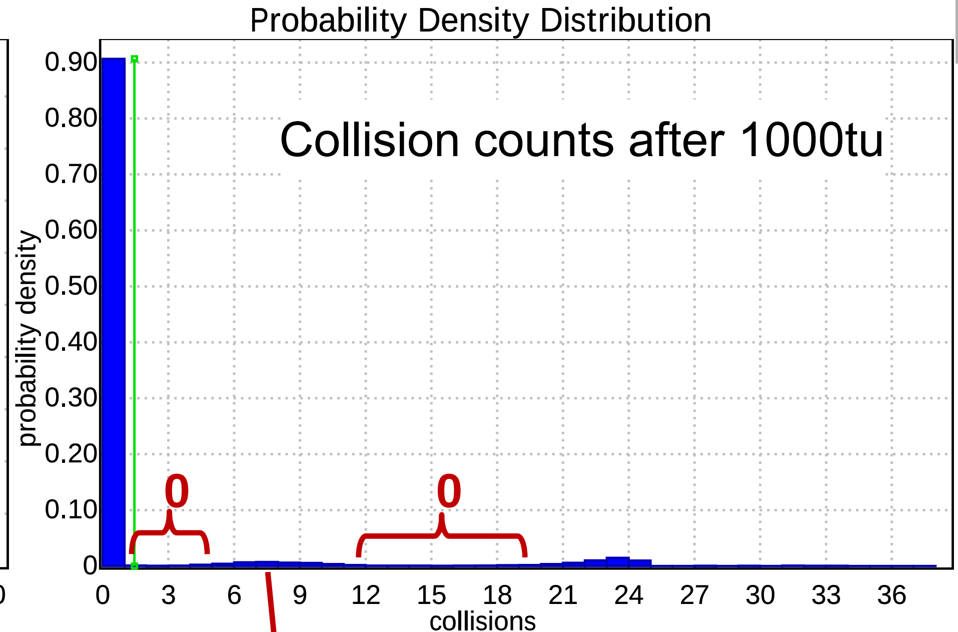


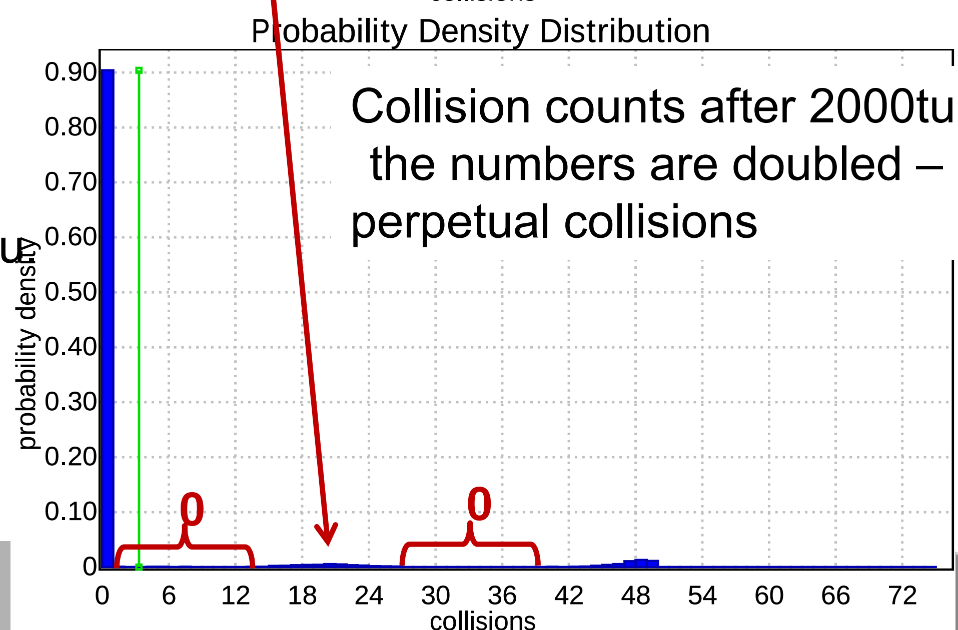


- The first collisions happen before **500**tu.
- It is unlikely (**8.2%**) that there will be **0** collisions.
- And if they happen, they are perpetual.

10-Node Ring

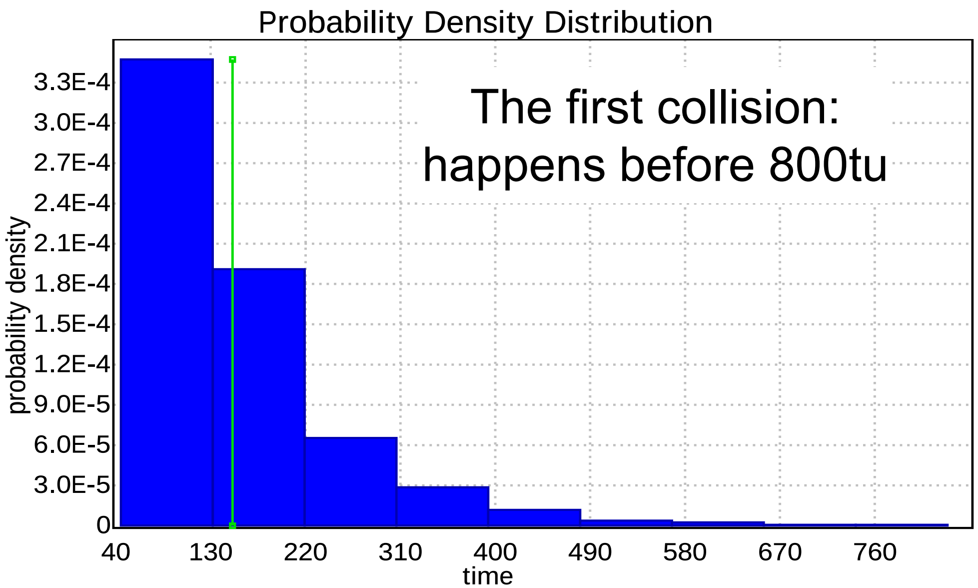


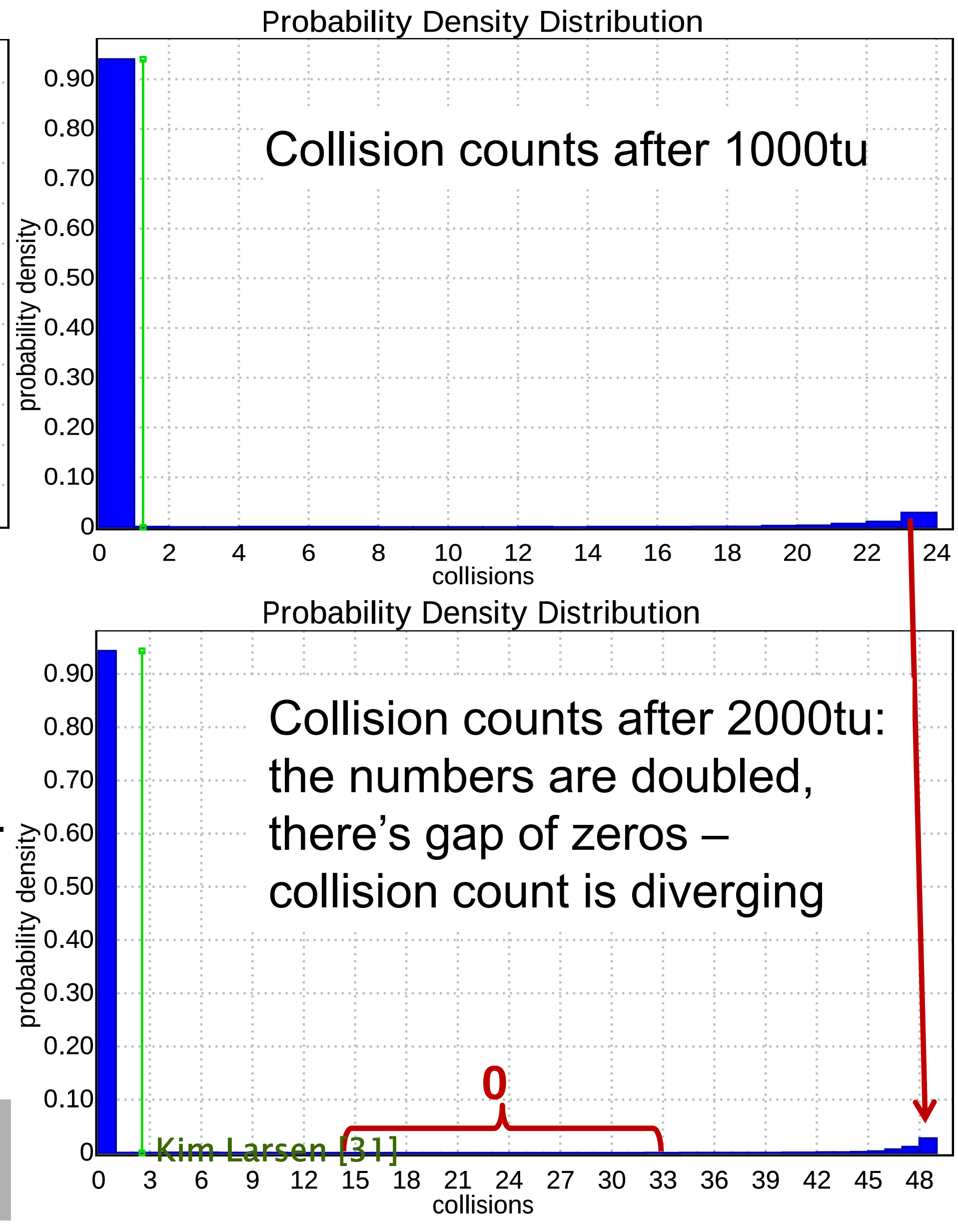




- The first collisions can be as late as **920**tu
- It is very likely (**>90%**) that there will be **0** collisions.
- But if they happen, they are perpetual.

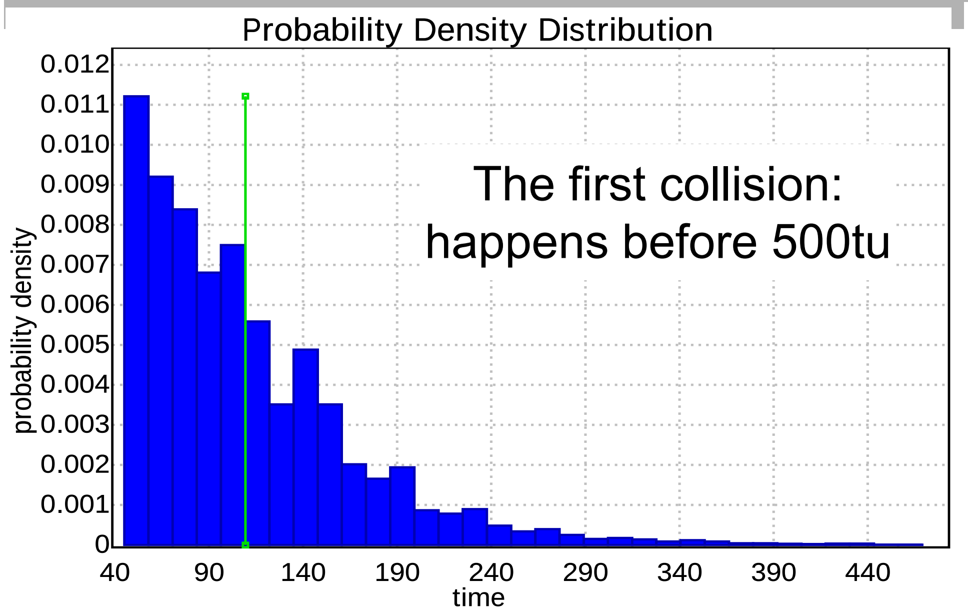
10-Node Chain

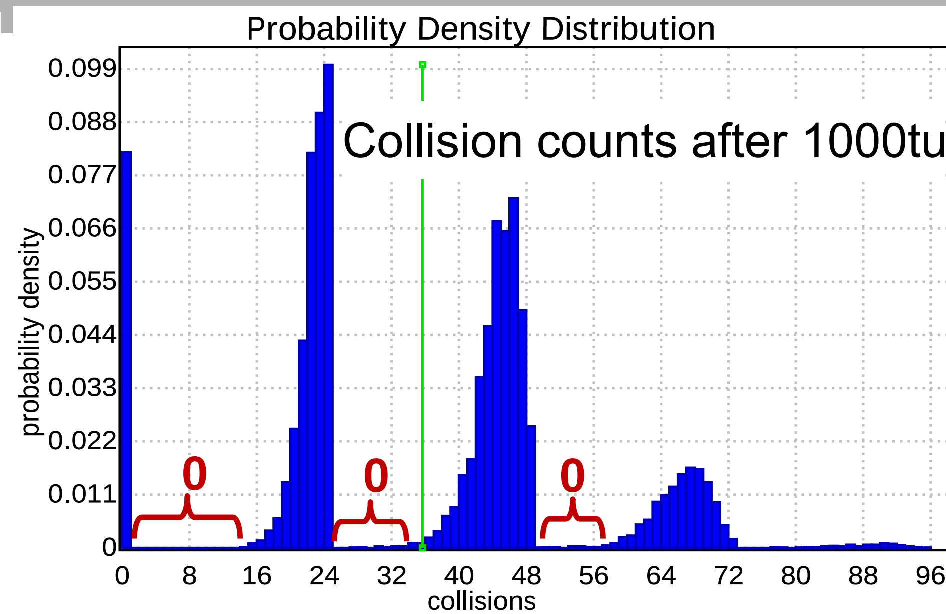


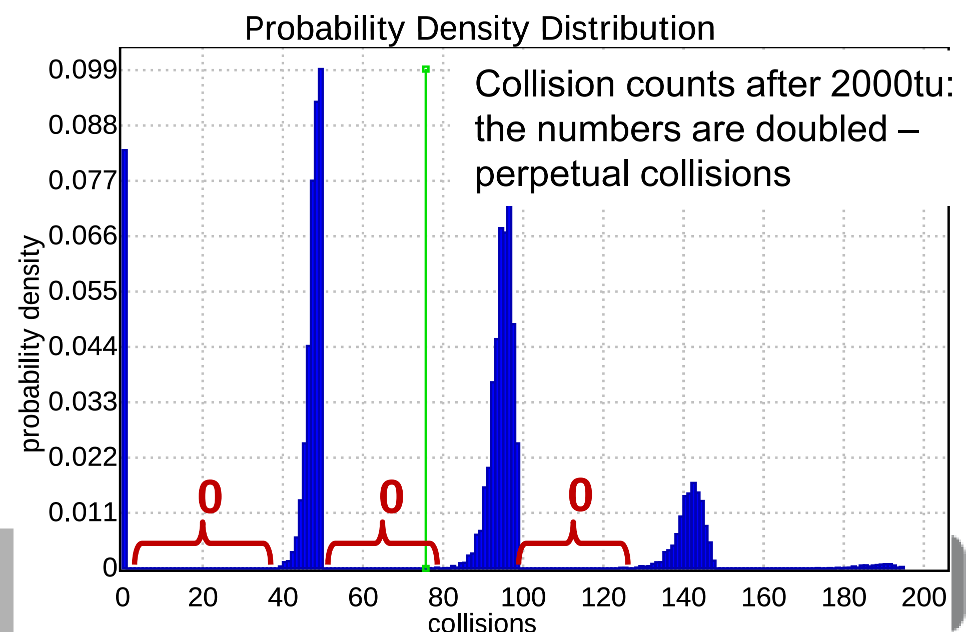


- The first collisions can be as late as **800**tu.
- It is very likely (**>94%**) that there will be **0** collisions.
- But if they happen, some are perpetual.

10-Node Star



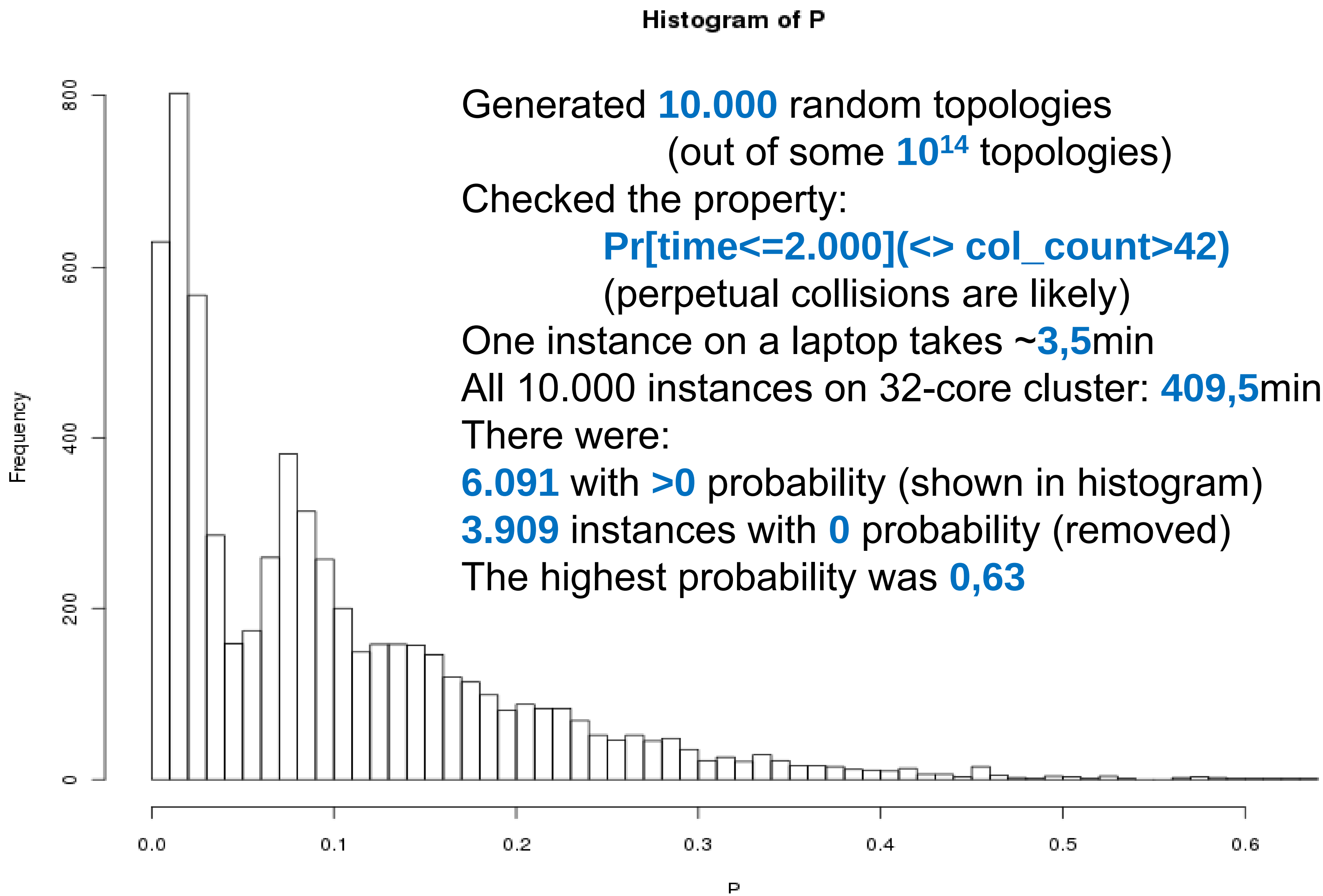




- The first collisions happen before **500**tu.
- It is unlikely (**8.2%**) that there will be **0** collisions.
- And if they happen, they are perpetual.

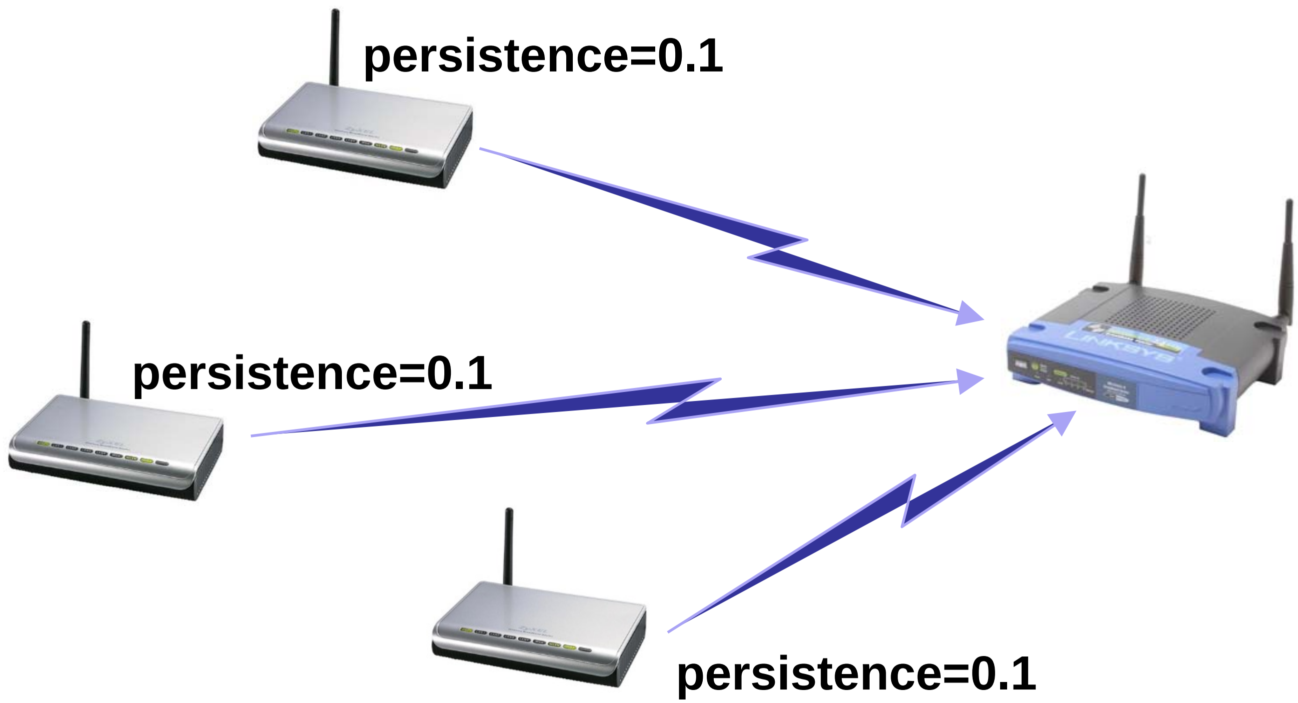
10-Node Random Topologies
Distributed SMC

Generated **10.000** random topologies
(out of some **10^{14}** topologies)
Checked the property:
Pr[time<=2.000](<> col_count>42)
(perpetual collisions are likely)
One instance on a laptop takes ~**3,5**min
All 10.000 instances on 32-core cluster: **409,5**min
There were:
6.091 with **>0** probability (shown in histogram)
3.909 instances with **0** probability (removed)
The highest probability was **0,63**



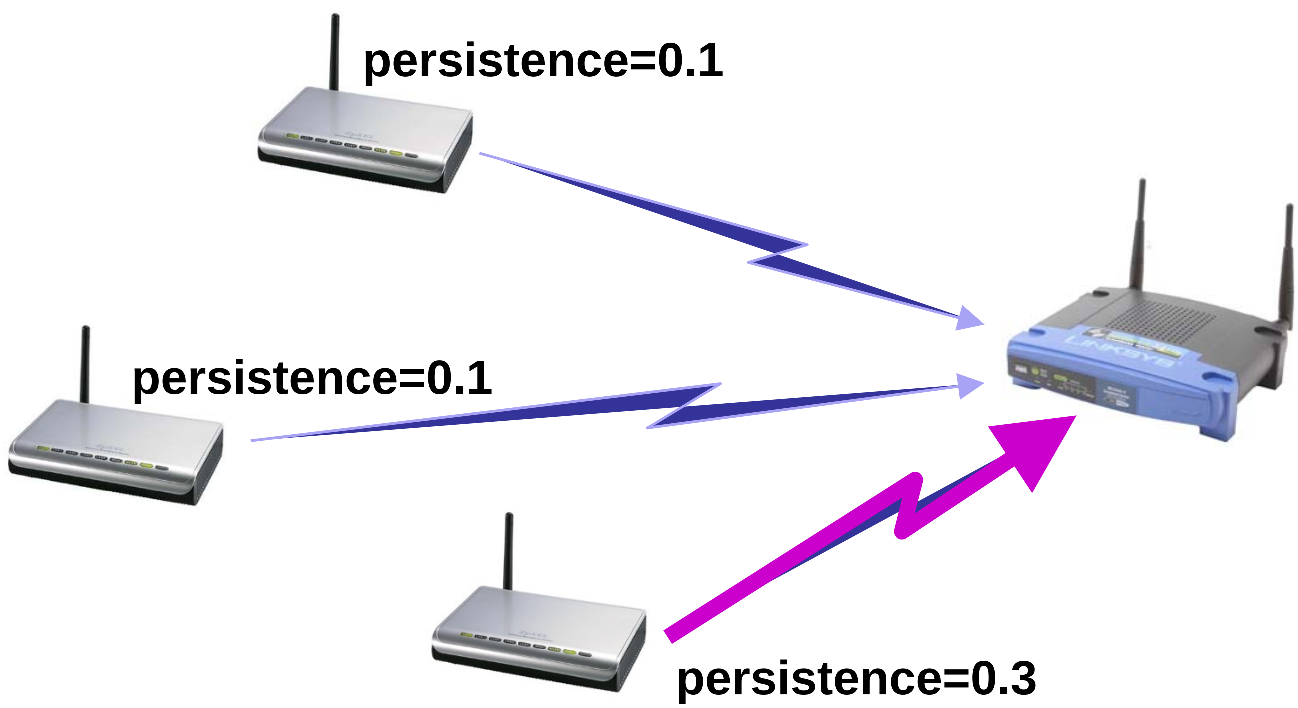
Nash Eq in Wireless Ad Hoc Networks

Consider a wireless network, where there are nodes that can independently adapt their parameters to achieve better performance



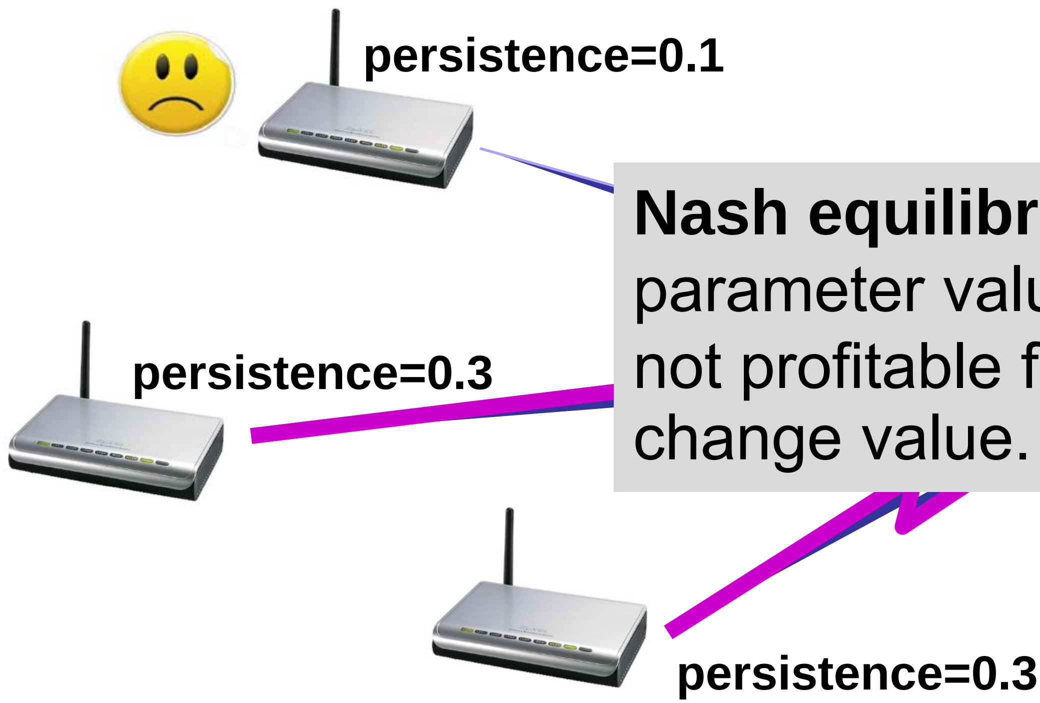
Nash Eq in Wireless Ad Hoc Networks

Consider a wireless network, where there are nodes that can independently adapt their parameters to achieve better performance



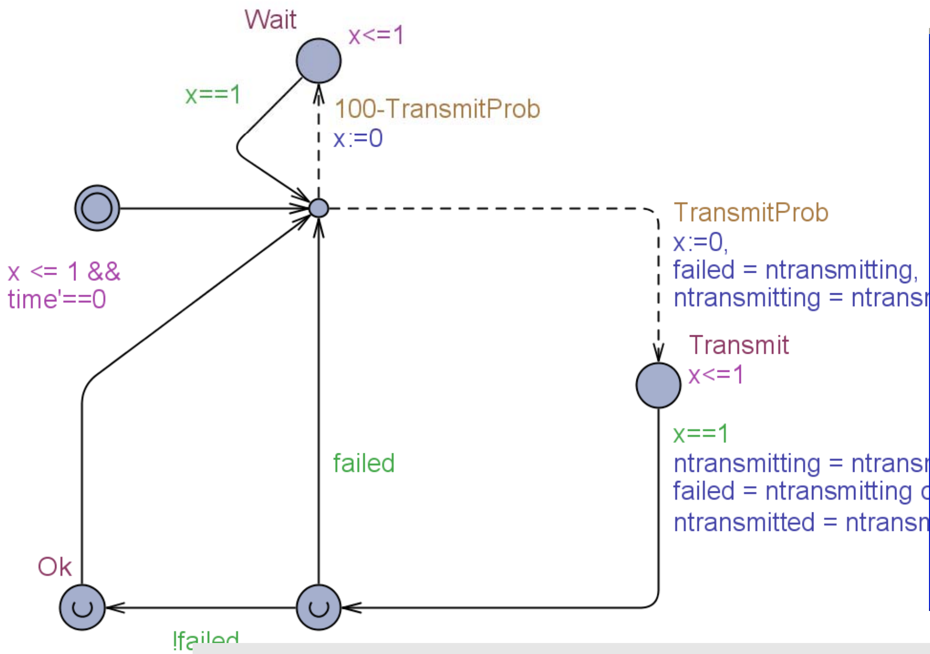
Nash Eq in Wireless Ad Hoc Networks

Consider a wireless network, where there are nodes that can independently adapt their parameters to achieve better performance

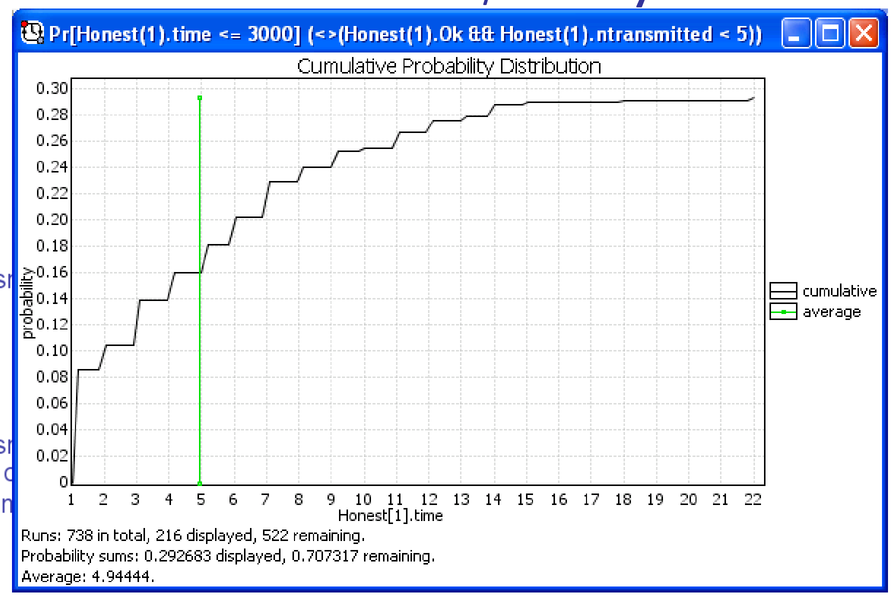


Nash equilibrium (NE): e.g. parameter values such that it's not profitable for a node to change value.

Aloha CSMA/CD protocol



TransmitProb=0.5 / Utility=0.29



Pr[Node.time <= 3000](<>(Node.Ok && Node.ntransmitted <= 5))

- Simple random access protocol (based on p-persistent ALOHA)
  - several nodes sharing the same wireless medium
  - each node has always data to send, and it sends data after a random delay
  - delay geometrically distributed with parameter p=TransmitProb

Distributed Algoritm for Computing Nash Equilibrium

<u>Input</u>: $S=\{s_i\}$ – finite set of strategies, $U(s_i, s_k)$ – utility function

<u>Goal</u>: find s_i s.t. $\forall s_k\ U(s_i, s_i) \geq U(s_i, s_k)$, where $s_i, s_k \in S$

<u>Algorithm</u>:

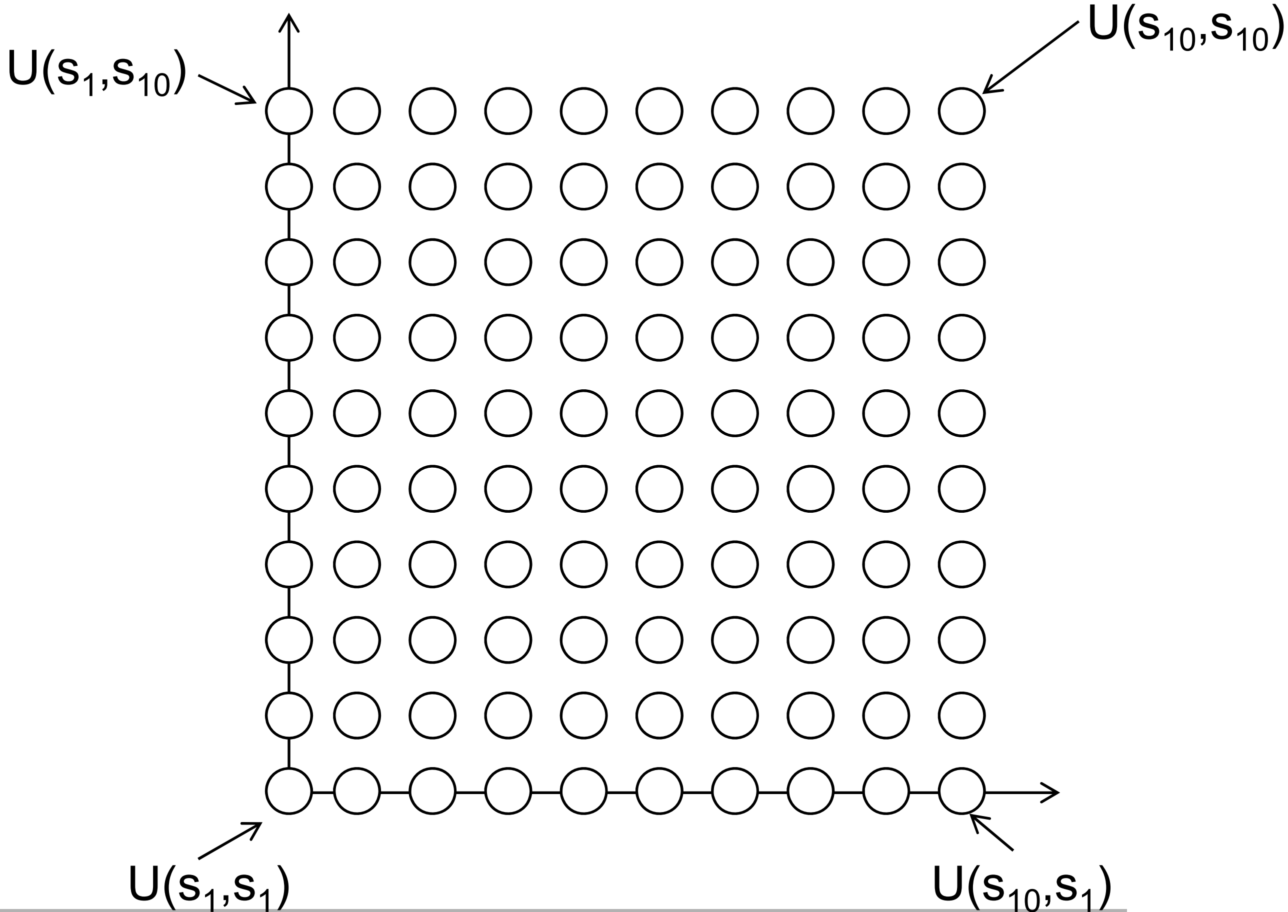
for every $s_i \in S$ **compute** $U(s_i,s_i)$
candidates := S
while len(*candidates*)>1:
    pick some unexplored pair $(s_i,s_k) \in candidates \times S$
    compute $U(s_i, s_k)$
    if $U(s_i,s_k) > U(s_i,s_i)$:
        remove s_i from candidates
    if $\forall s_k\ U(s_i, s_k)$ is already computed:
        return s_i

<u>**We can apply statistics to *prove* that (s_i,s_i) satisfies Nash equilibrium**</u>

Distributed algorithm for computing Nash equilibrium

Input: $S=\{s_1, s_2, \ldots, s_{10}\}$ – finite set of strategies, $U(s_i, s_j)$ – utility function

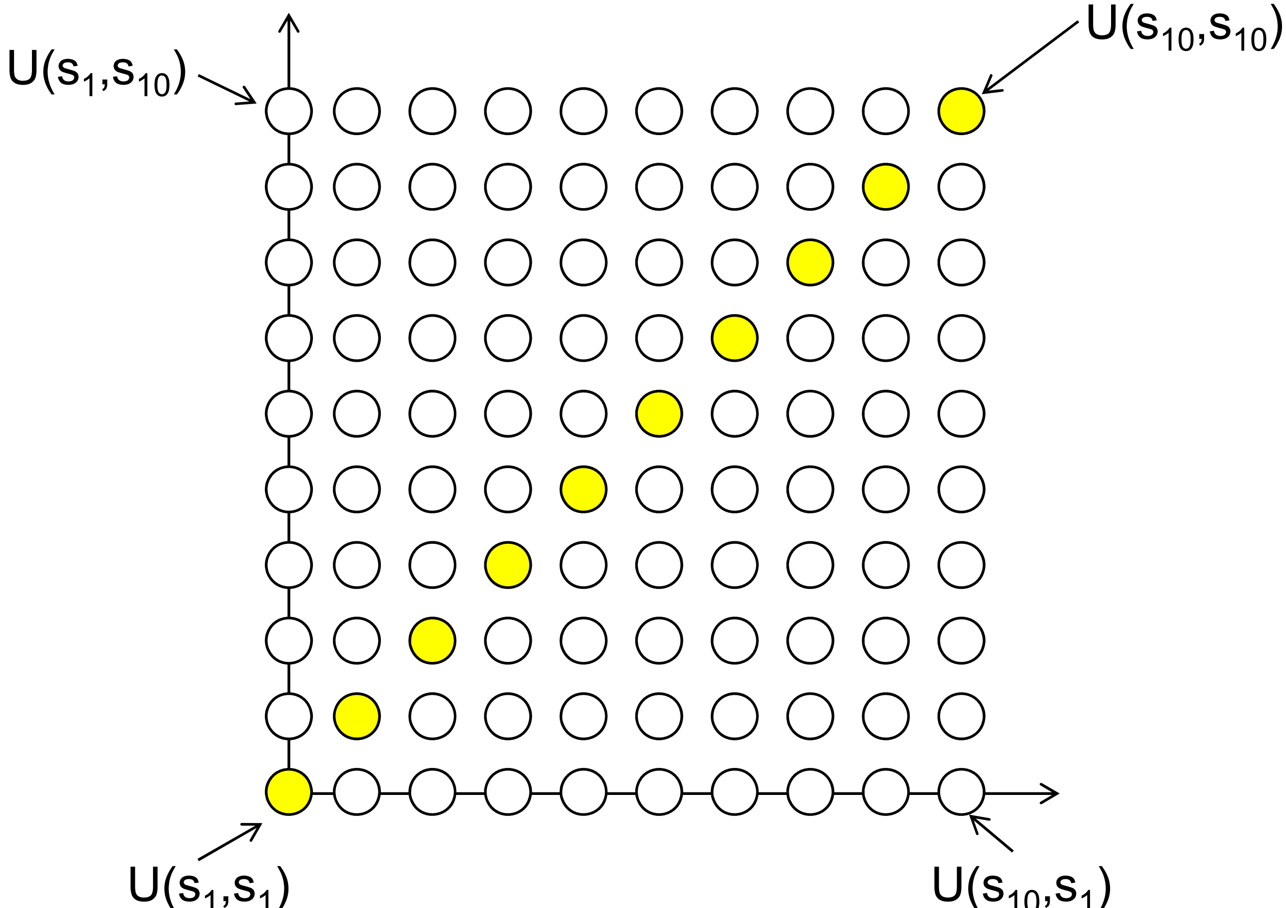
Goal: find s_i s.t. $\forall s_k\ U(s_i, s_i) \geq U(s_i, s_k)$, where $s_i, s_k \in S$



Distributed algorithm for computing Nash equilibrium

Input: $S=\{s_1, s_2, \ldots, s_{10}\}$ – finite set of strategies, $U(s_i, s_j)$ – utility function

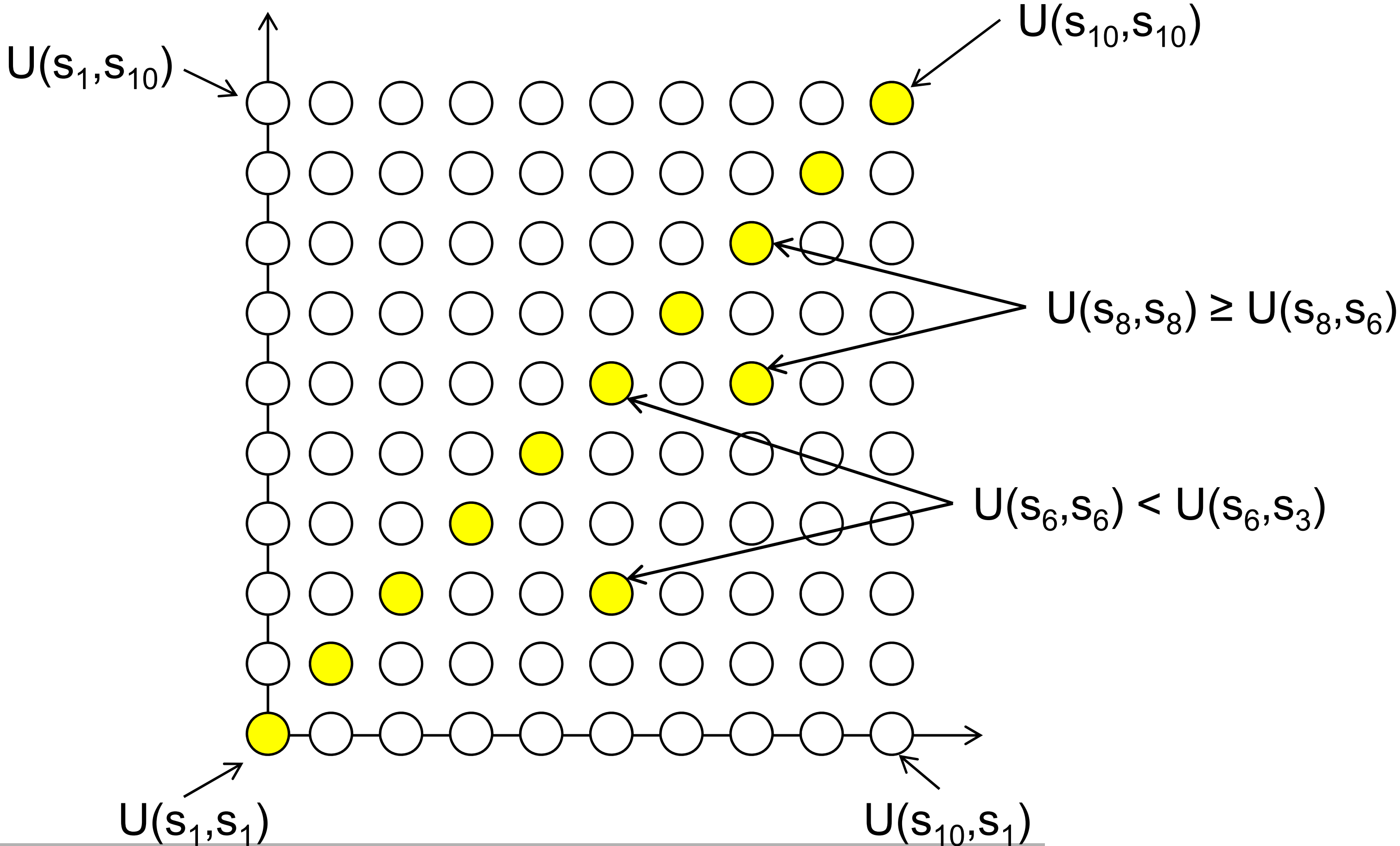
Goal: find s_i s.t. $\forall s_k$ $U(s_i, s_i) \geq U(s_i, s_k)$, where $s_i, s_k \in S$



Distributed algorithm for computing Nash equilibrium

Input: $S=\{s_1, s_2, \ldots, s_{10}\}$ – finite set of strategies, $U(s_i, s_j)$ – utility function

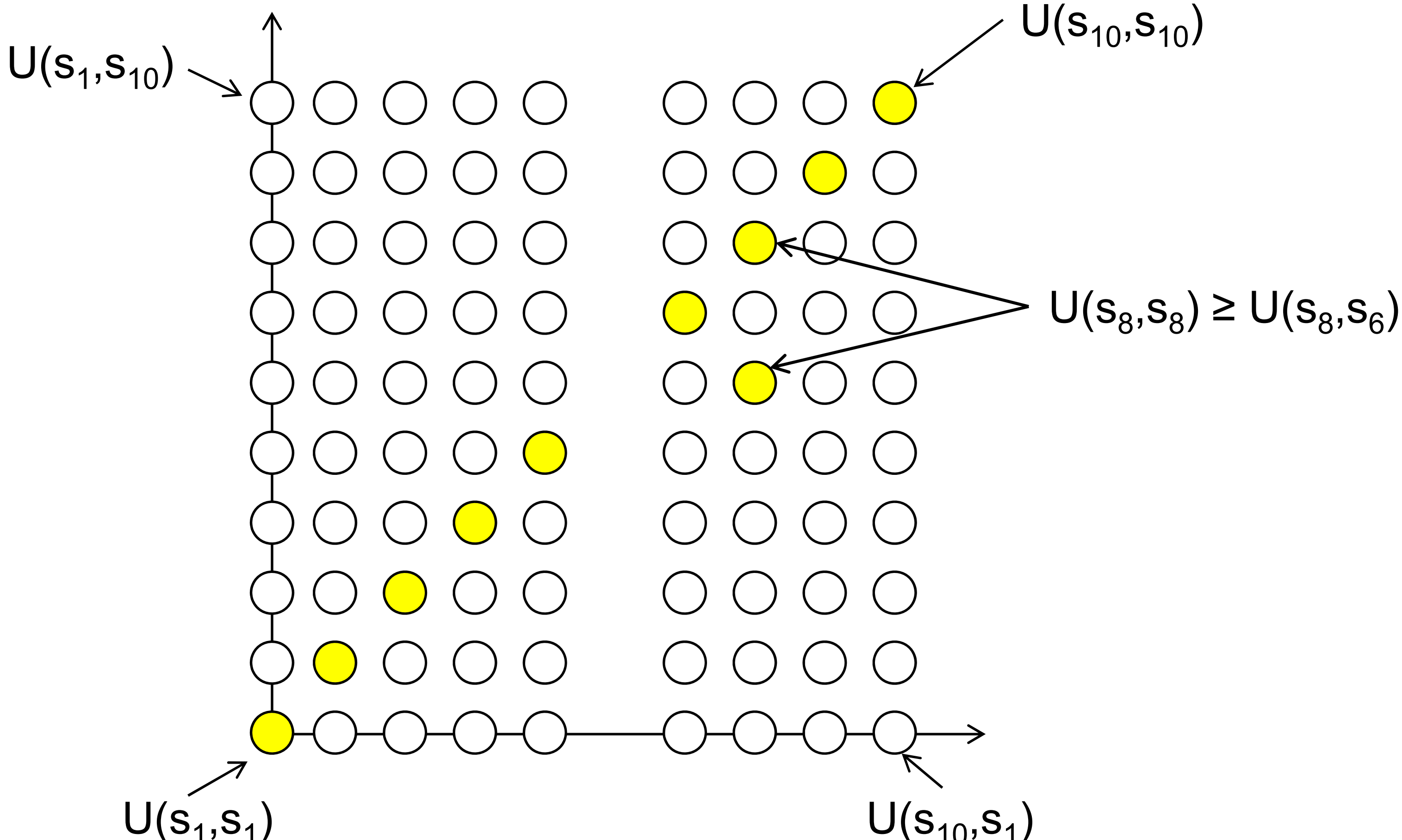
Goal: find s_i s.t. $\forall s_k\ U(s_i, s_i) \geq U(s_i, s_k)$, where $s_i, s_k \in S$



Distributed algorithm for computing Nash equilibrium

Input: $S=\{s_1, s_2, \ldots, s_{10}\}$ – finite set of strategies, $U(s_i, s_j)$ – utility function

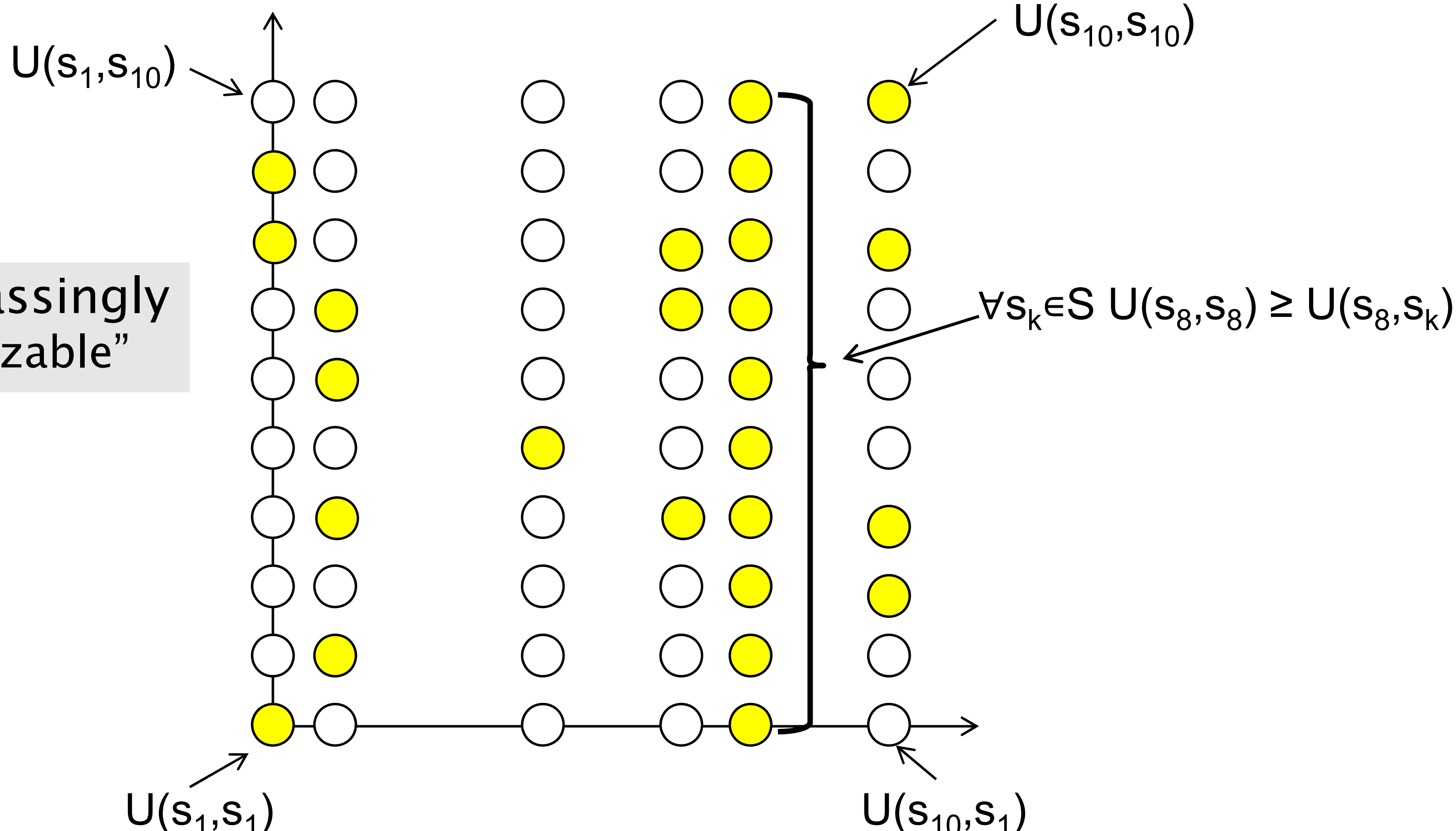
Goal: find s_i s.t. $\forall s_k\ U(s_i, s_i) \geq U(s_i, s_k)$, where $s_i, s_k \in S$



Distributed algorithm for computing Nash equilibrium

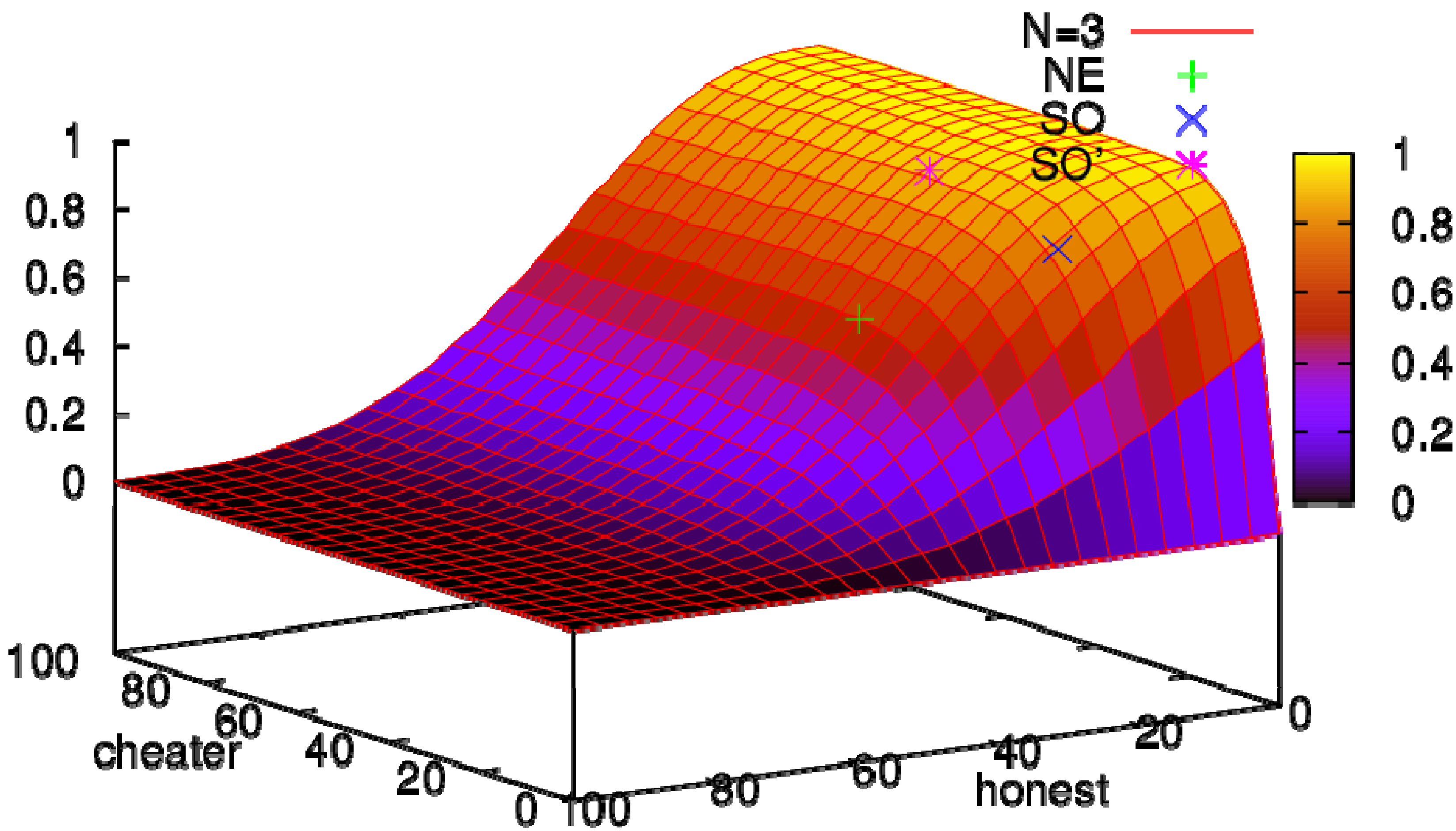
Input: $S=\{s_1, s_2, \ldots, s_{10}\}$ – finite set of strategies, $U(s_i, s_j)$ – utility function

Goal: find s_i s.t. $\forall s_k\ U(s_i, s_i) \geq U(s_i, s_k)$, where $s_i, s_k \in S$



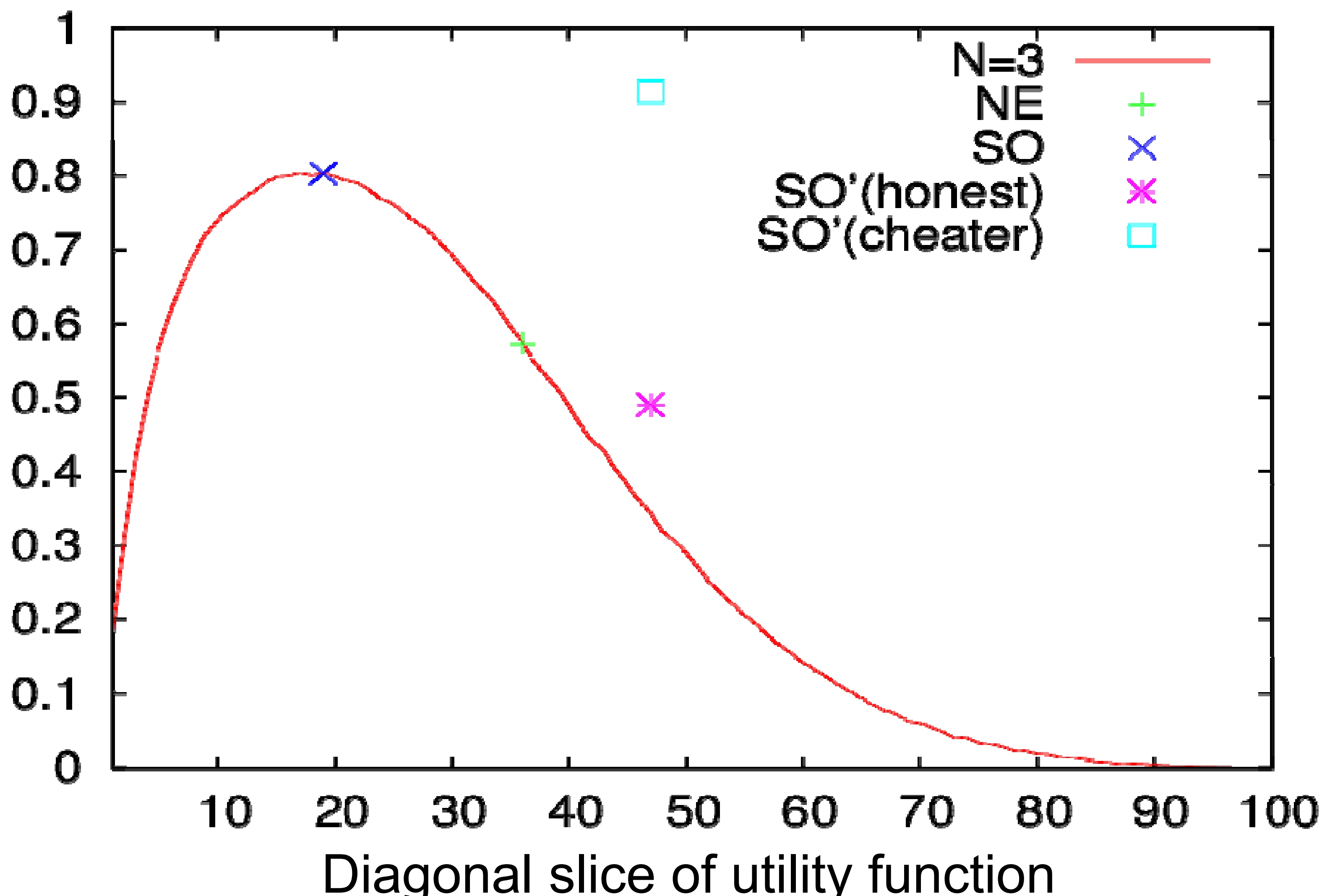
"Embarrassingly Parallelizable"

Results (3 nodes)



Value of utility function for the cheater node

Results (3 nodes)



Results

	N=2	N=3	N=4	N=5	N=6	N=7
Nash Eq (TrnPr)	0.32	0.36	0.36	0.35	0.32	0.32
$U(s_{NE},s_{NE})$	0.91	0.57	0.29	0.15	0.10	0.05
Opt (TrnPr)	0.25	0.19	0.14	0.11	0.09	0.07
$U(s_{opt}, s_{opt})$	0.93	0.80	0.68	0.58	0.50	0.44

Symmetric Nash Equilibrium and Optimal strategies for different number of network nodes

#cores	4	8	12	16	20	24	28	32
Time	38m	19m	13m	9m46s	7m52s	7m04s	6m03s	5m

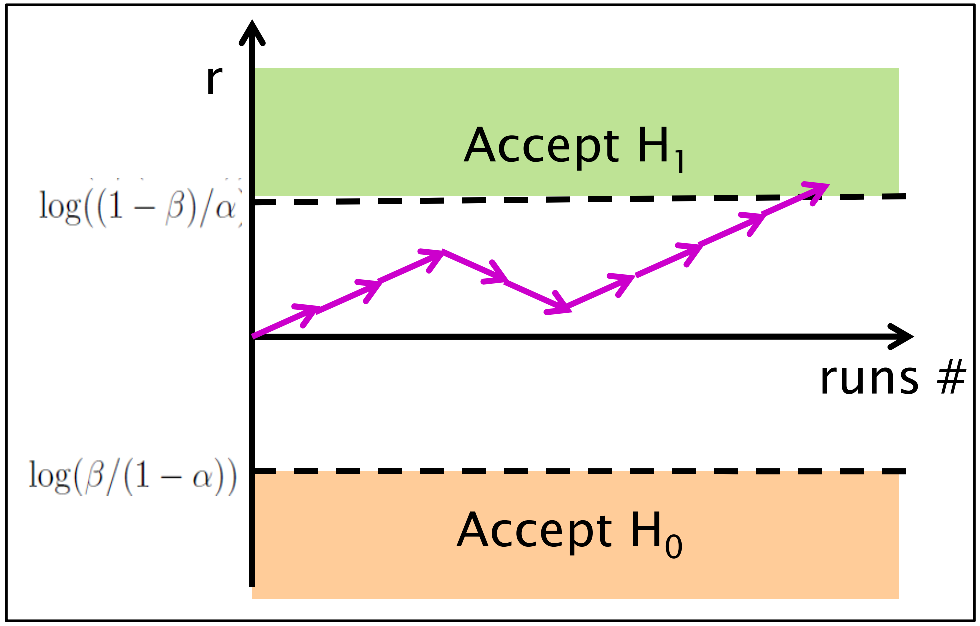
Time required to find Nash Equilibrium for N=3
100x100 parameter values
(8xIntel Core2 2.66GHz CPU)

Overview

- Statistical Model Checking in UPPAAL
  - Estimation
  - Testing
- Distributed SMC for Parameterized Models
  - Parameter Sweeps
  - Optimization
  - Nash Equilibria
- Distributing Statistical Model Checking
  - Estimation
  - Testing
- Parameter Analysis of DSMC
- Conclusion

Bias Problem

- Suppose that generating accepting runs is fast and non-accepting runs is slow.
- 1-node exploration:
  - Generation is sequential, only the outcomes count.
- N-node exploration:
  - There may be an unusual peak of accepting runs generated more quickly by some nodes that will arrive long before the non-accepting runs have a chance to be counted!
  - The decision will be biased toward accepting runs.

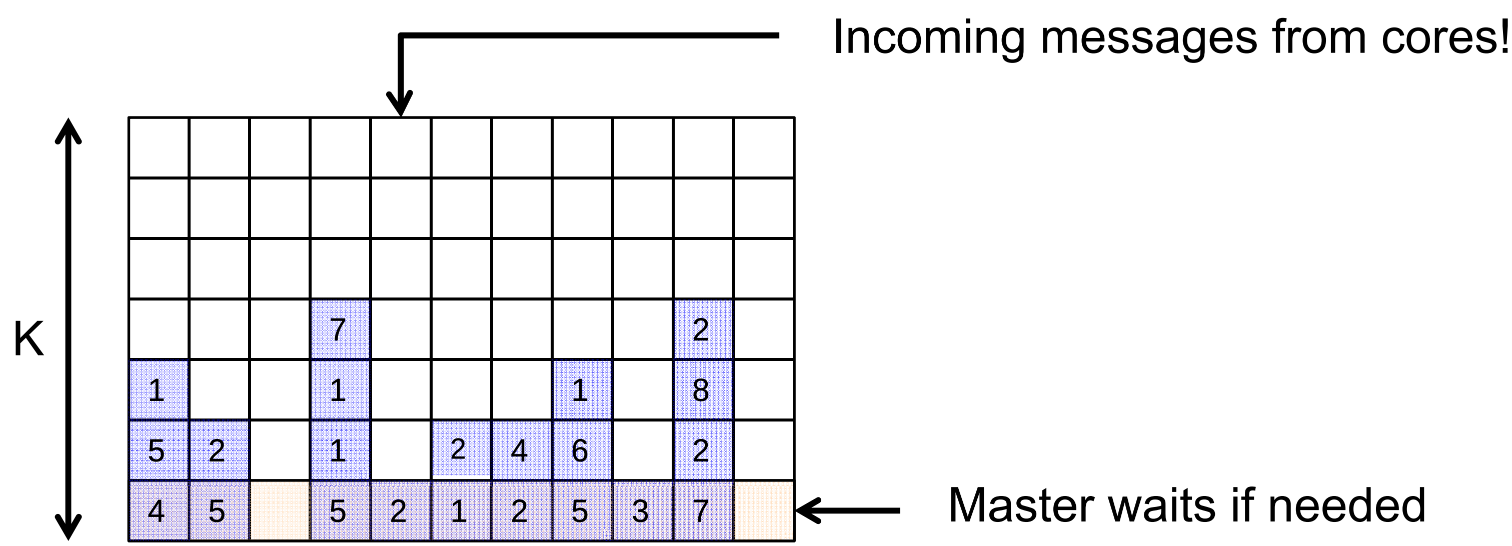


Solving Bias [Younes'05]

Queue the results at a master, use Round-Robin between nodes to accept the results.

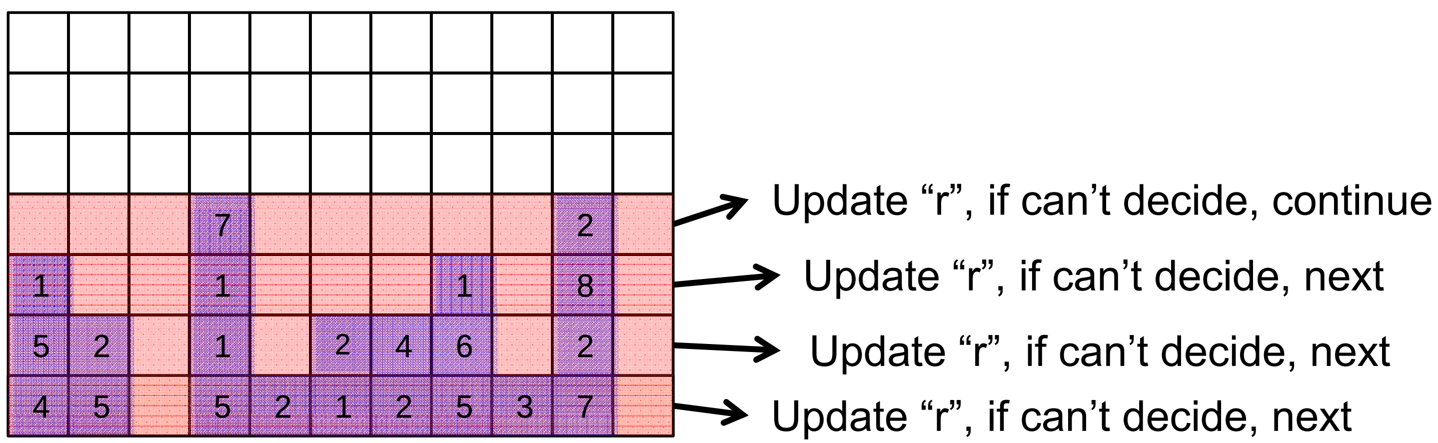
Our Implementation

- Use a batch of **B** (e.g 10) runs, transmit one count per batch.
- Use asynchronous communication (MPI)
- Queue results at the master and wait only when the buffer (size=K) is full.



Our Implementation

- Senders have a buffer of (K) asynchronously sent messages and blocks only when the buffer is full.
- The master periodically add results in the buffer.



Experiment on Multi-Core

- Machine: i7 4*cores HT, 4GHz.
  - Hyperthreading is an interesting twist:
    - threads share execution units,
    - have unpredictable running times
      (may run on same physical core if $<$ 8 threads).
- Model: Train Gate with 20 trains.
- Configuration – B=40, K=64
- Property:
  "mutual exclusion on the bridge within time $\leq$ 1000"

  H0: accept if Pr $\geq$ 0.9999
  H1: accept if Pr $\leq$ 0.9997
  α=0.001, β=0.001.

Performance

- Compared to base non-MPI version.
- Min, average, max *.
- 4.99 max speedup on a quad-core.

	Speedup			Efficiency		
1	0.95	0.98	1.00	95%	98%	100%
2	1.86	1.94	1.98	93%	97%	99%
3	2.78	2.89	2.96	93%	96%	99%
4	3.33	3.76	3.90	83%	94%	98%
5	2.97	3.22	3.66	59%	64%	73%
6	3.61	3.74	3.87	60%	62%	65%
7	4.09	4.31	4.47	58%	62%	64%
8	3.65	4.73	**4.99**	46%	59%	62%

Base time
Min=44.35s
Avg=44.62s
Max=45.49s

Early Cluster Experiments

- Xeons 5335, 8 cores/node.
- Estimation
  Firewire protocol
  22 properties
  node x cores
  speed-up, efficiency

1x1	1x2	1x4	1x8
1, 100% 16min	1.8, 92%	3.5, 88%	6.7, 84%

2x1	2x2	2x4	2x8	4x8
1.8, 92%	3.9, 98%	6.8, 85%	12.3, 77%	19.6, 61%

1x1	1x2	1x4
1, 100% 6m30s	1.7, 86%	3.3, 83%

2x1	2x4	4x2
1.7, 84%	5.9, 74%	7.1, 89%

- Estimation
  Lmac protocol
  1 property
- Encouraging results despite simple distribution.

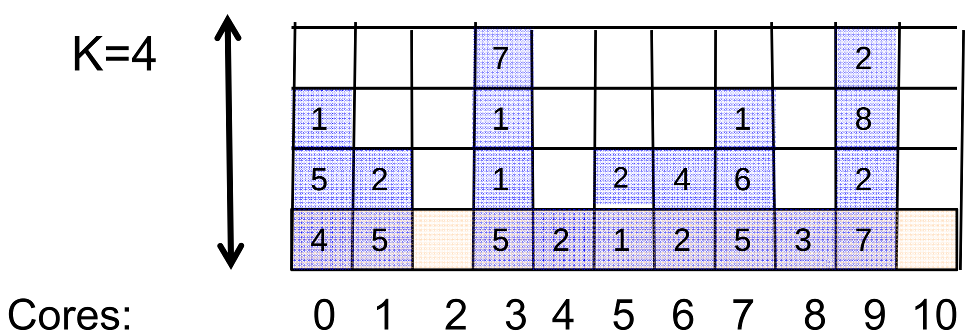
Thanks to Jaco van de Pol, Axel Belifante, Martin Rehr, and Stefan Blom for providing support on the cluster of the University of Twente

Overview

- Statistical Model Checking in UPPAAL
  - Estimation
  - Testing
- Distributed SMC for Parameterized Models
  - Parameter Sweeps
  - Optimization
  - Nash Equilibria
- Distributing Statistical Model Checking
  - Estimation
  - Testing
- DSMC of DSMC
- Conclusion

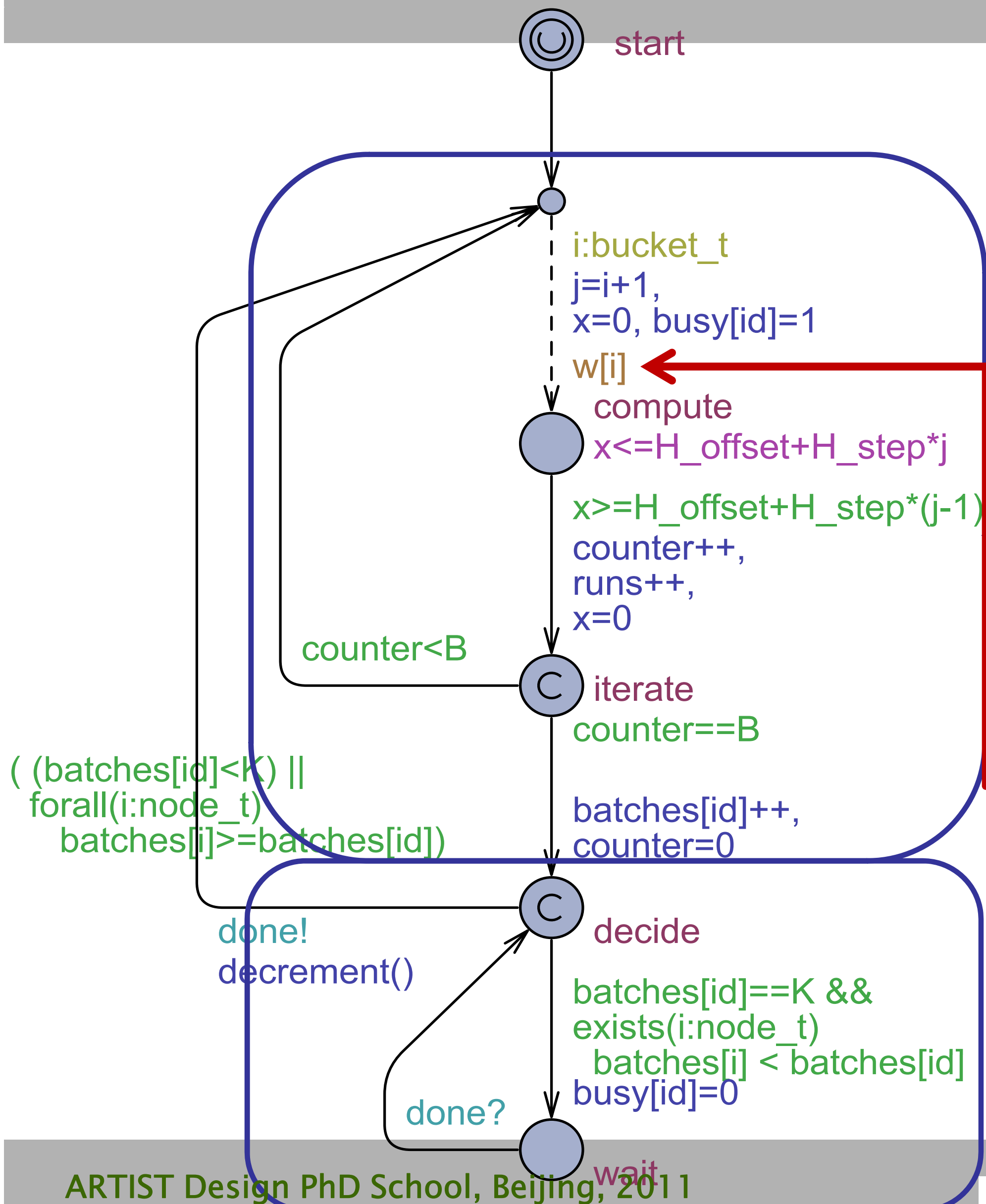
Distributed SMC

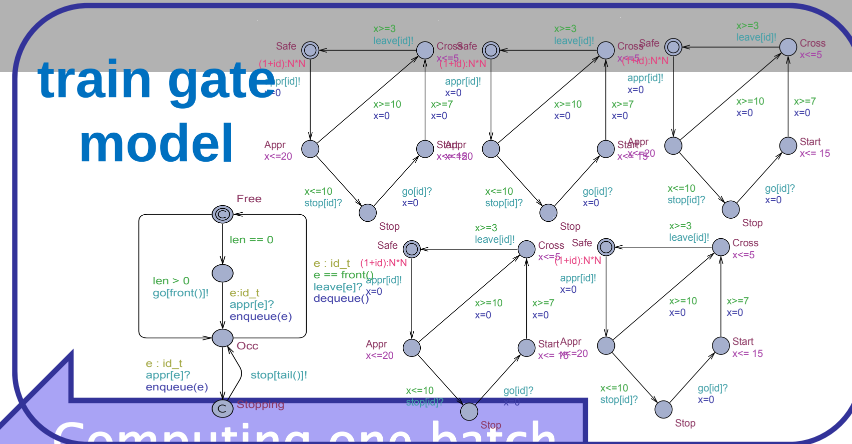
- SMC simulations can be distributed across a cluster of machines with N number of cores.
- The simulations are grouped into batches of B number of simulations in each to avoid bias.
- Each core is not allowed to be ahead by more than K batches than any other core.



Core0 is computing **4th** batch
Core2 is computing **1st** batch
Core1 is computing **3rd** batch
Core9 is blocked, **waiting** for Core2+10
Core3 is blocked, **waiting** for Core2+10
Only **complete** row of batches is used

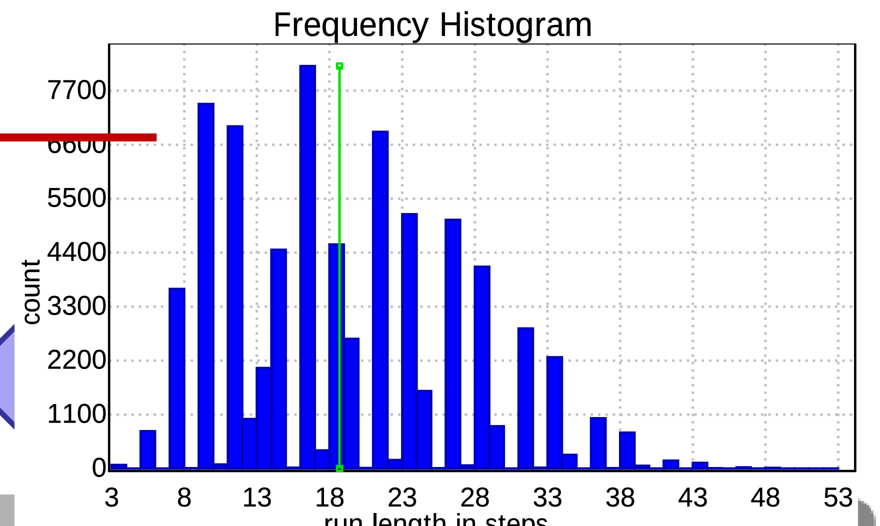
Distributed SMC: Model of a Core





Computing one batch

Pr[# <= 100](<> Train(5).Cross)



generation time ~ simulation steps

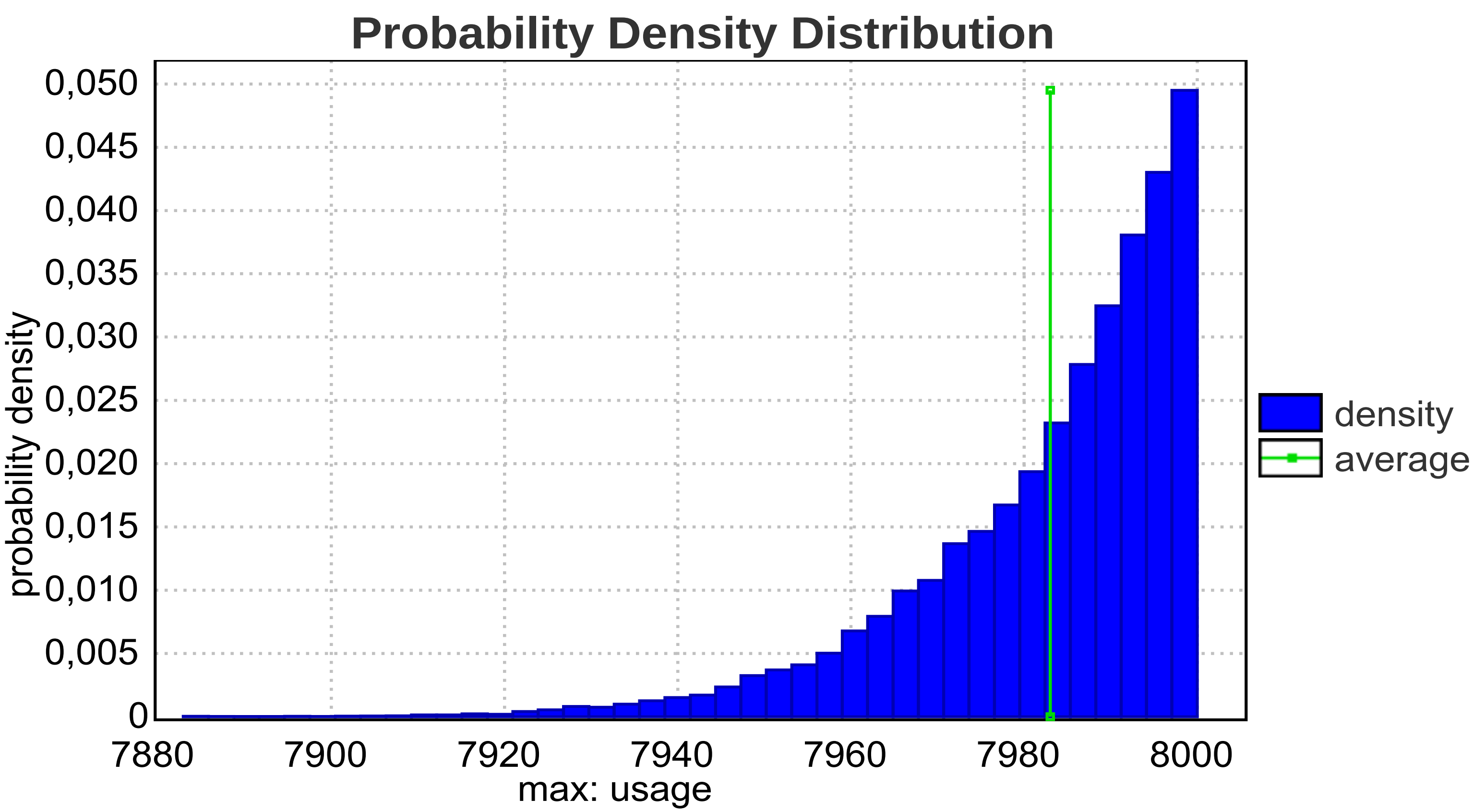
DSMC: CPU Usage Time

Parameter instantiation:

N=8, B=100, K=2

Property used:

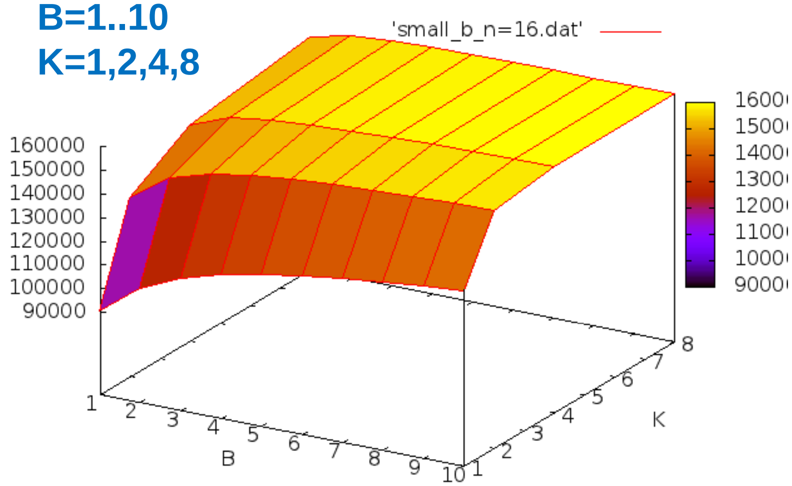
E[time<=1000; 25000] (max: usage)



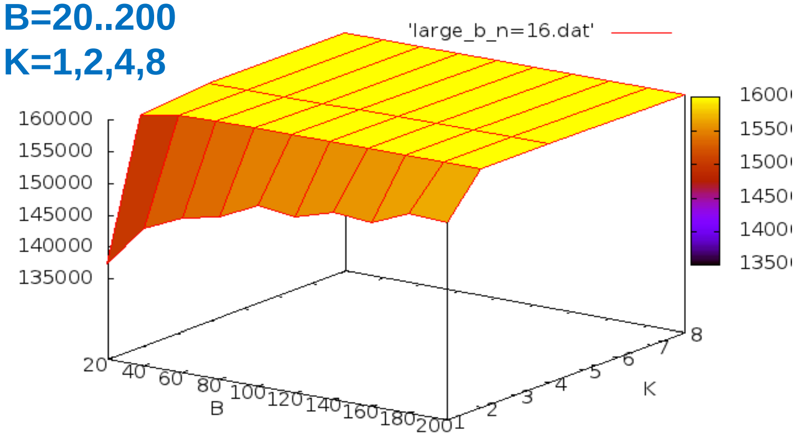
Runs: 25000 in total, 25000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Average: 7983.02.

DSMC Performance Analysis

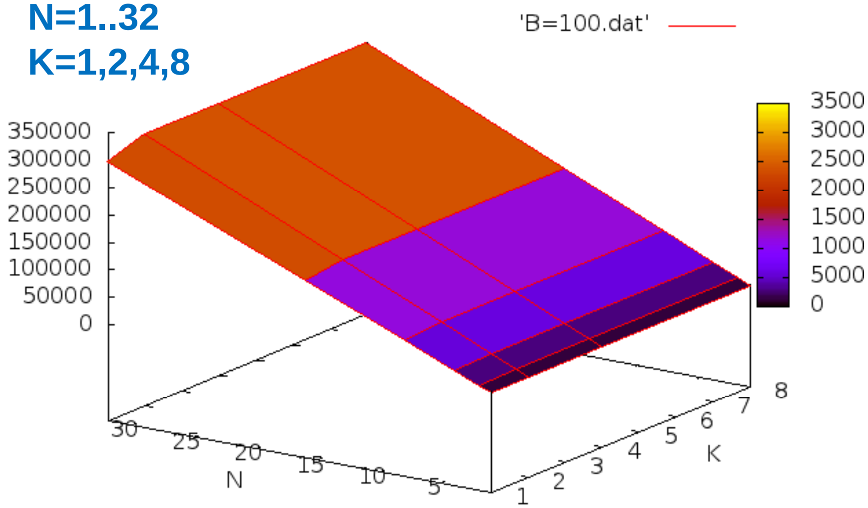
N=16
B=1..10
K=1,2,4,8



N=16
B=20..200
K=1,2,4,8



B=100,
N=1..32
K=1,2,4,8



Property used:
E[time<=1000; 1000] (max: usage)

Conclusions:

K=1 has huge effect and should be avoided.

K=2 has effect if B<20.

K>2 are indistinguishable on homogeneous cluster.

K>2 and B>20: number of simulations scale linearly to the number of cores used.

Conclusion

- Preliminary experiments indicate that distributed SMC in UPPAAL scales very nicely.
- More work to identify impact of parameters for distributing individual SMC?
- How to assign statistical confidence to parametric analysis, e.g. optimum or NE?
- UPPAAL 4.1.4 available
  (support for SMC, DSMC, 64-bit,..)