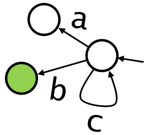
Formal Approaches to Decision-Making under Uncertainty

Arnd Hartmanns
Formal Methods and Tools
UNIVERSITY OF TWENTE

Decision-Making under Uncertainty

We build models of systems with…



choices

random choices

random timing

costs/ rewards

communication protocols

randomised algorithms

distributed systems

fault tolerance

privacy

biological processes

security

Markov chains

Markov decision processes

probabilistic timed automata

...

…to check and optimise the systems with respect to:

response times

survivability

throughput

reliability

power usage

safety

dependability

availability

resilience

system up $\mathrm{U}^{\leq 75\,\mathrm{s}}$ clean shutdown

$\mathbb{P}(\diamond\ \text{crash}) = ?$

$\mathbb{S}(\text{power usage})$

$\mathbb{E}(\text{time to finished})$

Probabilistic Uncertainty

Safety/reliability: $\mathbb{P}(\mathrm{F}^{\leq t}\ bad)$

→ probability of failure within bounded time
e.g. system crash within duration of flight

Availability: $\lim_{t \to \infty} \frac{1}{t} \int_0^t \mathbb{P}(\mathrm{F}^{=t}\ bad) dt$

→ steady-state probability of correct operation
e.g. web server uptime

Rewards: $\mathbb{E}$(reward until goal)

→ expected accumulated reward
e.g. energy consumed until recharge

Probabilistically Uncertain Decisions

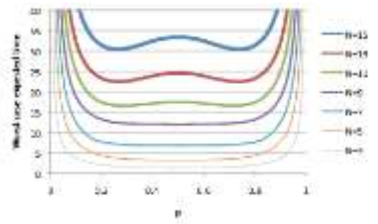
$\mathbb{P}_{opt}(\mathrm{F}\,T)$ for $opt \in \{\min, \max\}$ and target state set T

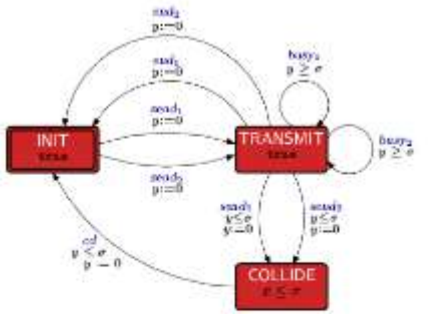
with $T = bad$: <u>minimum</u> probability to reach a bad state

with $T = goal$: <u>maximum</u> probability to reach a goal/safe state

$\mathbb{E}_{opt}$(reward until T) for $opt \in \{\min, \max\}$ and target state set T

<u>minimum</u>/<u>maximum</u> reward accumulated to state in T

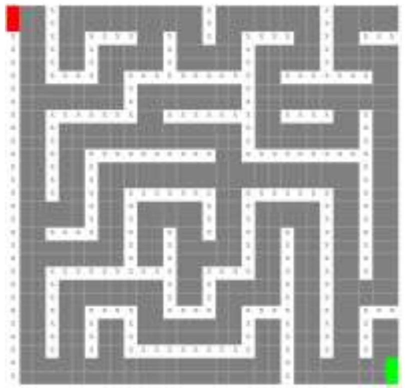




But also:

– What is the <u>optimal strategy</u>?

– Can we quickly find a <u>sufficient strategy</u> satisfying a requirement, e.g. $\mathbb{P}(\mathrm{F}\,T) < 10^{-8}$ or $\mathbb{E}$(r.u. T) ≥ 60?



Computing and Optimising Probabilities

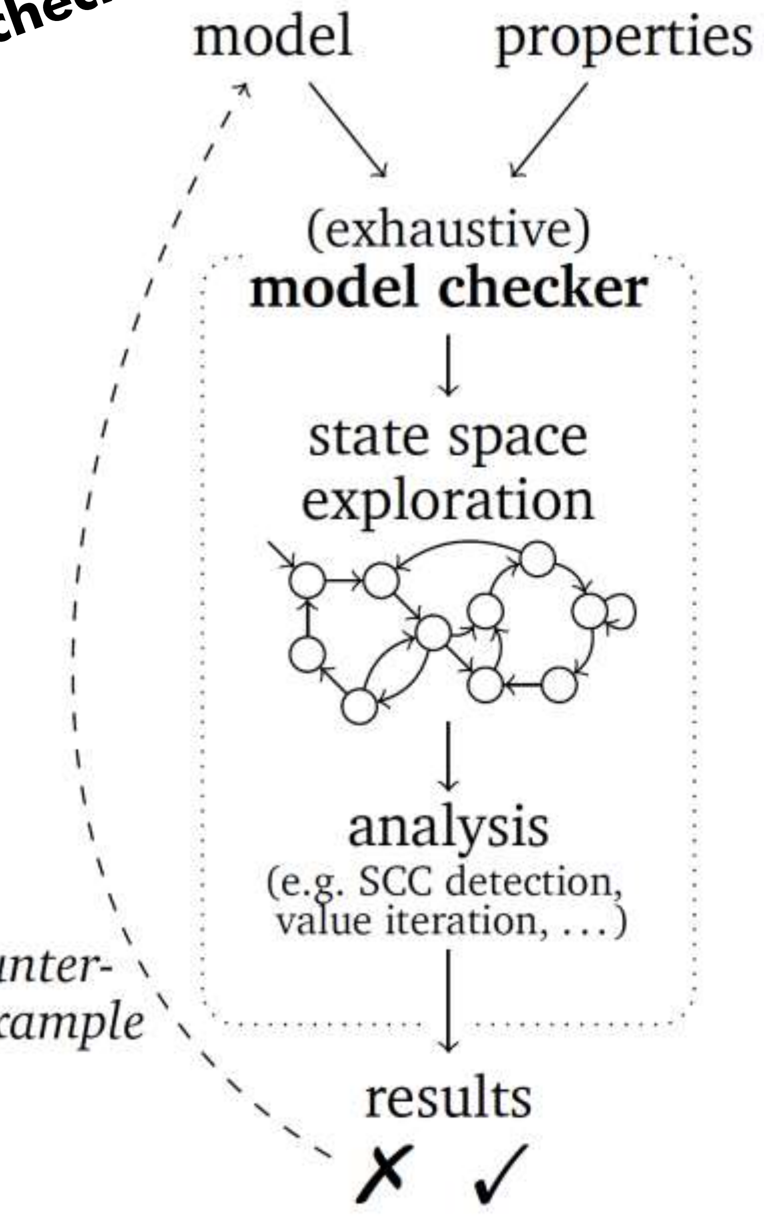
Our main technology: **model checking** **= probabilistic model checking**

→ automatic verification technique to check whether a system meets its specification



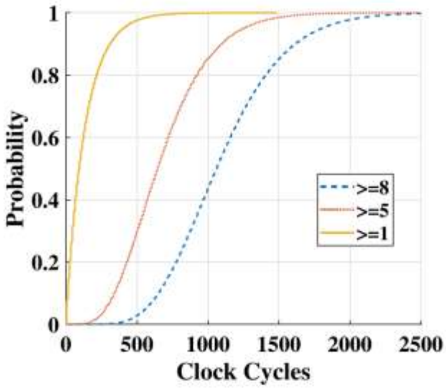
ACM Turing Award 2007
to Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis for model checking

…but we'll also look into
statistical model checking **Monte Carlo simulation**
& reinforcement learning

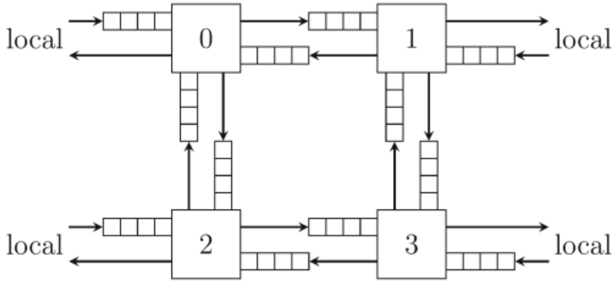


Three Examples

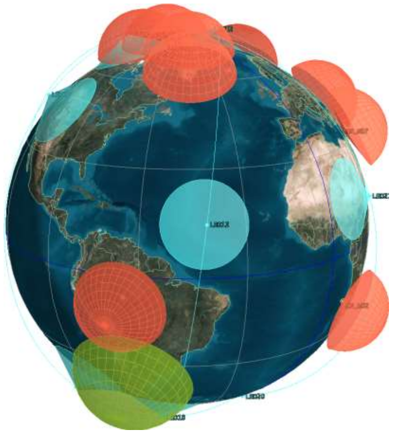
1 Power supply noise in a network-on-chip system

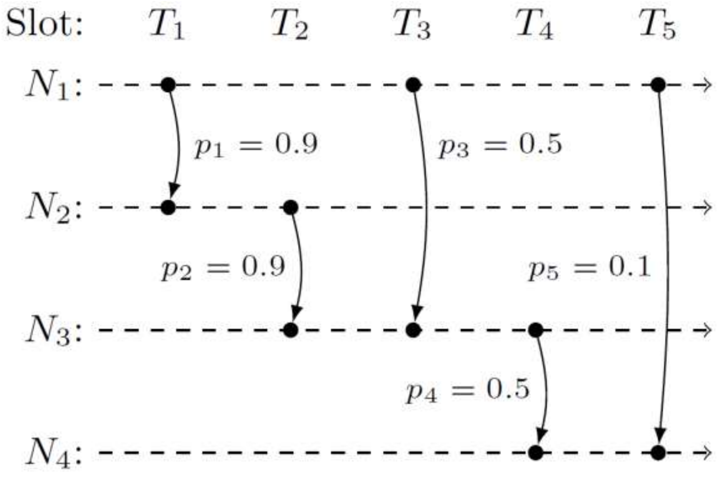


(b) CDF for *inductiveNoise*

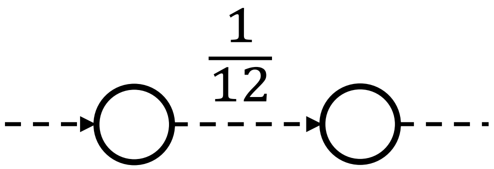


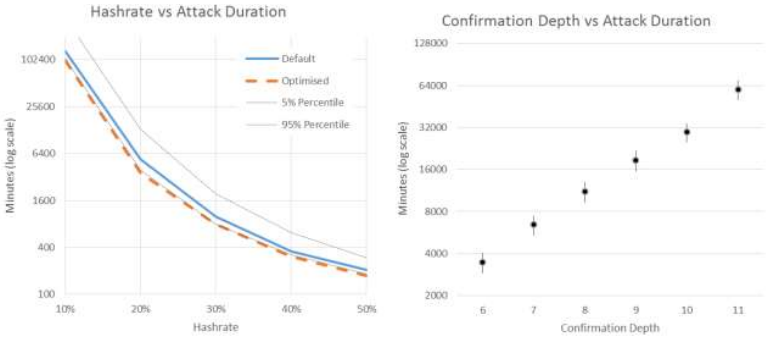
2 Delay-tolerant routing in satellite constellations





3 Optimising an attack to erode trust in Bitcoin

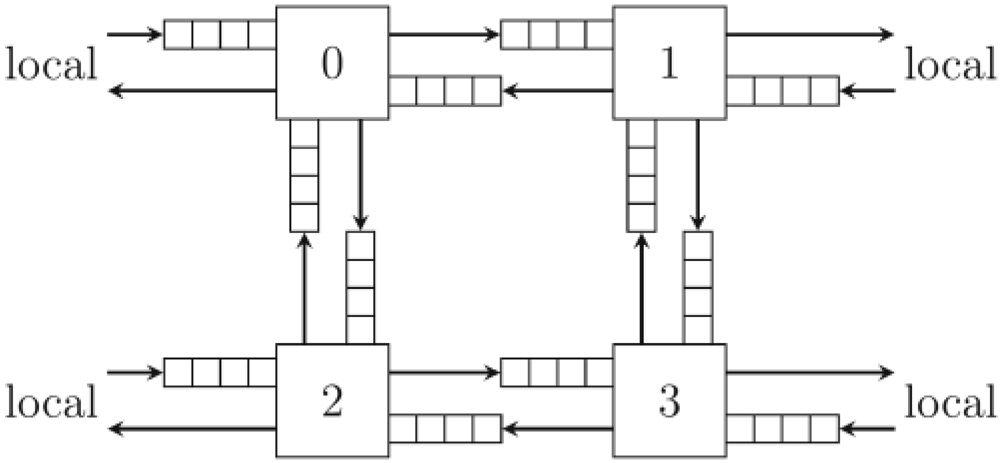




2 Power supply noise in a network-on-chip system

Network-on-Chip Case Study

Recent application: **power supply noise in network-on-chip routing**



2×2 NoC: 4 processors, 4 routers

Every-other-cycle and bursty generation

Uniformly random destinations

Round-robin routing priority policy

DTMC model

Goal: insights into power supply noise

– voltage drop from simultaneous switching

– resistive and inductive noise

↳ by rate of current change

(FMICS 2021)


Probabilistic Verification for Reliability of a Two-by-Two Network-on-Chip System

Riley Roberts[1(✉)], Benjamin Lewis[1(✉)], Arnd Hartmanns[2], Koushik Chakraborty[1], Prabal Basu[3], Sanghamitra Roy[1], and Zhen Zhang[1(✉)]

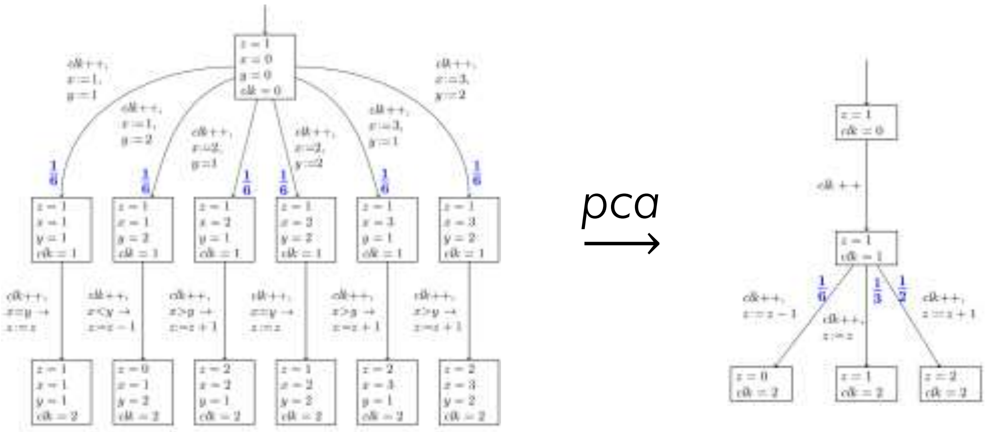
[1] Utah State University, Logan, UT, USA

Network-on-Chip Modelling

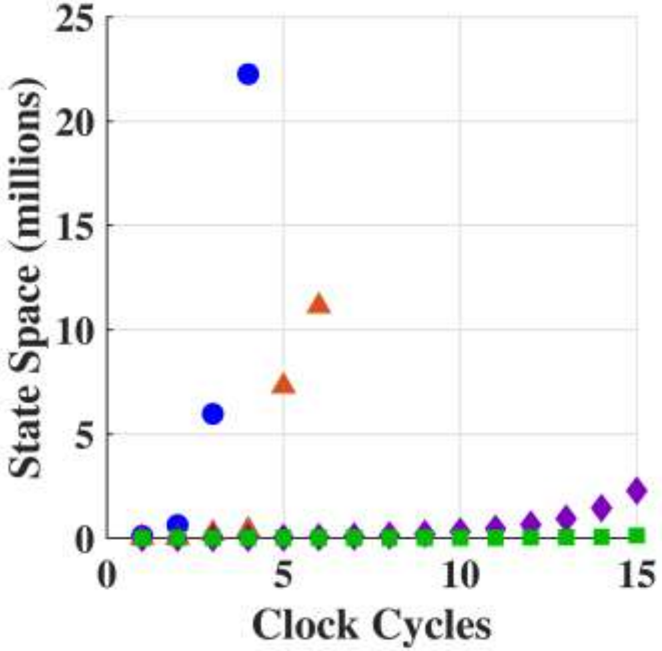
1. Concrete model: FIFO channels of capacity 4, …

```
datatype channel = {int direction,
         int id, bool serviced, int priority,
         queue buffer};
datatype router = {int unserviced,
         int totalUnserviced, channel[] channelArray};
```

2. Predicate abstraction: replace complex data types by predicates
3. Probabilistic choice abstraction: delay randomness until relevant
4. Abstract buffers to counters



DTMC tractable for model checking with mcsta

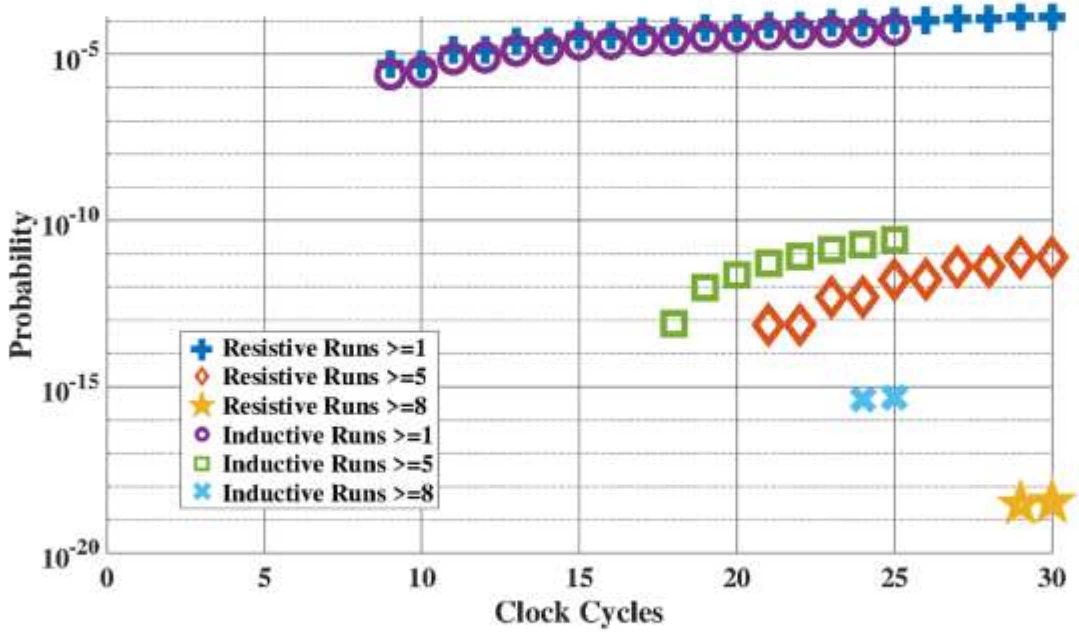


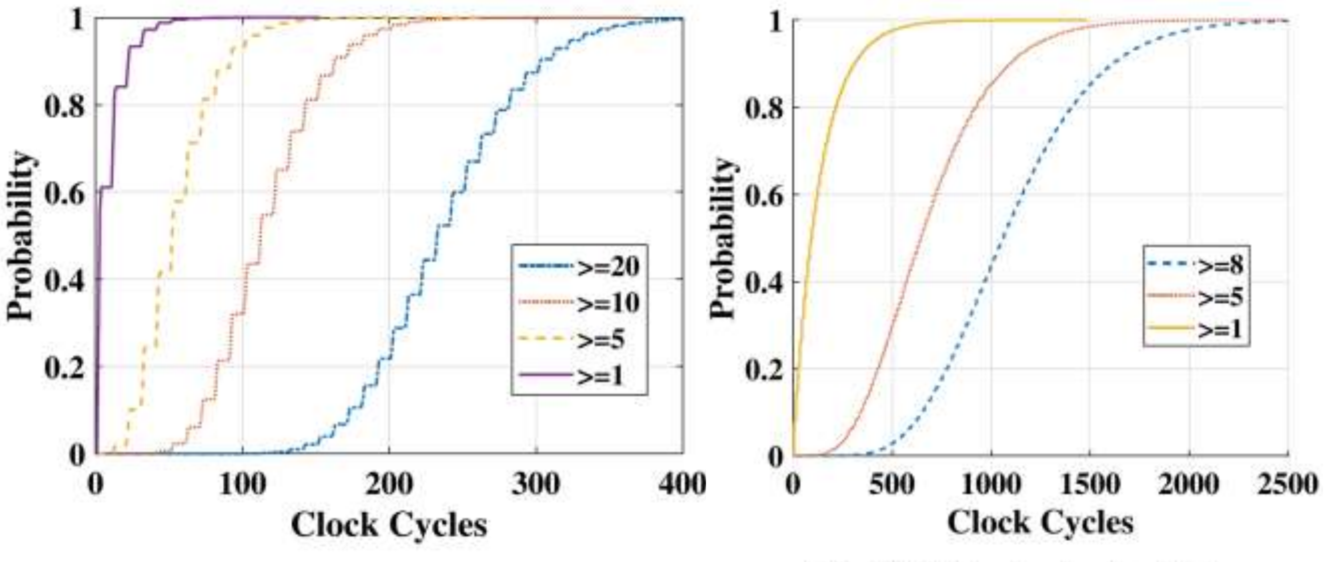
Network-on-Chip Model Checking

mcsta: check for number of noise-inducing events in n clock cycles

every-other-cycle flit generation: unbounded state space too large, but bounded analysis possible

bursty flit generation: queues regularly empty out, allowing full state space generation for reward-bounded analysis

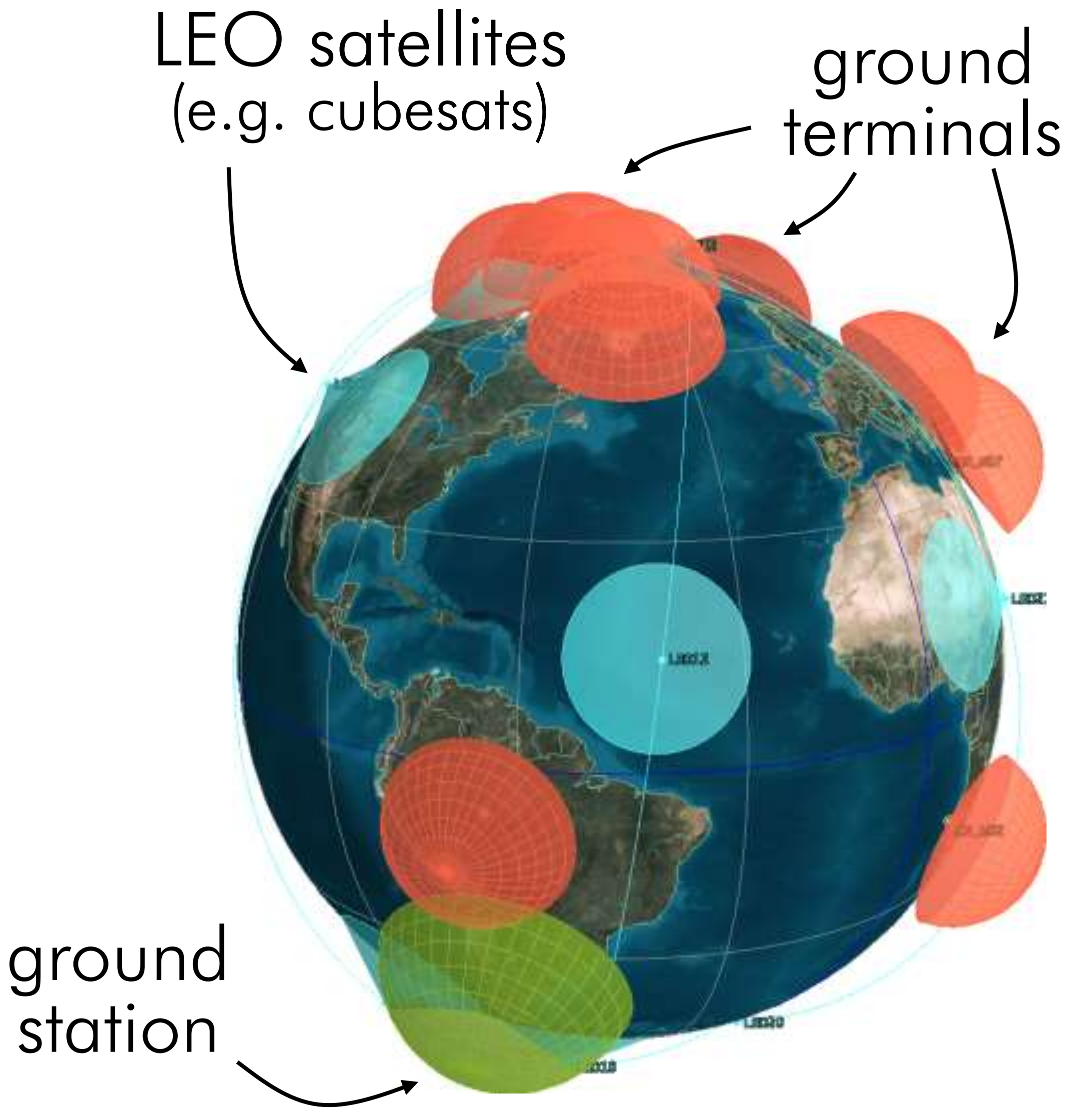




(a) CDF for *resistiveNoise* (b) CDF for *inductiveNoise*

3 Delay-tolerant routing in satellite constellations

Routing in Satellite Constellations



Delay-tolerant network:
data hops from satellite to satellite when close **contact**

Random message loss:

inaccurate orbits *interference*
node faults *incomplete data*
…

→ optimise delivery probability with $\leq n$ copies

(NASA Formal Methods 2020)

Sampling Distributed Schedulers for Resilient Space Communication

Pedro R. D'Argenio[1,2,3], Juan A. Fraire[1,2,3], and Arnd Hartmanns[4]

[1] CONICET, Córdoba, Argentina

Distributed Information

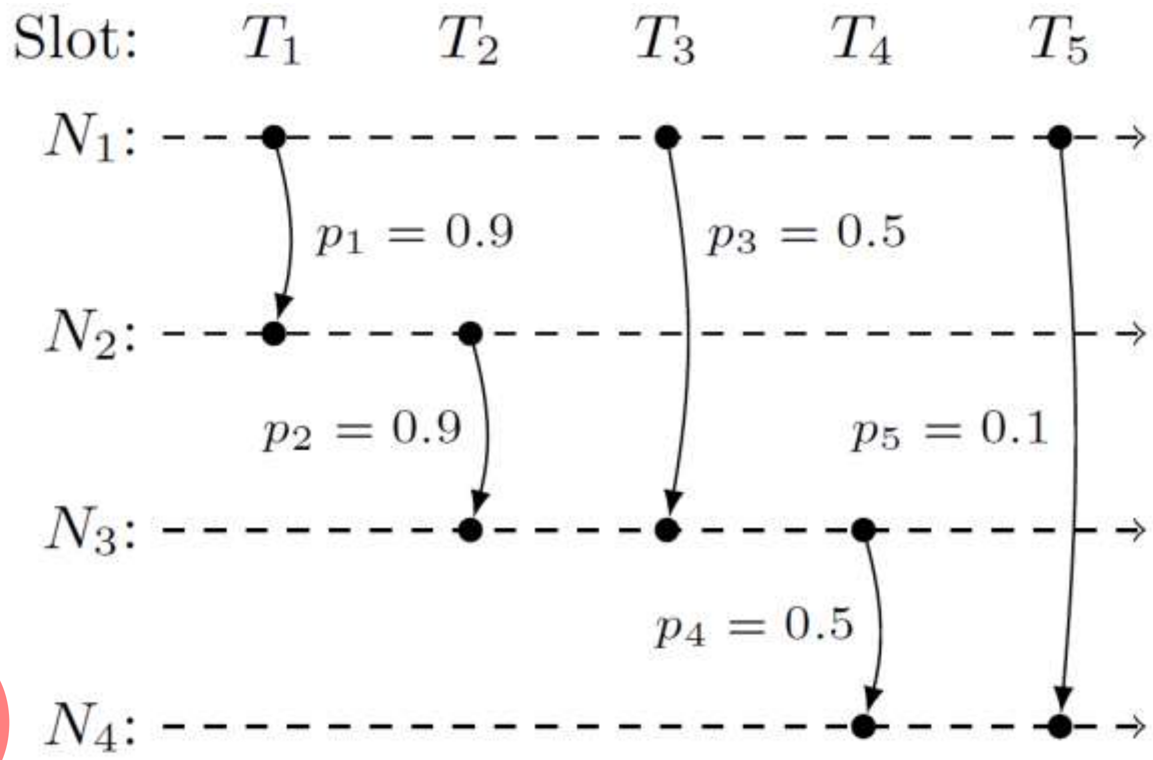
Strategy choices + probabilities = Markov decision process **MDP**

→ use probabilistic model checking to find best routing strategy **= "scheduler"**

Best scheduler for N_1 (2 copies):

1. T_1: send copy #1 to N_2
2. T_3: send copy #2 to N_3
   only if N_3 did not get copy #1
3. T_5: send copy #2 to N_4
   if it was not sent in T_3

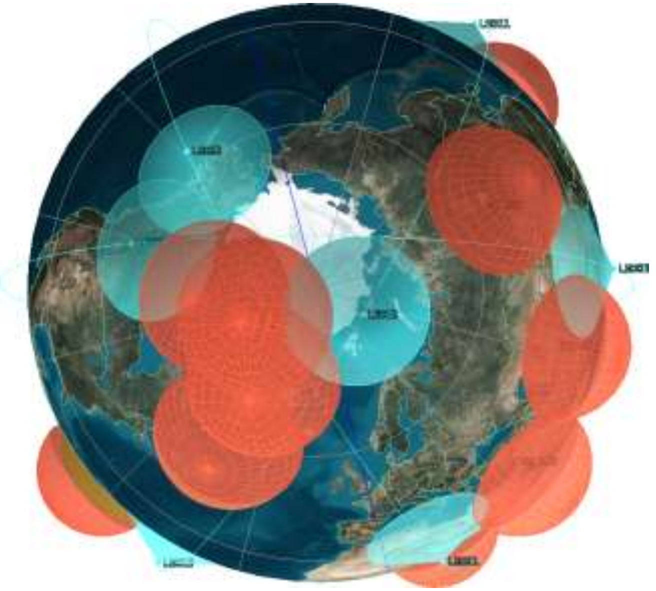
global information ⚡ *N_1 cannot know this!* ☹



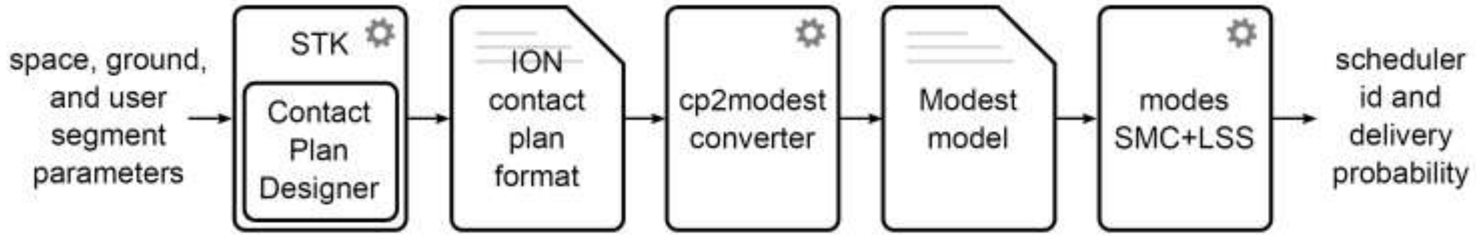
uncertain contact plan (abstraction)

Scheduler Sampling for Space Routing

Solution: use SMC with scheduler sampling
→ only feed local information to scheduler



Toolchain



Modest models

```
process Node1(int(0..COPIES) copies) {
   alt {
   :: nop1; rcv                                    // slot 1: contact with node 2
   :: when(copies >= 1) snd1to2_
      rcv palt {
      :0.9: {= data1 = 1, dest1
      :0.1: {= /* lost */ =} }
   :: when(copies >= 2) snd1to2_
      rcv palt {
      :0.9: {= data1 = 2, dest1
      :0.1: {= /* lost */ =} }
   :: rcv2to1;
      rcv {= 1: copies += dest2=
   };
   rcv;
   ... }
```

	PMC	SMC-LSS-1000		SMC-LSS-10000		SMC-LSS-100000	
model	global	global	distrib.	global	distrib.	global	distrib.
example/unrel.	0.493	0.48 (0.49)	0.46 (0.47)	0.49 (0.49)	0.46 (0.47)	0.49 (0.49)	0.46 (0.46)
example/acks	0.505	0.49 (0.50)	0.46 (0.48)	0.50 (0.50)	0.50 (0.50)	0.50 (0.50)	0.50 (0.51)
walker/unrel.	0.438	0.03 (0.06)	0.21 (0.30)	0.10 (0.16)	0.30 (0.37)	0.26 (0.33)	0.37 (0.38)
walker/acks	0.734	0.36 (0.38)	0.47 (0.48)	0.38 (0.40)	0.54 (0.60)	0.45 (0.47)	0.54 (0.56)

SMC results

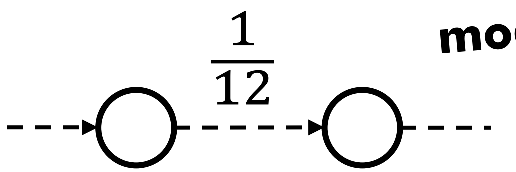
4 Optimising an attack to erode trust in Bitcoin

Modelling and Attacking Bitcoin

Bitcoin: cryptocurrency blockchain

average time to add block: 12 minutes

CTMC model



just kill Bitcoin

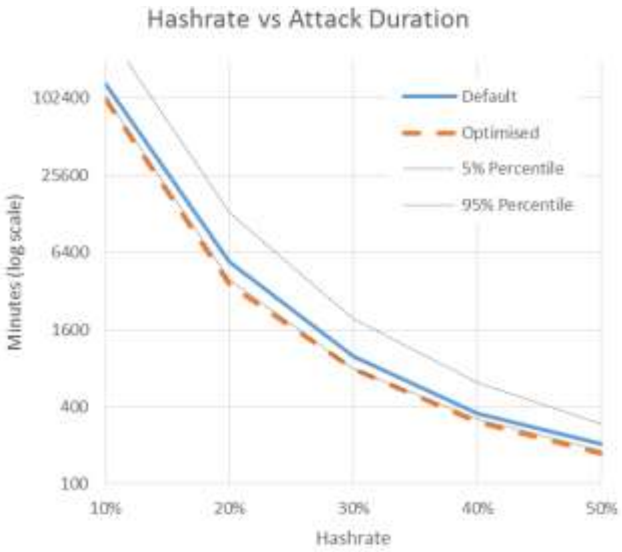
~~Double-spending~~ attack:

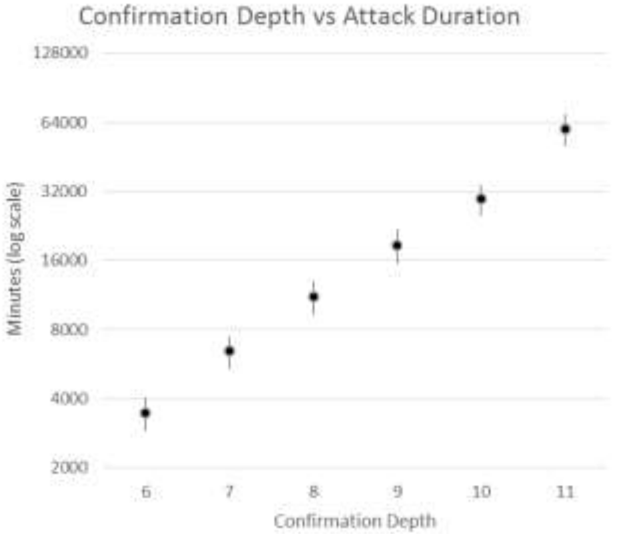
secretly work on own fork until it is longer than the main one

→ when to abandon own fork and restart attack?

MA model

Earlier work: manual check of different strategies using UPPAAL SMC





(NASA Formal Methods 2018)

Twenty Percent and a Few Days – Optimising a Bitcoin Majority Attack

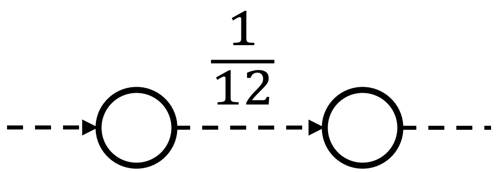
Ansgar Fehnker[1] and Kaylash Chaudhary[2]

[1] Formal Methods and Tools Group, University Twente

[2] School of Computing, Information, and Mathematical Sciences, University of the South Pacific

Optimising an Attack on Bitcoin

Bitcoin: cryptocurrency blockchain
average time to add block: 12 minutes



~~Double-spending~~ attack:

secretly work on own fork until it is longer than the main one

→ when to abandon own fork and restart attack?

```
process TrustAttacker()
{
  do {
  :: rate((1/12) * M) {= m_len = min(CD, m_len + 1), m_d
  :: sln {= m_diff-- =}; // public fork extended
     alt { // strategy choice: restart or continue malicious fork
     :: rst {= m_len = 0, m_diff = 0 =} // can always resta
     :: when(m_diff > -DB) cnt // can continue if not too far
     }
  }
}
```

With a Modest MA model:

synthesise the optimal attack strategy

(RW Summer School 2019)


A Modest Markov Automata Tutorial*

Arnd Hartmanns[1] and Holger Hermanns[2,3]

[1] University of Twente, Enschede, The Netherlands
[2] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[3] Institute of Intelligent Software, Guangzhou, China

Abstract. Distributed computing systems provide many important ser- ... understand why and how well they work, it is ... models of the systems'

Formal Approaches to Decision-Making under Uncertainty

Plan for the Week

Monday: Discrete-Time Markov Chains (DTMCs)

Tuesday: Markov Decision Processes (MDPs)

Wednesday: Model-Checking, Learning, and Statistical Algorithms

Thursday: Program your own probabilistic model checker

To pass:
1. Deliver models from Tuesday
2. Deliver exercise solutions from Wednesday
2. Deliver model checker from Thursday

can work in pairs

Prepare for Later

can work in pairs

To avoid overloading the wifi later:

1. Download and unzip **the Modest Toolset** from
   `https://www.modestchecker.net/Downloads/`
2. Install **GraphViz** via your package manager or from
   `https://graphviz.org/download/`
3. Make sure you have **Python 3.7** or newer, see
   `https://www.python.org/downloads/`
4. Optional: Download and install **Visual Studio Code** from
   `https://code.visualstudio.com/Download`

Course Material

Slides, links, etc. are made available at

https://arnd.hartmanns.name/rio2023/