Rare Event Simulation for Highly Dependable Systems with Fast Repairs

Daniël Reijsbergen Pieter-Tjerk de Boer
Werner Scheinhardt Boudewijn Haverkort

ROCKS meeting, 5 October, 2010

Outline

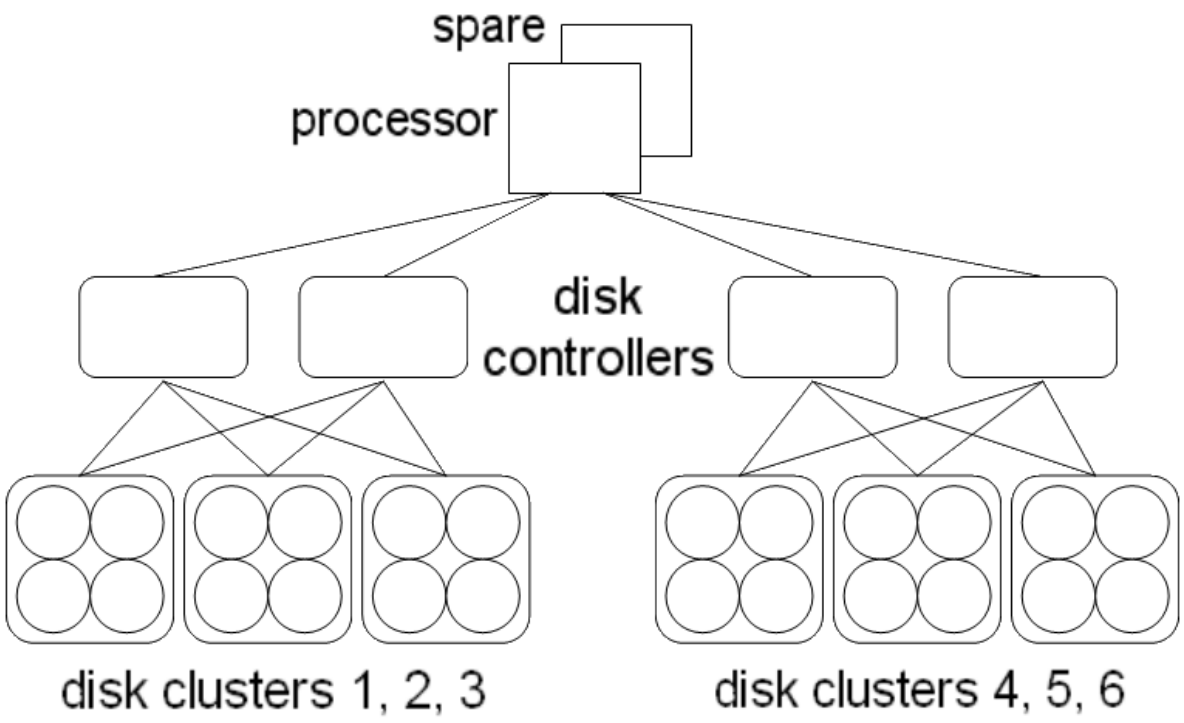
Outline

Problem Setting

- Highly Dependable (stochastic) System:
  - System consists of several *component types*.
  - *Redundancy* due to spare components of each type.
  - *Components* fail and are repaired after time intervals of random length.
  - When more than k_i components of some type i are down, system *as a whole is down*.
- Goal:
  - Quantify dependability of the system.
- Possible dependability measures:
  - **Unreliability**: probability of system failure before time τ.
  - **Unavailability**: long-run time fraction that system is down.

Example

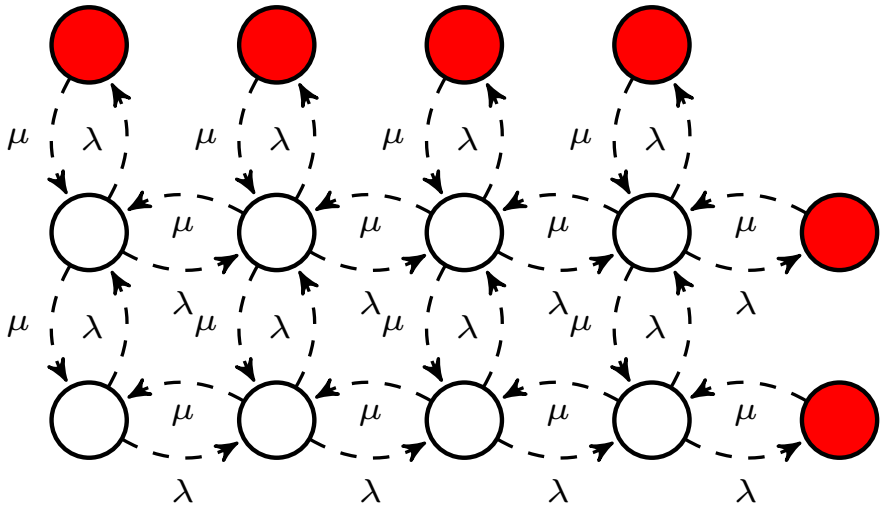
Distributed Database System:



Modelling

- High-level modelling approach:
  - Model system as generalised Petri net.
- Low-level modelling approach:
  - Model the system as a (labeled) CTMC.
- Specify starting state.
- Specify dependability measure using CSL.
- Verify whether CSL-formula holds in starting state.

Toy Example (two component types)



- Possible dependability measures:
  - Prob. of hitting the red state before time τ, the *unreliability*: $\mathcal{P}_{\bowtie p}(\Diamond^{<\tau}$ red $)$
  - Steady-state probability of being in a red state, the *unavailability*: $\mathcal{S}_{\bowtie p}$(red)

Solution Methods

- Analytic (possible in case of steady-state unavailability)
- Numerical (e.g. Gauss-Seidel)
  - Need the *low-level* model description (CTMC)
- Statistical (i.e. simulation)
  - Need only the *high-level* description (Petri net).
  - Consequence: *state space explosion problem* alleviated.

Outline

Monte Carlo for $\Phi_\tau = \Diamond^{<\tau}$red

- Monte Carlo:
  - Generate N *sample paths* ω_i.
  - Produce an *estimate* $\hat{\pi}$ for the probability $\mathbb{P}(\Phi_\tau)$:

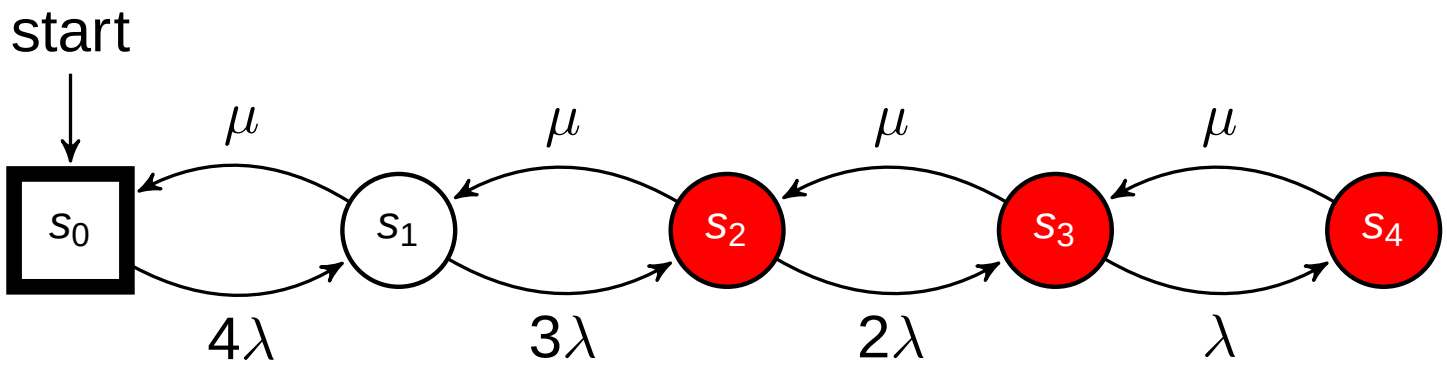
$$\hat{\pi} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\Phi_\tau}(\omega_i)$$

$$= \frac{\text{number of runs that } \textit{satisfied } \Phi_\tau}{\textit{total} \text{ number of runs}}$$

  - Finally, test whether $\mathcal{P}_{\bowtie p}(\Phi_\tau)$ holds.

Generating Sample Paths (1-1)

$$x_0 = s_0 : t_0 = 0$$

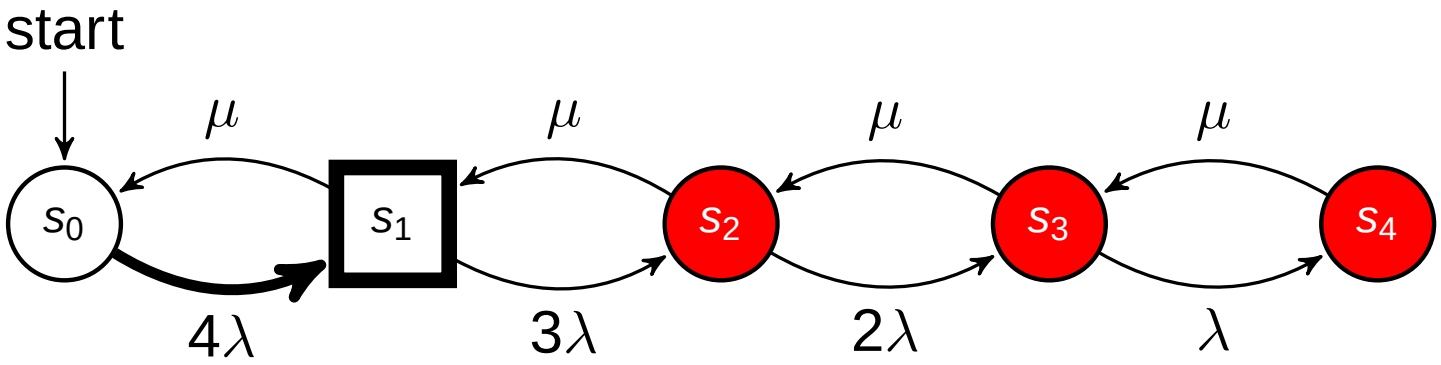


$\Phi_\tau = \Diamond^{<\tau}$ red, $\tau = 1$, $\lambda = 1$, $\mu = 6,000$

Then we stop when we reach a red state or time runs out.

Generating Sample Paths (1-2)

$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.46190$

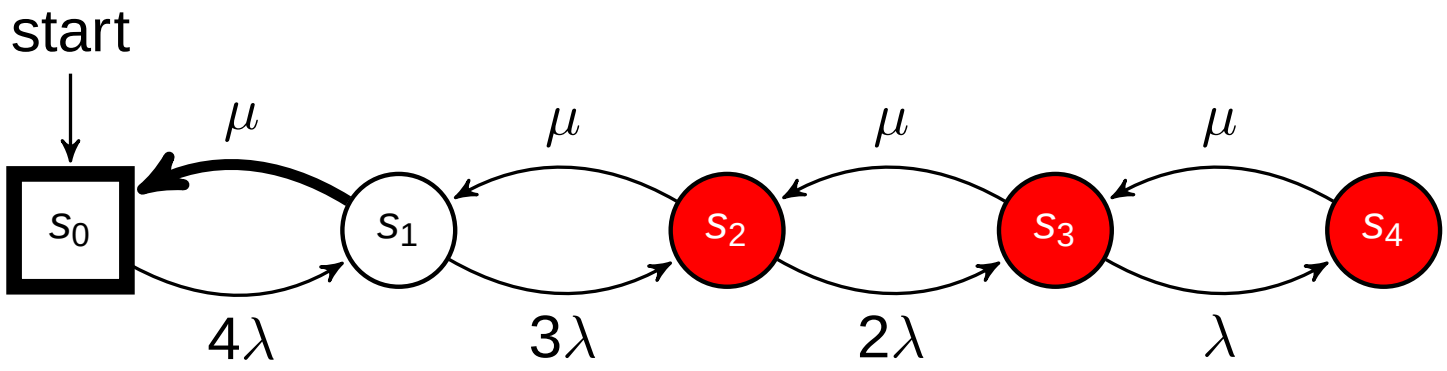


$\Phi_\tau = \Diamond^{<\tau}$ red, $\tau = 1$, $\lambda = 1$, $\mu = 6,000$

Then we stop when we reach a red state or time runs out.

Generating Sample Paths (1-3)

$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.46190$
$x_2 = s_0 : t_2 = 0.46191$

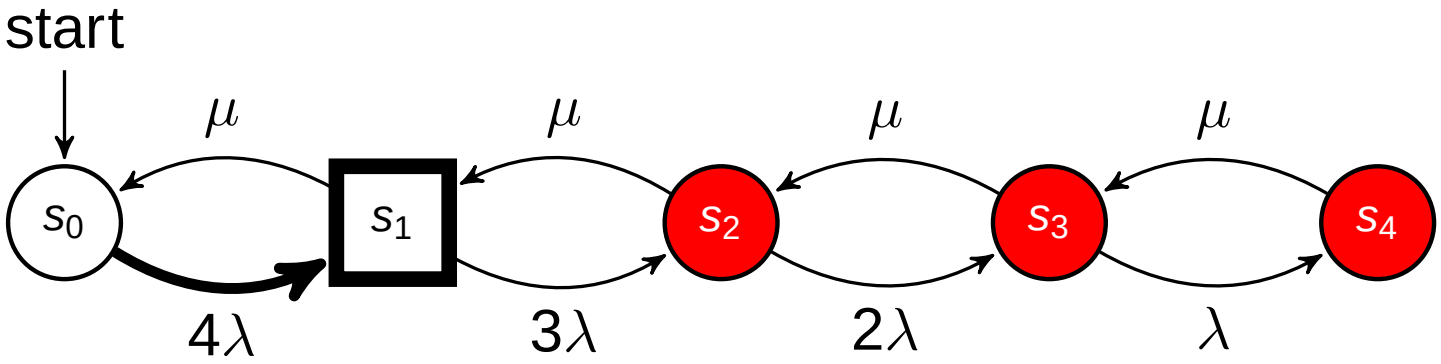


$\Phi_\tau = \Diamond^{<\tau}$ red, $\tau = 1$, $\lambda = 1$, $\mu = 6,000$

Then we stop when we reach a red state or time runs out.

Generating Sample Paths (1-4)

$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.46190$
$x_2 = s_0 : t_2 = 0.46191$
$x_3 = s_1 : t_3 = 0.64262$

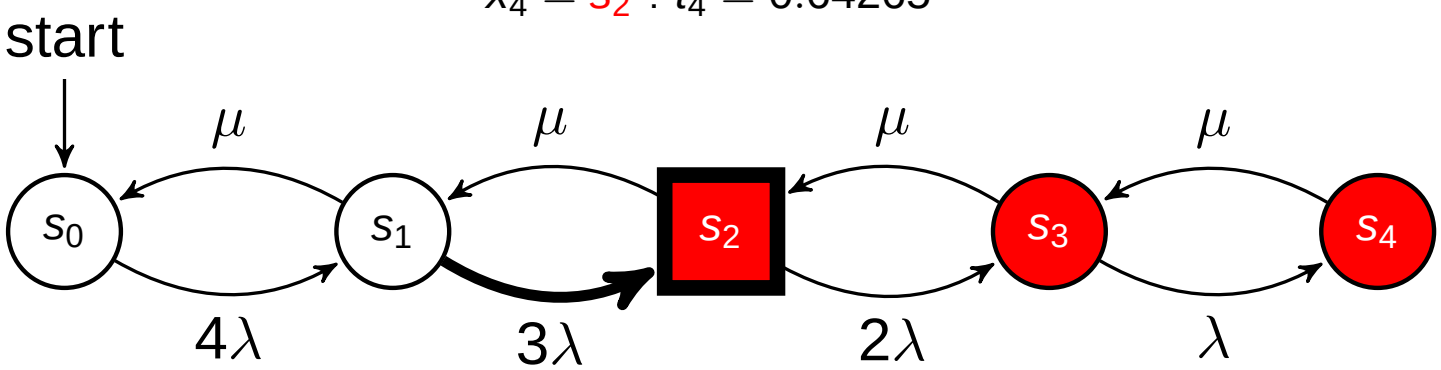


$\Phi_\tau = \Diamond^{<\tau}$ red, $\tau = 1$, $\lambda = 1$, $\mu = 6,000$

Then we stop when we reach a red state or time runs out.

Generating Sample Paths (1-5)

$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.46190$
$x_2 = s_0 : t_2 = 0.46191$
$x_3 = s_1 : t_3 = 0.64262$
$x_4 = s_2 : t_4 = 0.64265$

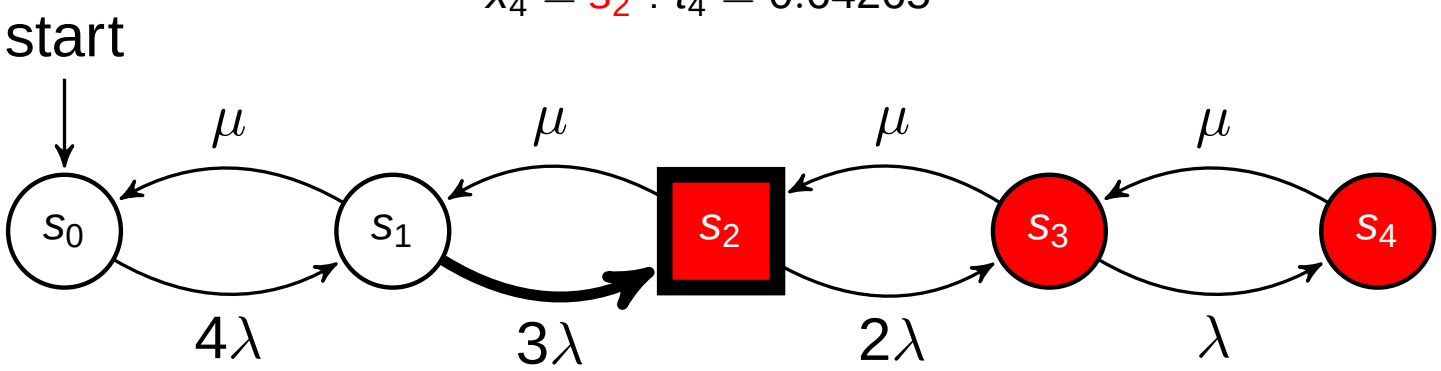


$\Phi_\tau = \Diamond^{<\tau}$ red, $\tau = 1$, $\lambda = 1$, $\mu = 6,000$

Then we stop when we reach a red state or time runs out.

Generating Sample Paths (1-6)

$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.46190$
$x_2 = s_0 : t_2 = 0.46191$
$x_3 = s_1 : t_3 = 0.64262$
$x_4 = s_2 : t_4 = 0.64265$



Sample path: $\omega_i = ((x_0, t_0), (x_1, t_1), (x_2, t_2), (x_3, t_3), (x_4, t_4))$

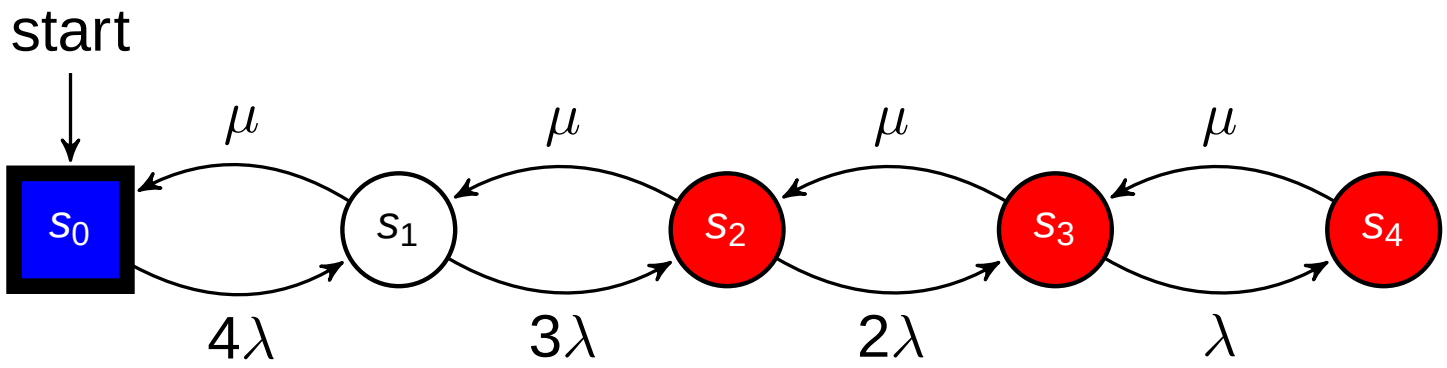
$\mathbf{1}_{\Phi_\tau}(\omega_i) = 1$

Monte Carlo for $\mathcal{S}_{\bowtie p}(\text{red})$

- We assume that the Markov Chain is ergodic.
- Consider *busy cycles*. Busy cycle starts and ends when process enters a regenerative state.
- $Z =$ total time spent in red state during busy cycle, D duration of busy cycle.
- Renewal-reward: unavailability $= \dfrac{\mathbb{E}(Z)}{\mathbb{E}(D)}$

Generating Sample Paths (2-1)

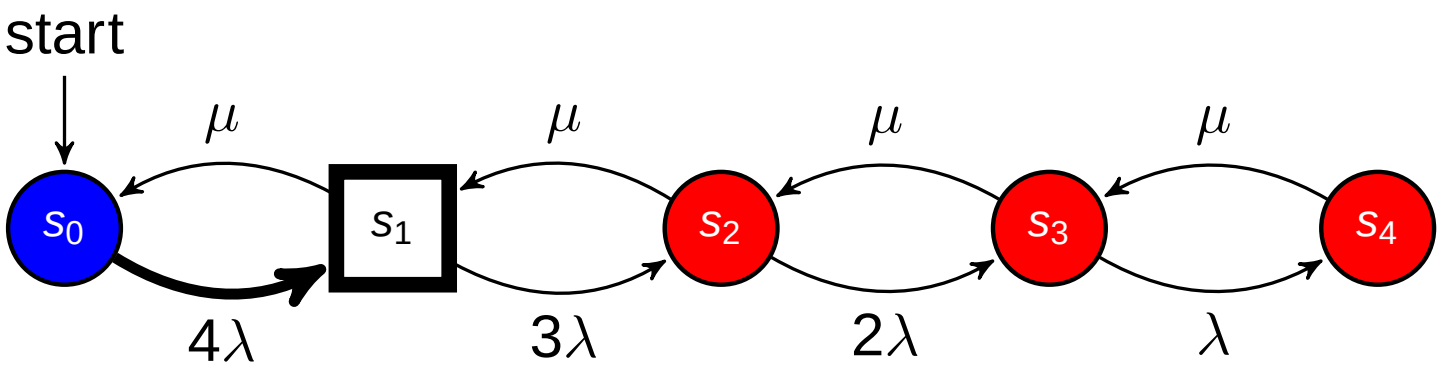
$$x_0 = s_0 : t_0 = 0$$



We first leave the regenerative state.

Generating Sample Paths (2-2)

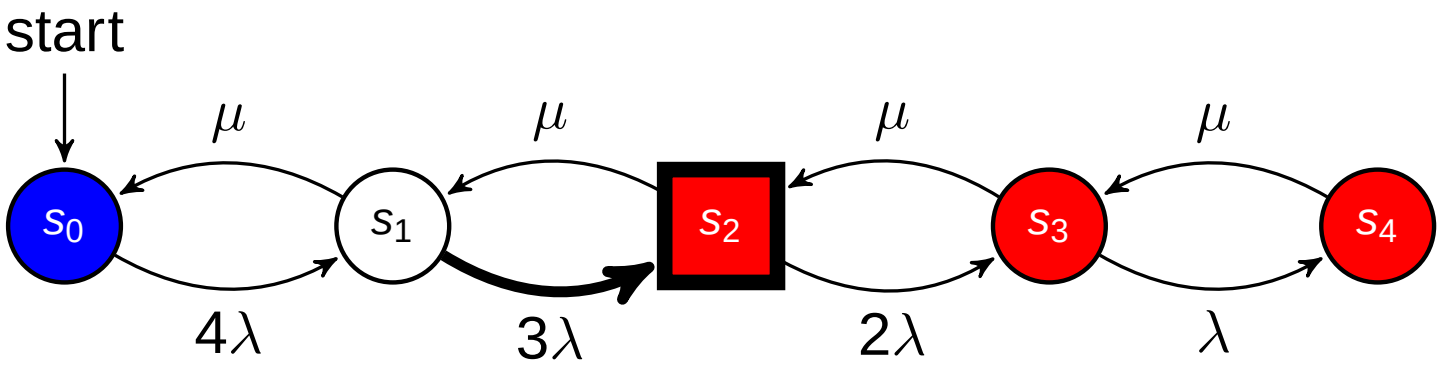
$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.05823$



Then we stop when we return to the regenerative state.

Generating Sample Paths (2-3)

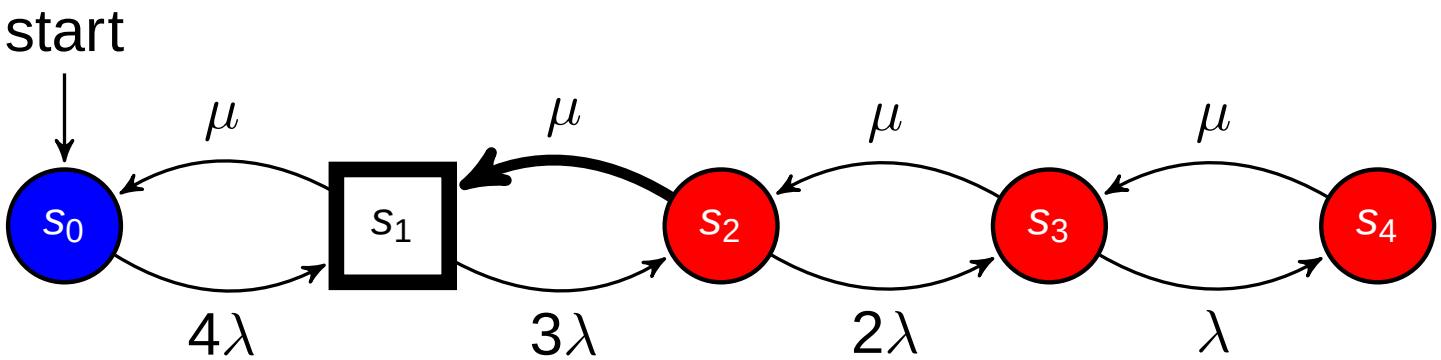
$$x_0 = s_0 : t_0 = 0$$
$$x_1 = s_1 : t_1 = 0.05823$$
$$x_2 = s_2 : t_2 = 0.05830$$



Then we stop when we return to the regenerative state.

Generating Sample Paths (2-4)

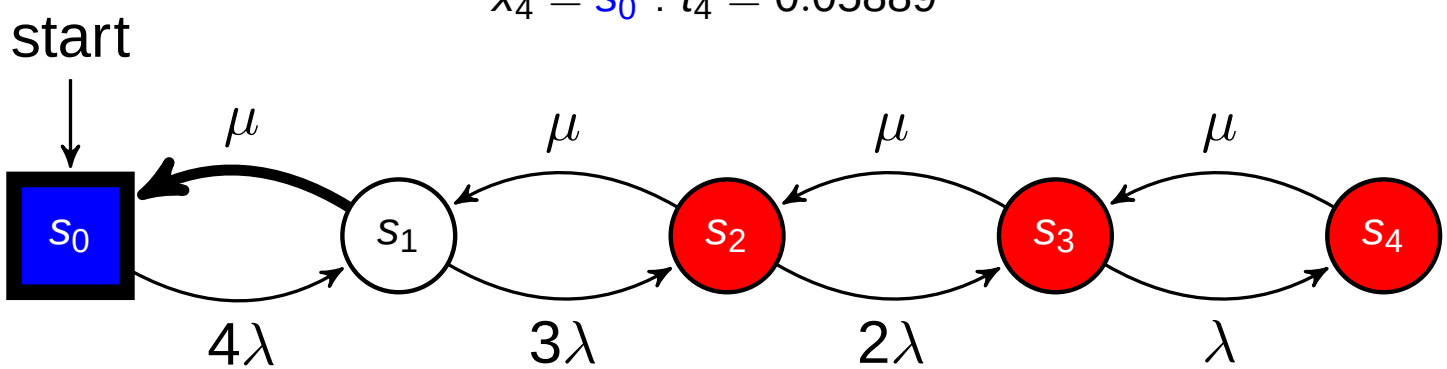
$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.05823$
$x_2 = s_2 : t_2 = 0.05830$
$x_3 = s_1 : t_3 = 0.05886$



Then we stop when we return to the regenerative state.

Generating Sample Paths (2-5)

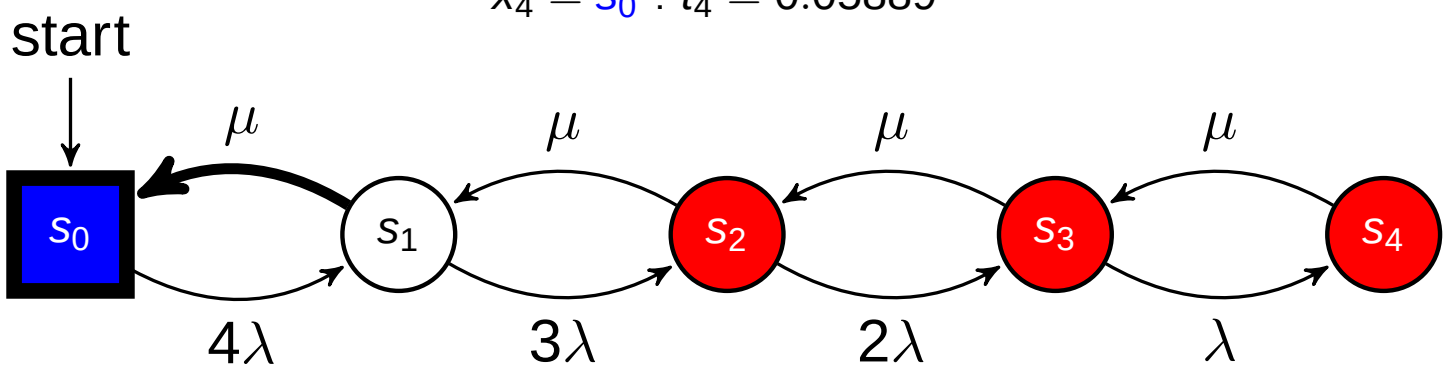
$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.05823$
$x_2 = s_2 : t_2 = 0.05830$
$x_3 = s_1 : t_3 = 0.05886$
$x_4 = s_0 : t_4 = 0.05889$



Then we stop when we return to the regenerative state.

Generating Sample Paths (2-6)

$x_0 = s_0 : t_0 = 0$
$x_1 = s_1 : t_1 = 0.05823$
$x_2 = s_2 : t_2 = 0.05830$
$x_3 = s_1 : t_3 = 0.05886$
$x_4 = s_0 : t_4 = 0.05889$



Sample path: $\omega_i = ((x_0, t_0), (x_1, t_1), (x_2, t_2), (x_3, t_3), (x_4, t_4))$
$Z(\omega_i) : 0.00056$
$D(\omega_i) : 0.05889$

Rare Events

- *Problem*: the unreliability/unavailability is *small*.
- *Large number* of simulation runs needed before path ω_i with $\mathbf{1}_{\Phi_\tau}(\omega_i) = 1$ or $Z(\omega_i) > 0$ is sampled.
- Therefore, we want to *speed up* the simulation.
- Our approach: *Importance Sampling*.

Importance Sampling (focus on $\Phi_\tau = \Diamond^{<\tau}$ red)

- Standard: path ω is sampled with likelihood $\mathbb{P}(\omega)$.
- New measure: path ω is sampled with likelihood $\mathbb{P}^*(\omega)$.
- Importance sampling principle:

  $$\hat{\pi} = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbb{P}(\omega_i)}{\mathbb{P}^*(\omega_i)} \mathbf{1}_{\Phi_\tau}(\omega_i)$$

  is an unbiased estimator of $\mathbb{P}(\Phi_\tau)$.
- Typically smaller variance if $\mathbb{P}^*(\omega) > \mathbb{P}(\omega)$ when ω satisfies Φ_τ.

Zero Variance

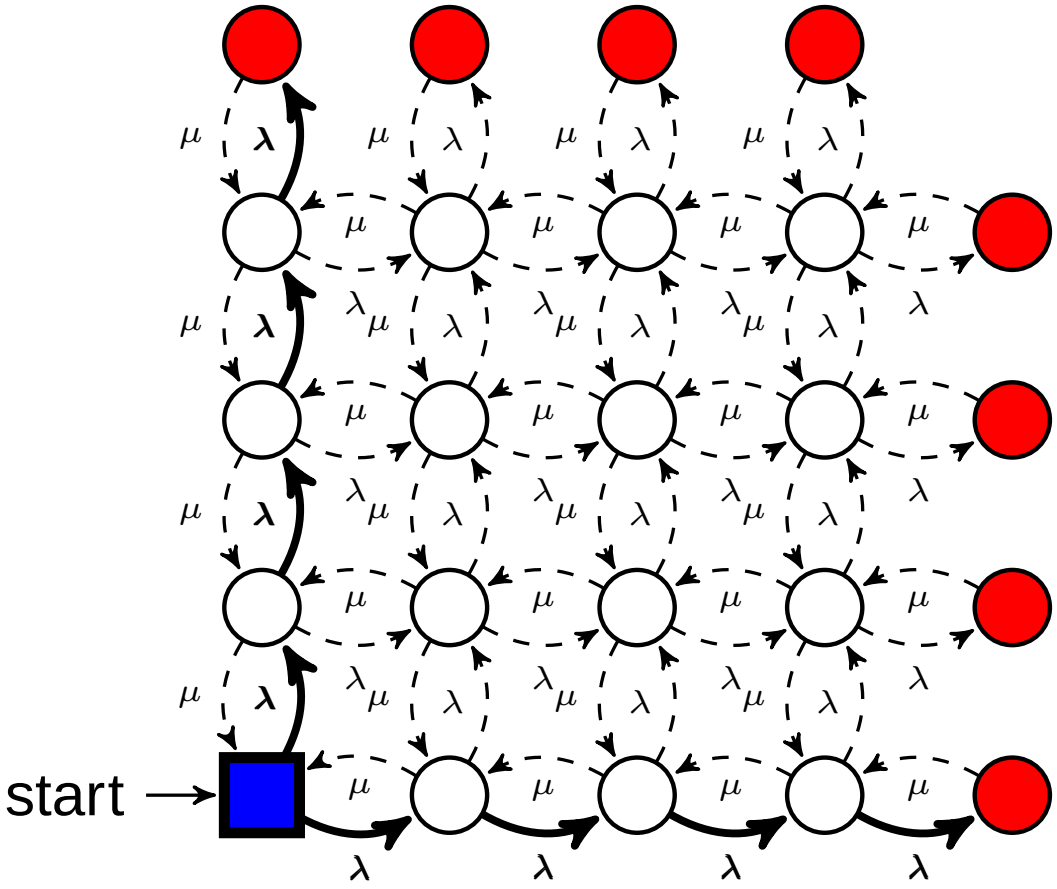
- Best possible change of measure results in *zero variance*.
- The zero variance measure is given by:

$$Pr^*(x \to x', \delta) = Pr(x \to x', \delta) \frac{\mathbb{P}_{x',t-\delta}(\Phi_\tau)}{\mathbb{P}_{x,t}(\Phi_\tau)}, \qquad (1)$$

  where $\mathbb{P}_x(\cdot) =$ the probability of $\cdot$ *starting* in state x, with $t - \delta$ time units until τ.
- Of course, we don't know $\mathbb{P}_{x',t-\delta}(\Phi_\tau)$ (or else we wouldn't simulate).
- So we construct an educated *guess* for $\mathbb{P}_{x',t-\delta}(\Phi_\tau)$.
- Then replace $\mathbb{P}_{x',t-\delta}(\Phi_\tau)$ by guess in (1)

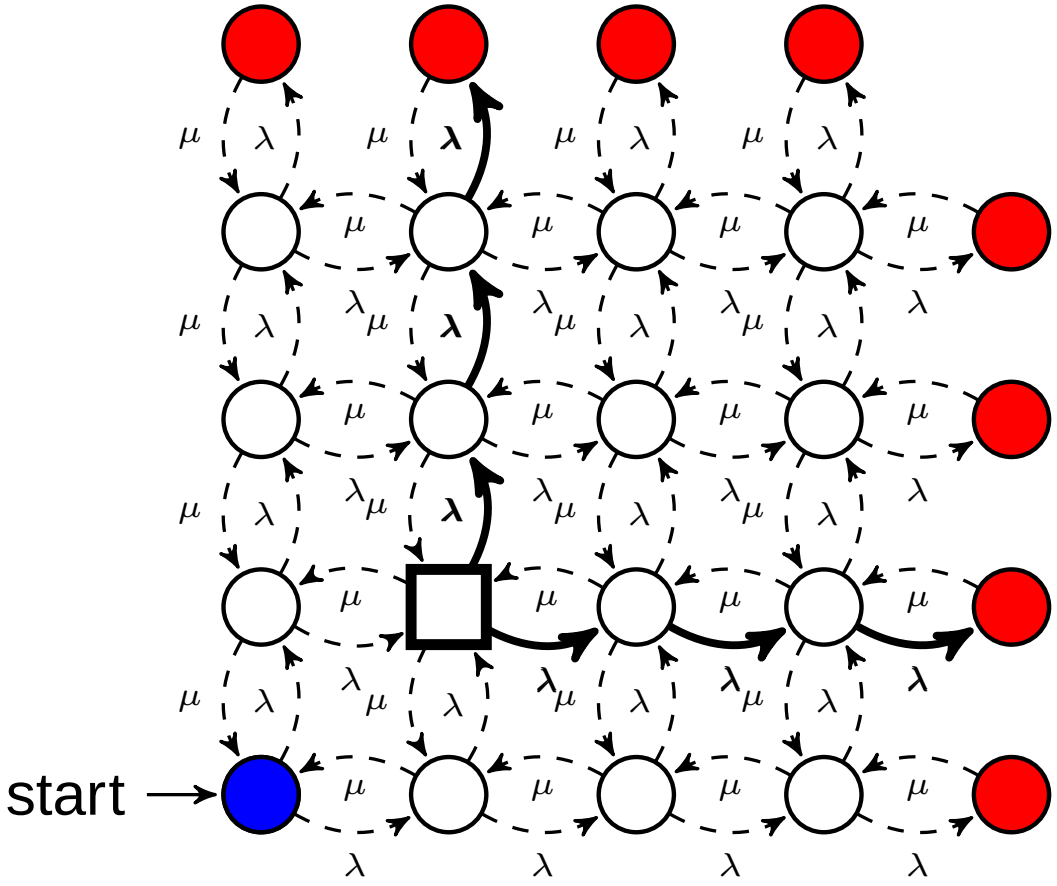
Outline

Straight Paths (1)



Guess for $\mathbb{P}_x(\Phi)$: prob. of *straight paths* from x to red states.

Straight Paths (2)



Do this for each state *x* encountered during the simulation.

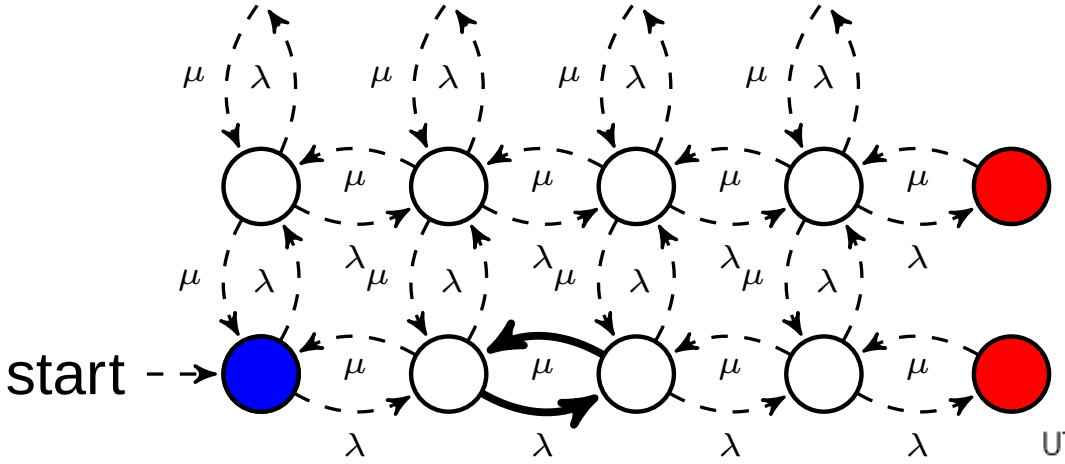
But What About Cycles?

Two types of cycles:

- Cycles through the blue state
- Other cycles

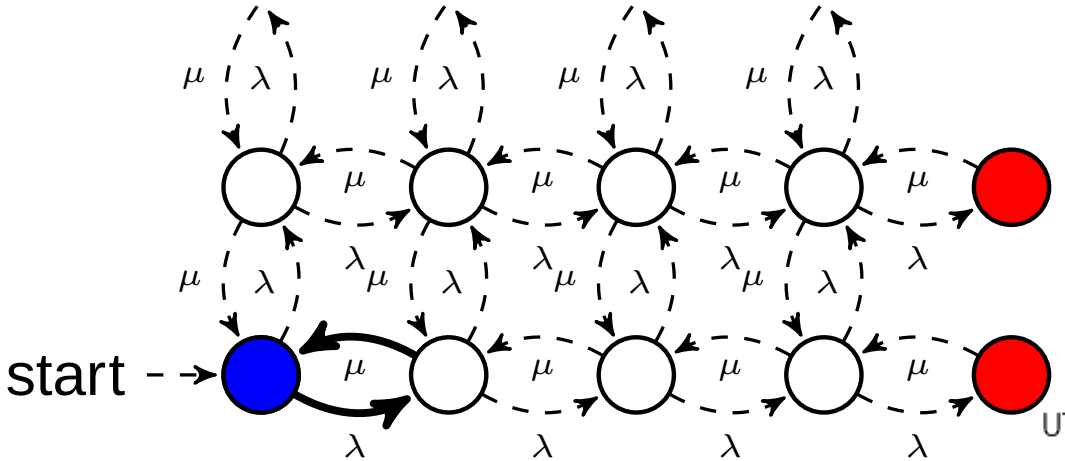
Interior Cycles

- Assume that our path eventually hits the red state, this occurs with probability $\approx \left(\frac{\lambda}{\mu+\lambda}\right)^3$.
- *Extra cycles* in the '*interior*' of the state space are $\approx \frac{\lambda}{\mu+\lambda}$ times less likely.
- Hence, these extra cycles are very unlikely when $\mu \gg \lambda$ and the interior cycles need not be considered for a *good guess* for $\mathbb{P}_{x,t}(\Phi_\tau)$.



Cycles Through Blue State

- Prob. of cycle through blue state is *not* small even if $\mu \gg \lambda$.
- Cycles through the blue state are *only* unlikely when there is not much time left $\rightarrow$ *time dependence*.
- So when τ is high the straight paths produce a poor approximation for $\mathbb{P}_{x,t}(\Phi_\tau)$.
- Results: Change of measure based on this guess pushes the system to red states too quickly, estimator will have high variance.



Final Approximation

Final guess for $\mathbb{P}_{x,t}(\Phi_\tau)$, based on these and some other considerations:

$$\hat{\mathbb{P}}_{x,t}(\Phi_\tau) = q(x) + q(0) \cdot \lambda_0 \cdot (\tau - t)$$

where

$$\begin{aligned} q(x) &\triangleq \text{probability of straight paths to overflow.} \\ 0 &\triangleq \text{regenerative state.} \\ \lambda_0 &\triangleq \text{total exit rate of the regenerative state.} \\ \tau - t &\triangleq \text{time left on the clock.} \end{aligned}$$

This can be used to construct a well-performing new measure.

Other Aspects

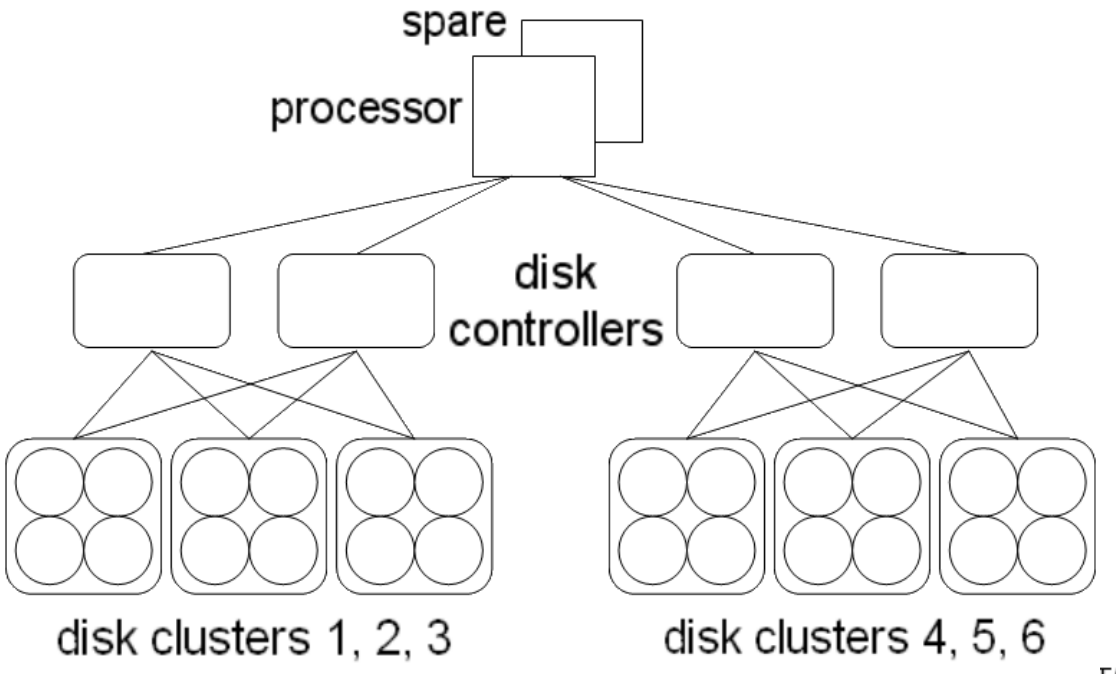
- Sojourn times:
  - Only use $\hat{\mathbb{P}}_{x,t}(\Phi_\tau)$ for *transition probabilities*.
  - At each step sojourn times are sampled *conditional* on not exceeding τ (“forcing”, see [16]).
- Unavailability ($\mathcal{S}_{\bowtie p}(\mathit{red})$)
  - Estimated using importance sampling with the *same change of measure*.

Outline

Test Case

Distributed Database System. Consists of 9 component types:

- $n = 2$ processors
- 2 disk controller sets, n controllers per set.
- 6 disk clusters, $2n$ disks per cluster.



Failure/Repair Rates

- Total failure rate for component type depends on *number of working components*.
- *Dedicated repair*: one repair unit for each component type.

Rates per time unit:

unit	failure rate (per item)	repair rate (per group)
disks	λ	μ
disk controllers	3λ	μ
processors	3λ	μ

Numerical Results

- New method (**Path-IS**) is compared to:
  - Monte Carlo (MC)
  - Balanced Failure Biasing + Forcing (BFB)
    (= general Importance Sampling technique in which failures and repairs are made equally likely)
  - Model checking tool PRISM
- Parametric regimes considered:
  - Low λ (rare failures)
  - High μ (fast repairs)
  - High n (high redundancy)

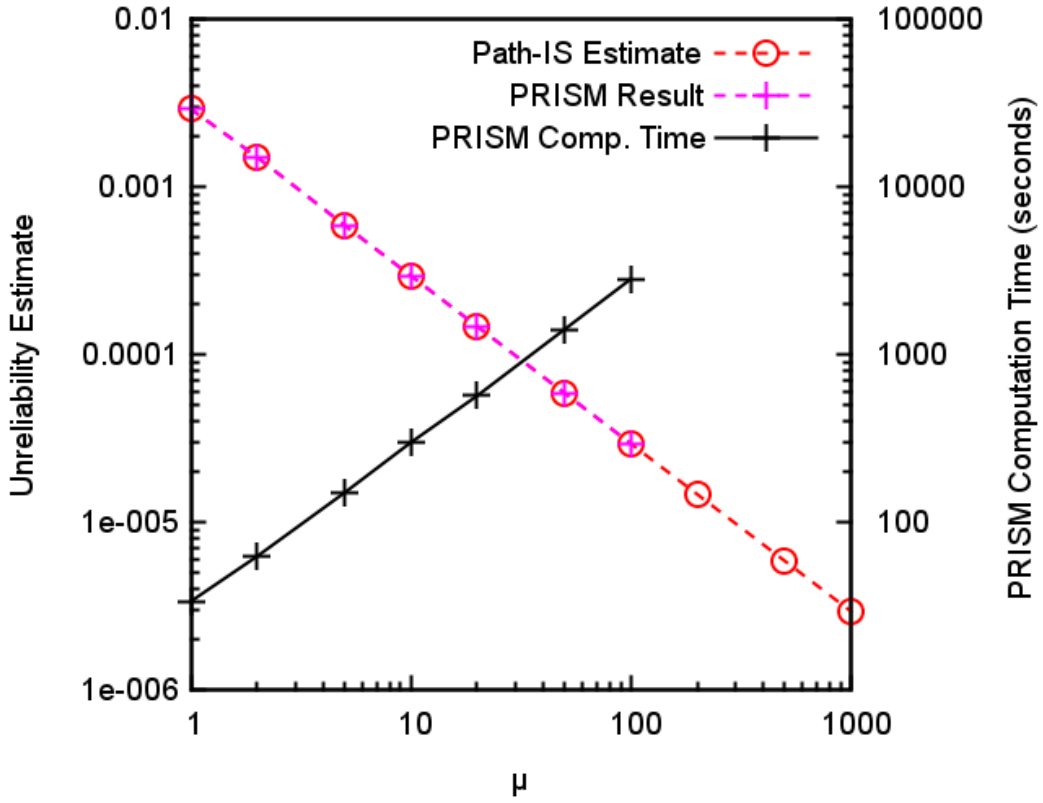
Small λ (Rare Failures)

	$\hat{\pi}$ $(\times 10^{-9})$	# runs	run time
MC	0 ± 0	18 438 588	34 sec
BFB	2.936 ± 0.024	1 042 866	34 sec
Path-IS	$\mathbf{2.937 \pm 0.001}$	**992 231**	34 sec
	π $(\times 10^{-9})$	# states	
PRISM	2.936	421 875	34 sec

Unreliability ($\hat{\pi}$) results when $\mu = 1$; $n = 2$, $\lambda = \frac{1}{6} \cdot 10^{-6}$.

When unreliability is low because λ is low, PRISM is most efficient.

High μ (Fast Repairs)



Path-IS run time: 10 seconds. 95%-confidence interval invisibly small.
PRISM run time increases linearly as μ is made bigger.

Why PRISM is inefficient for high μ

- PRISM uses *uniformisation* for transient probabilities.
- *Uniformisation rate* increases as μ increases, and PRISM's computation time increases linearly in the unif. rate.
- Efficiency of Path-IS does not degrade as μ increases.

Bigger n (More Redundancy)

	$\hat{\pi}$ $(\times 10^{-6})$	# runs	runtime
MC	5.0442 ± 2.9809	2 180 715	60 sec
BFB	4.6725 ± 0.8131	517 488	60 sec
Path-IS	4.6643 ± 0.0114	249 041	60 sec
	π $(\times 10^{-6})$	# states	run time
PRISM	4.6704	7 529 536	695 sec

Unreliability ($\hat{\pi}$) results when $n = 3$; $\mu = 1, \lambda = 1/6000$.

- When $n = 3$, PRISM takes much longer for computation.

High n (High Redundancy)

	$\hat{\pi}$ $(\times 10^{-14})$	# runs	run time
MC	0 ± 0	1 121 338	60 sec
BFB	0.0053 ± 0.0079	106 501	60 sec
Path-IS	9.9668 ± 0.2231	81 626	60 sec
	π	# states	run time
PRISM	*N.A.*	1 655 595 487	∞

Unreliability ($\hat{\pi}$) results when $n = 6$; $\mu = 1, \lambda = 1/6000$.

- When $n = 6$, PRISM is not able to produce an estimate (memory problems).

Why PRISM fails for high *n*

n	state space size (PRISM)	# non-zeros in transition rate matrix
2	421 875	5 737 500
3	7 529 536	111 329 568
4	66 430 125	1 027 452 600
5	382 657 176	6 087 727 800
6	1 655 595 487	26 853 394 932

Unavailability

- Long-run fraction of time spent in a *red* state.
- Known analytically for this class of models
  (but may be hard to compute due to roundoff errors).
- Denoted by v.

Low λ (rare failures)

	$\hat{v}$ $(\times 10^{-12})$	# runs	run time
MC	5.847 ± 11.460	386 538	$\approx$ 3 sec
BFB	3.532 ± 0.105	165 943	$\approx$ 3 sec
Path-IS	$\mathbf{3.521 \pm 0.036}$	**78 179**	$\approx$ 3 sec
	v $(\times 10^{-12})$	# states	run time
PRISM	3.500	421 875	$\approx$ 3 sec

Unavailability ($\hat{v}$) results when $\lambda = \frac{1}{6} \cdot 10^{-6}; \mu = 1, n = 2$.

- For low λ, PRISM is efficient.

High μ (fast repairs)

	$\hat{v}$ $(\times 10^{-12})$	# runs	run time
MC	0 ± 0	384 418	$\approx$ 3 sec
BFB	3.569 ± 0.106	146 603	$\approx$ 3 sec
Path-IS	$\mathbf{3.521 \pm 0.038}$	**69 341**	$\approx$ 3 sec
	v $(\times 10^{-12})$	# states	run time
PRISM	3.500	421 875	$\approx$ 3 sec

Unavailability ($\hat{v}$) results when $\mu = 1,000$; $\lambda = 1/6000, n = 2$.

- For high μ, PRISM is still efficient for computing steady-state probabilities.

High n (More Redundancy)

	$\hat{v}$ $(\times 10^{-16})$	# runs	run time
MC	0 ± 0	6 708 624	60 sec
BFB	0.148 ± 0.225	803 752	60 sec
Path-IS	$\mathbf{1.173 \pm 0.016}$	**205 654**	60 sec
	v	# states	run time
PRISM	N.A.	1 655 595 487	∞

Unavailability ($\hat{v}$) results when $n = 6$; $\mu = 1, \lambda = 1/6000$.

- For high n, PRISM fails.

Outline

Conclusions

- Path-based importance sampling approach introduced for wide range of models.
- Unlike other importance sampling methods, good performance for fast repairs.
- Relative performance of method depends on situation.

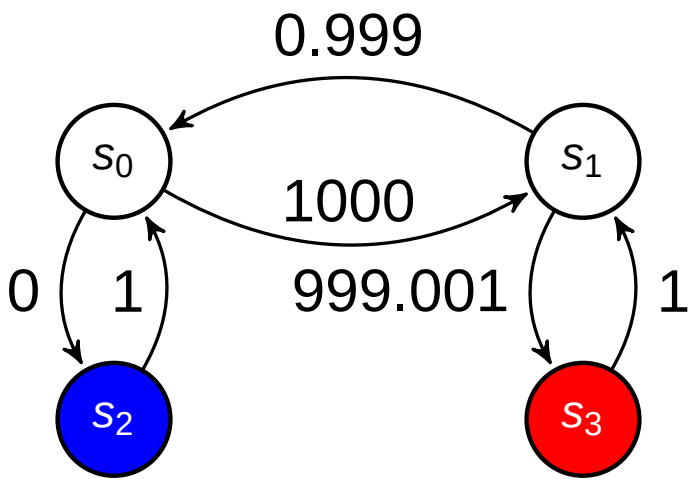
Better tool	low λ	high μ	high n
Unrel.	PRISM	Path-IS	Path-IS
Unav.	PRISM	PRISM	Path-IS

Future Work

- Further generalisation to models with interdependent components.
- Generalising this approach to Markov Reward Models.

Thank you for your attendance

Zero Variance Example



Zero variance measure for a toy example.

$$\mathbb{P}_{s_0}(\Phi) = \frac{\frac{1}{1000} \cdot \frac{1}{1000}}{1 \cdot \frac{999.001}{1000}} = \frac{1}{999001}$$

$$\mathbb{P}_{s_1}(\Phi) = \frac{\frac{1}{1000}}{\frac{999.001}{1000}} = \frac{1}{999.001}$$

Derivation IS

$$\mathbb{P}(\Phi) = \mathbb{E}_{\mathbb{P}}(\mathbf{1}_{\Phi}) = \sum_{\omega} \mathbf{1}_{\Phi}(\omega)\mathbb{P}(\omega) = \sum_{\omega} \mathbf{1}_{\Phi}(\omega)\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\mathbb{Q}(\omega) =$$

$$\mathbb{E}_{\mathbb{Q}}(\frac{\mathbb{P}}{\mathbb{Q}}\mathbf{1}_{\Phi})$$

Wald's Sequential test (1)

Hypotheses $H_0 : \mathbb{P}(\Phi) = p_0, H_1 : \mathbb{P}(\Phi) = p_1$.

Monte Carlo estimate $$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_\Phi(\omega_i) \triangleq \frac{S}{N}$$

Test statistic $$T = \frac{f_{H_0}(\mathbf{1}_\Phi(\omega_1), \ldots, \mathbf{1}_\Phi(\omega_N))}{f_{H_1}(\mathbf{1}_\Phi(\omega_1), \ldots, \mathbf{1}_\Phi(\omega_N))} = \frac{p_0^S(1-p_0)^{N-S}}{p_1^S(1-p_1)^{N-S}}.$$

Sample until T is larger than some threshold (accept H_0) or smaller than some other threshold (accept H_1).

Wald's Sequential test (2)

Hypotheses $H_0 : \mathbb{P}(\Phi) = p_0, H_1 : \mathbb{P}(\Phi) = p_1$.

Importance sampling estimate $\hat{p} = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbb{P}(\omega_i)}{\mathbb{P}^*(\omega_i)} \mathbf{1}_\Phi(\omega_i)$

Test statistic $T = \frac{f^*_{H_0}(\frac{\mathbb{P}(\omega_1)}{\mathbb{P}^*(\omega_1)} \mathbf{1}_\Phi(\omega_1), \ldots, \frac{\mathbb{P}(\omega_N)}{\mathbb{P}^*(\omega_N)} \mathbf{1}_\Phi(\omega_N))}{f^*_{H_1}(\frac{\mathbb{P}(\omega_1)}{\mathbb{P}^*(\omega_1)} \mathbf{1}_\Phi(\omega_1), \ldots, \frac{\mathbb{P}(\omega_N)}{\mathbb{P}^*(\omega_N)} \mathbf{1}_\Phi(\omega_N))}$.

Problem: f^*_H not trivial for fixed H.