Imperial College
London

Fluid analysis of Markov Models

Jeremy Bradley, Richard Hayden, Anton Stefanek

Imperial College London

Tutorial, SFM:DS 2013

20 June 2013

You can download this presentation now from:

http://www.doc.ic.ac.uk/~jb/pub/sfm-ds2013.pdf

or

http://tinyurl.com/sfm-ds-fluid

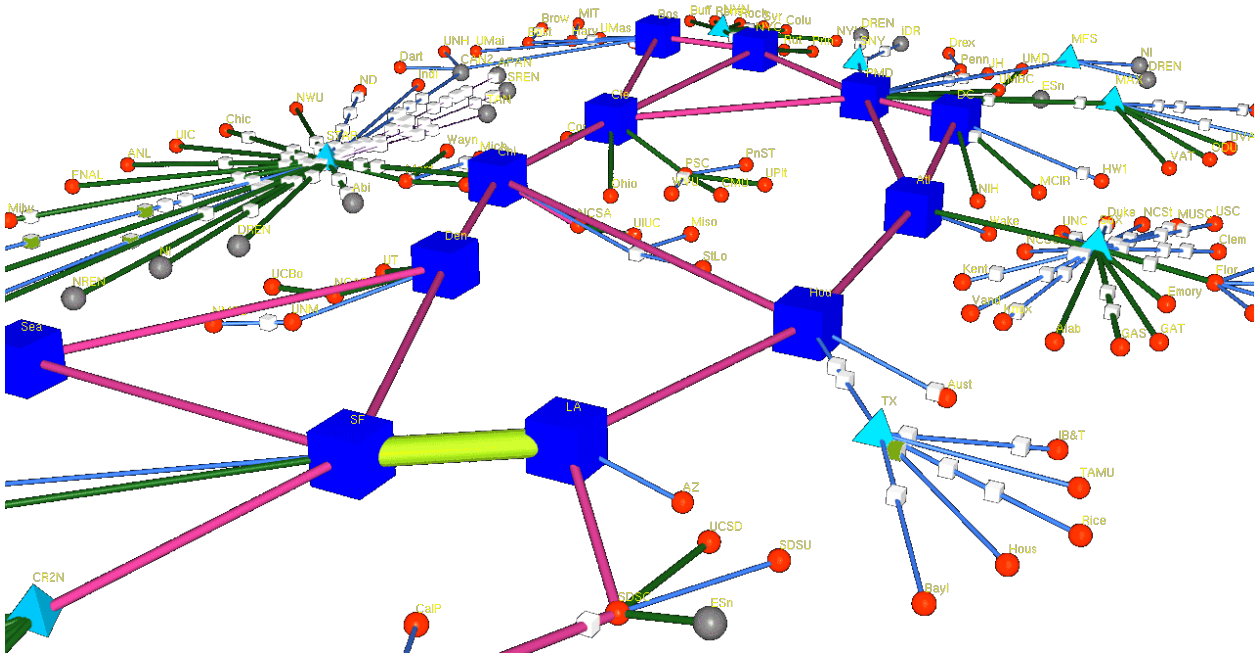
How can we...

scale

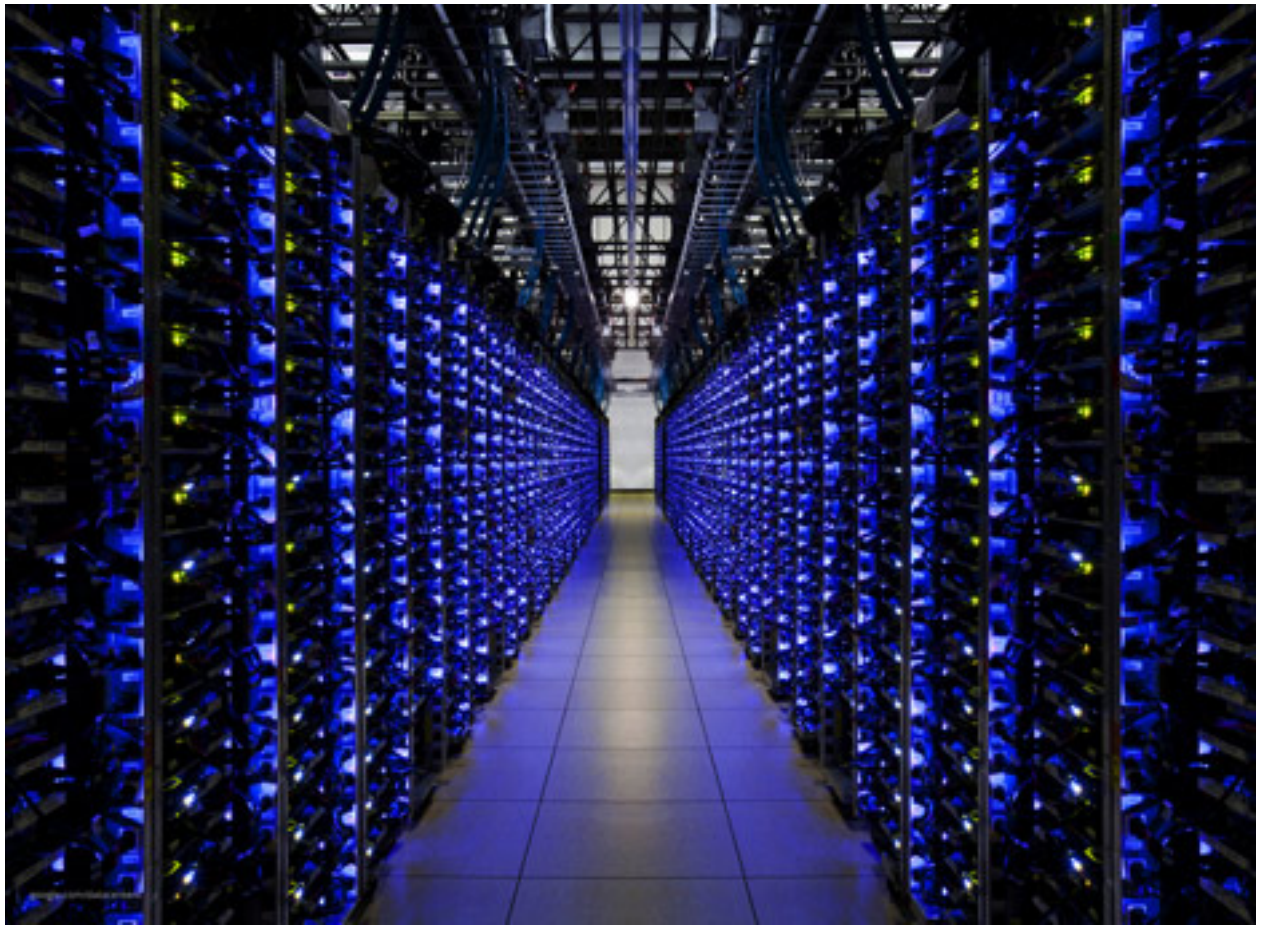
resource

provision

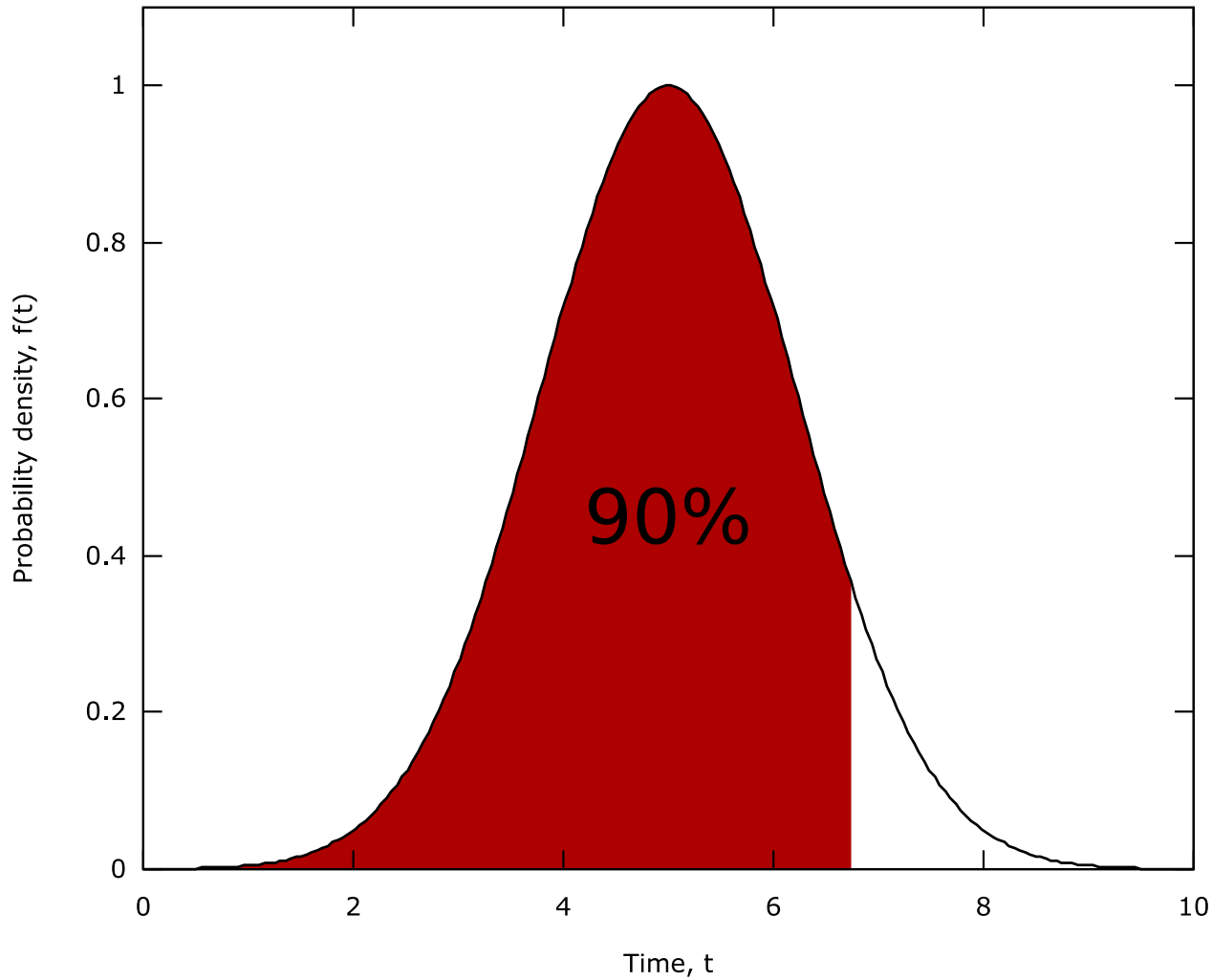
design


Brow
MIT
Bos
Buff
Colu
DREN
IDR
UNH
UMai
UMas
Dart
Drex
Penn
UMD
MFS
NI
DREN
ESn
ND
NWU
SREN
Chic
UIC
ANL
FNAL
Wayn
Chi
Ohio
PSC
PnST
UPit
HW1
MCIR
UNIH
Wake
UNC
Duke
NCSU
MUSC
USC
Clem
Kent
Emory
Vand
GAT
GAS
Atla
Hou
Aust
TX
IB&T
TAMU
Rice
Hous
Bayl
SF
LA
AZ
UCSD
SDSU
ESn
CalP
CR2N
Sea
UCSA
UIUC
Miso
SILo
UCBo
UT
NREN

or



to meet


Probability density, f(t)
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
Time, t
90%

while minimising

?

We want to be able to engineer complex systems

We want to be able to engineer complex systems

We want to be able to reason about performance

We want to be able to engineer complex systems

We want to be able to reason about performance

We want to be able to optimise key cost functions

We want to be able to engineer complex systems

We want to be able to reason about performance

We want to be able to optimise key cost functions

...at the same time

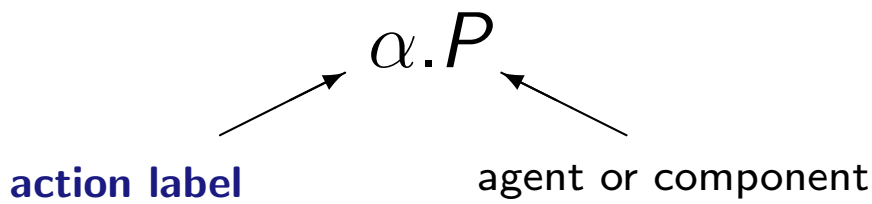
Process modelling with Stochastic systems

Process Algebra

A process algebra model consists of agents which engage in actions.

Process Algebra

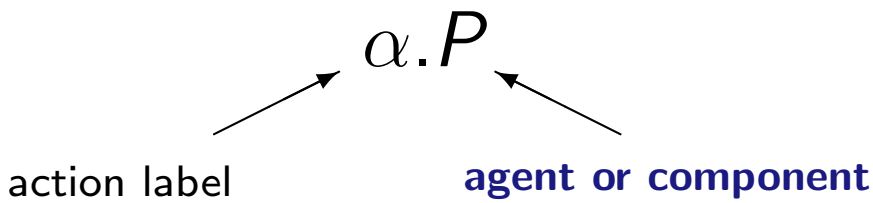
A process algebra model consists of agents which engage in actions.



Based on a slide by Jane Hillston

Process Algebra

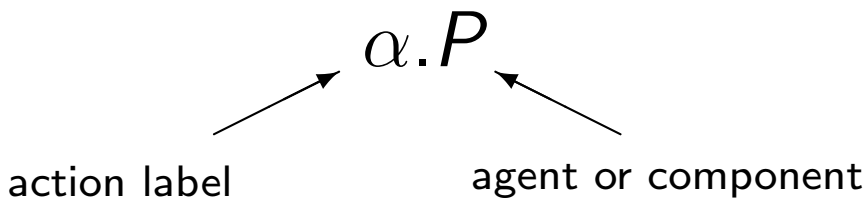
A process algebra model consists of agents which engage in actions.



Based on a slide by Jane Hillston

Process Algebra

A process algebra model consists of agents which engage in actions.

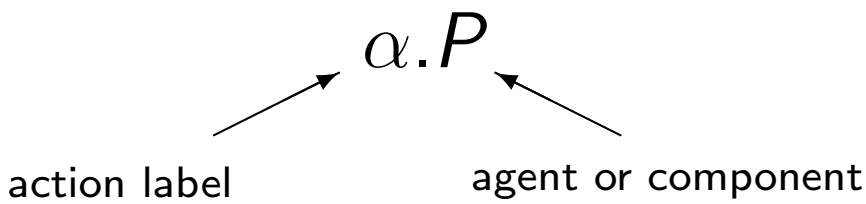


The semantics of the language define an underlying state space by way of a labelled transition system.

Based on a slide by Jane Hillston

Process Algebra

A process algebra model consists of agents which engage in actions.



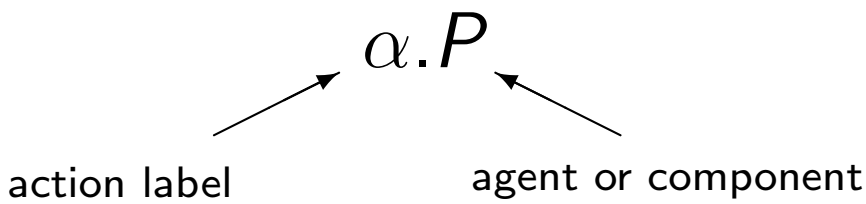
The semantics of the language define an underlying state space by way of a labelled transition system.

A process algebra model

Based on a slide by Jane Hillston

Process Algebra

A process algebra model consists of agents which engage in actions.



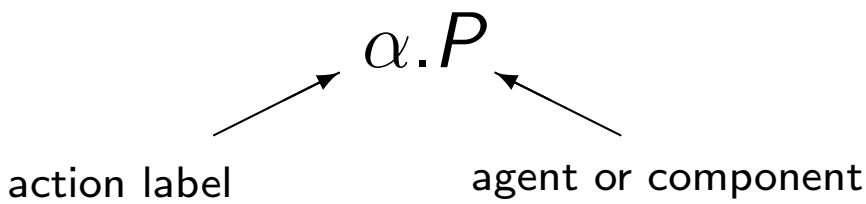
The semantics of the language define an underlying state space by way of a labelled transition system.

A process algebra model $\xrightarrow{\text{semantic rules}}$

Based on a slide by Jane Hillston

Process Algebra

A process algebra model consists of agents which engage in actions.



The semantics of the language define an underlying state space by way of a labelled transition system.

A process algebra model $\xrightarrow{\text{semantic rules}}$ Labelled transition system

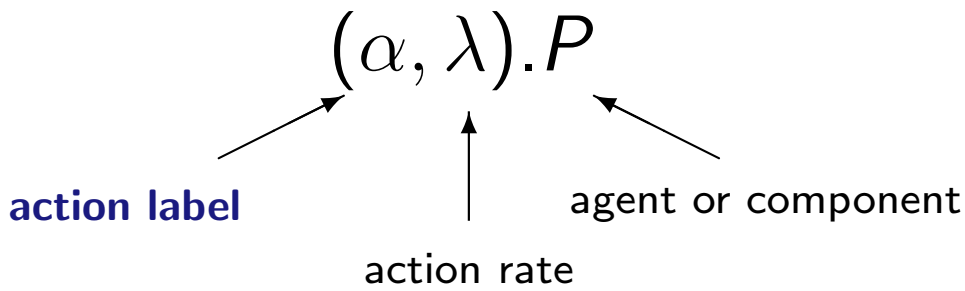
Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.

Stochastic Process Algebra

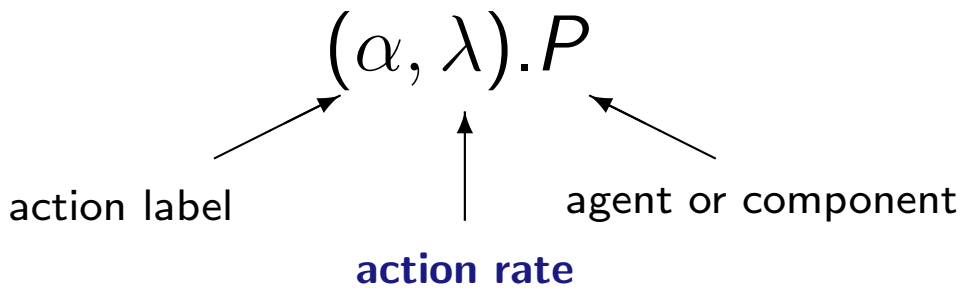
A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



Based on a slide by Jane Hillston

Stochastic Process Algebra

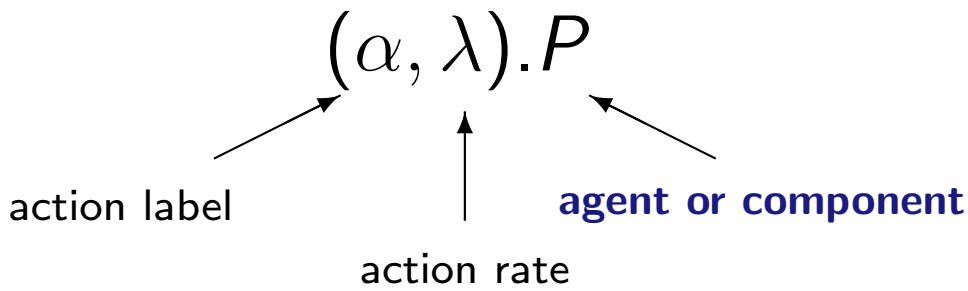
A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



Based on a slide by Jane Hillston

Stochastic Process Algebra

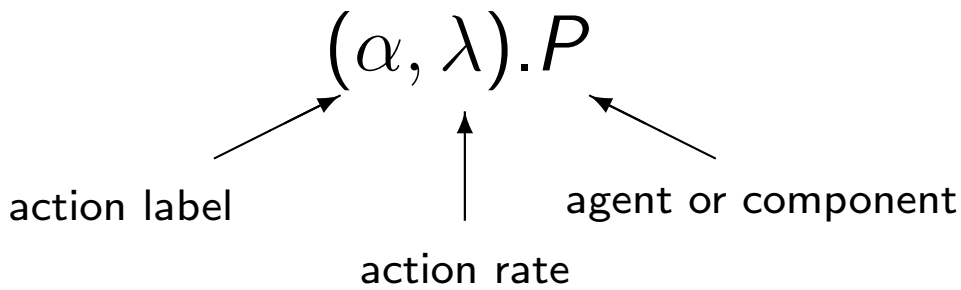
A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.

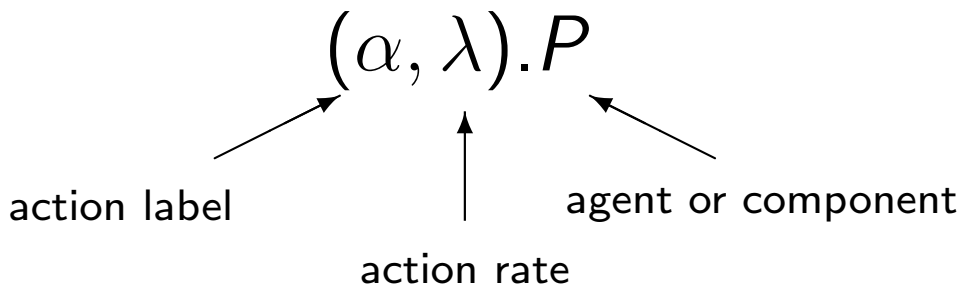


where the action duration is exponentially distributed with rate λ.

Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



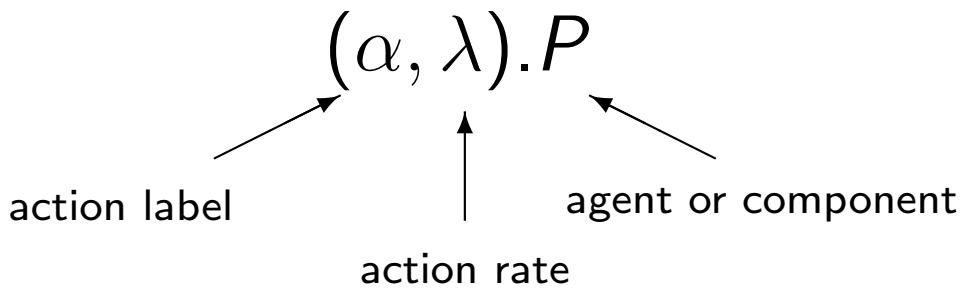
where the action duration is exponentially distributed with rate λ.

The semantics of the language define an underlying state space and also a performance model in terms of a CTMC.

Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



where the action duration is exponentially distributed with rate λ.

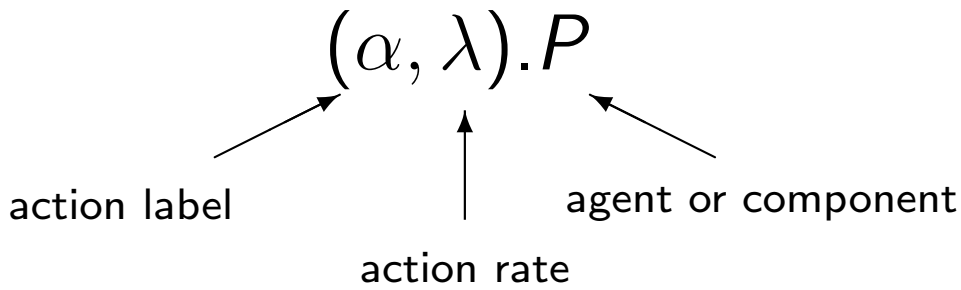
The semantics of the language define an underlying state space and also a performance model in terms of a CTMC.

SPA model

Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



where the action duration is exponentially distributed with rate λ.

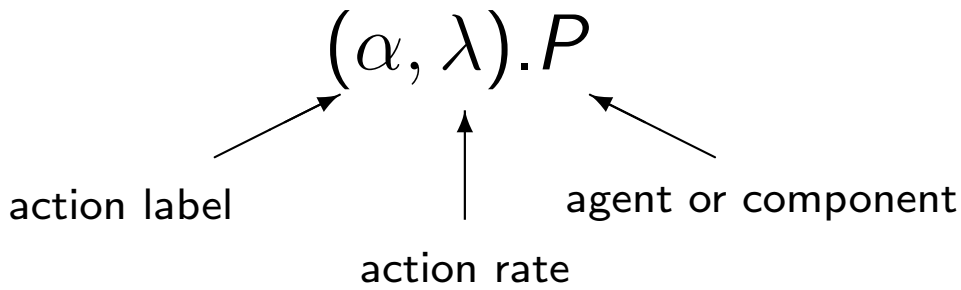
The semantics of the language define an underlying state space and also a performance model in terms of a CTMC.

SPA model $\xrightarrow{\text{semantics}}$

Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



where the action duration is exponentially distributed with rate λ.

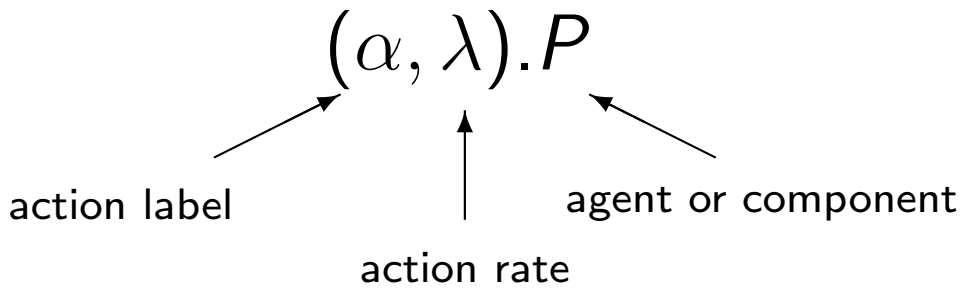
The semantics of the language define an underlying state space and also a performance model in terms of a CTMC.

SPA model $\xrightarrow{\text{semantics}}$ LTS

Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



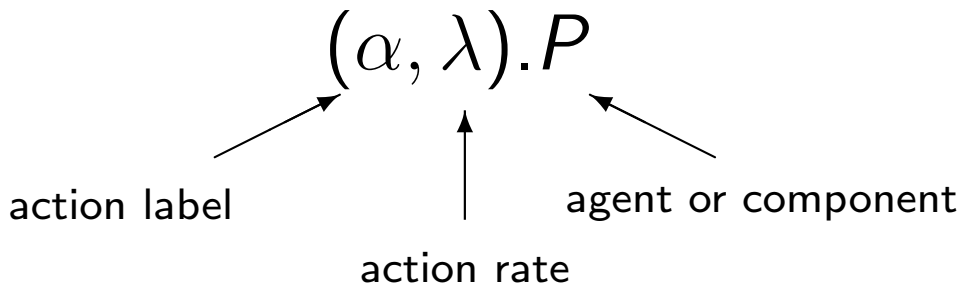
where the action duration is exponentially distributed with rate λ.

The semantics of the language define an underlying state space and also a performance model in terms of a CTMC.

Based on a slide by Jane Hillston

Stochastic Process Algebra

A **stochastic process algebra model** also consists of agents which engage in actions, but with the actions having a random duration associated with them.



where the action duration is exponentially distributed with rate λ.

The semantics of the language define an underlying state space and also a performance model in terms of a CTMC.

Based on a slide by Jane Hillston

PEPA: Stochastic process algebra

- Many SPAs exist and capture performance and behavioural features in different ways. e.g. iGSMPA[1], IMC[2], sFSP[3], EMPA[4], TIPP[5]
- PEPA[6] is useful because:
  - it is a formal, algebraic description of a system
  - it is compositional
  - it is parsimonious (succinct)
  - it is easy to learn!
  - it is used in research and in industry

[1] Mario Bravetti and Roberto Gorrieri. "Interactive Generalized Semi-Markov Processes". In: *Process Algebra and Performance Modelling Workshop*. Ed. by Jane Hillston and Manuel Silva. Centro Politécnico Superior de la Universidad de Zaragoza. Prensas Universitarias de Zaragoza, 1999, pp. 83–98.

[2] Holger Hermanns. "Interactive Markov Chains". PhD thesis. Universität Erlangen–Nürnberg, 1998.

[3] Thomas Ayles et al. "Adding Performance Evaluation to the LTSA Tool". In: *Proceedings of 13th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools*. 2003.

[4] Marco Bernardo and Roberto Gorrieri. "Extended Markovian Process Algebra". In: *CONCUR'96, Proceedings of the 7th International Conference on Concurrency Theory*. Ed. by Ugo Montanari and Vladimiro Sassone. Vol. 1119. Lecture Notes in Computer Science. Springer-Verlag, 1996, pp. 315–330.

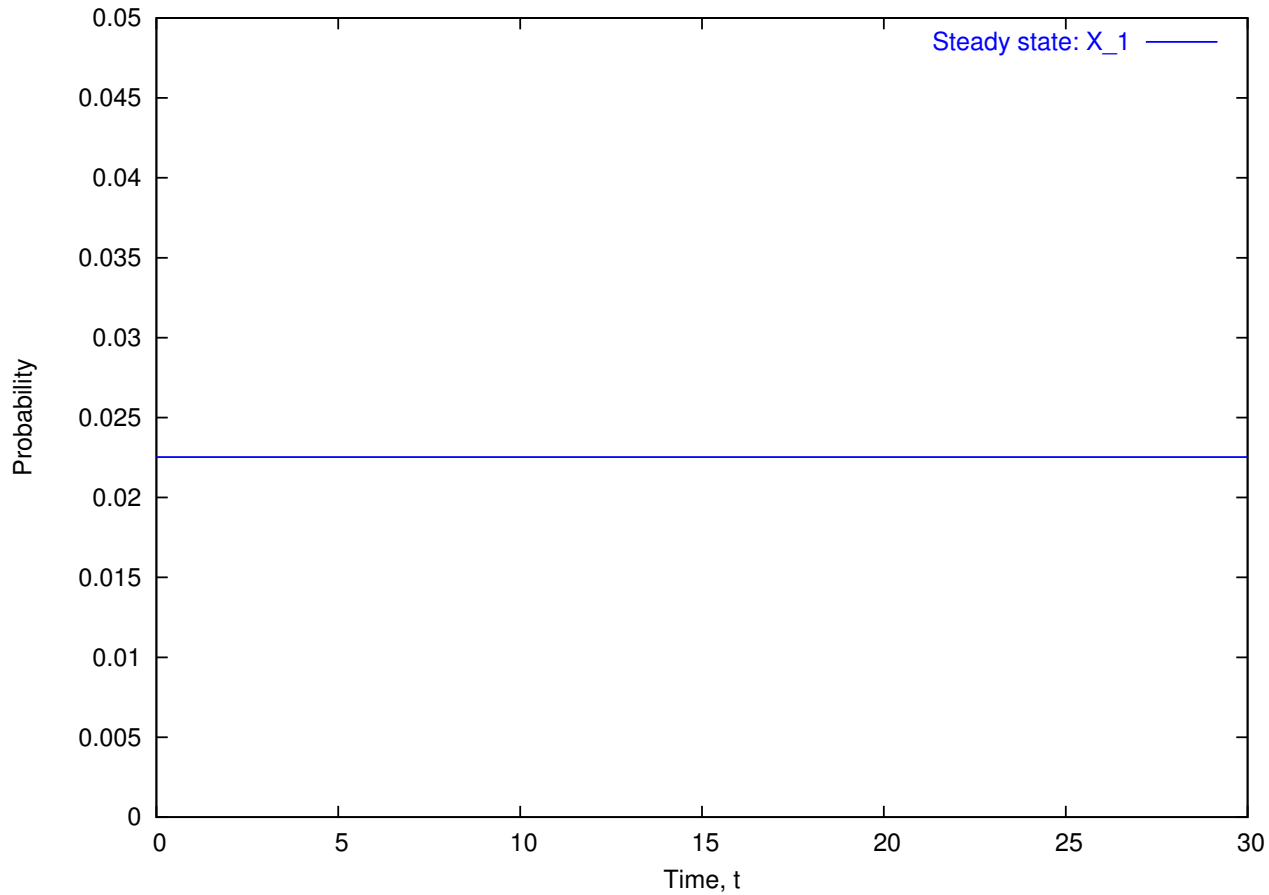
[5] Norbert Götz, Ulrich Herzog, and Michael Rettelbach. "TIPP—A Stochastic Process Algebra". In: *Process Algebra and Performance Modelling*. Ed. by Jane Hillston and Faron Moller. CSR Technical Report. Department of Computer Science, University of Edinburgh, 1993, pp. 31–36.

[6] Jane Hillston. *A Compositional Approach to Performance Modelling*. Vol. 12. Distinguished Dissertations in Computer Science. Cambridge University Press, 1996.

What can you do with PEPA?

It allows you to answer key performance questions

Steady state analysis

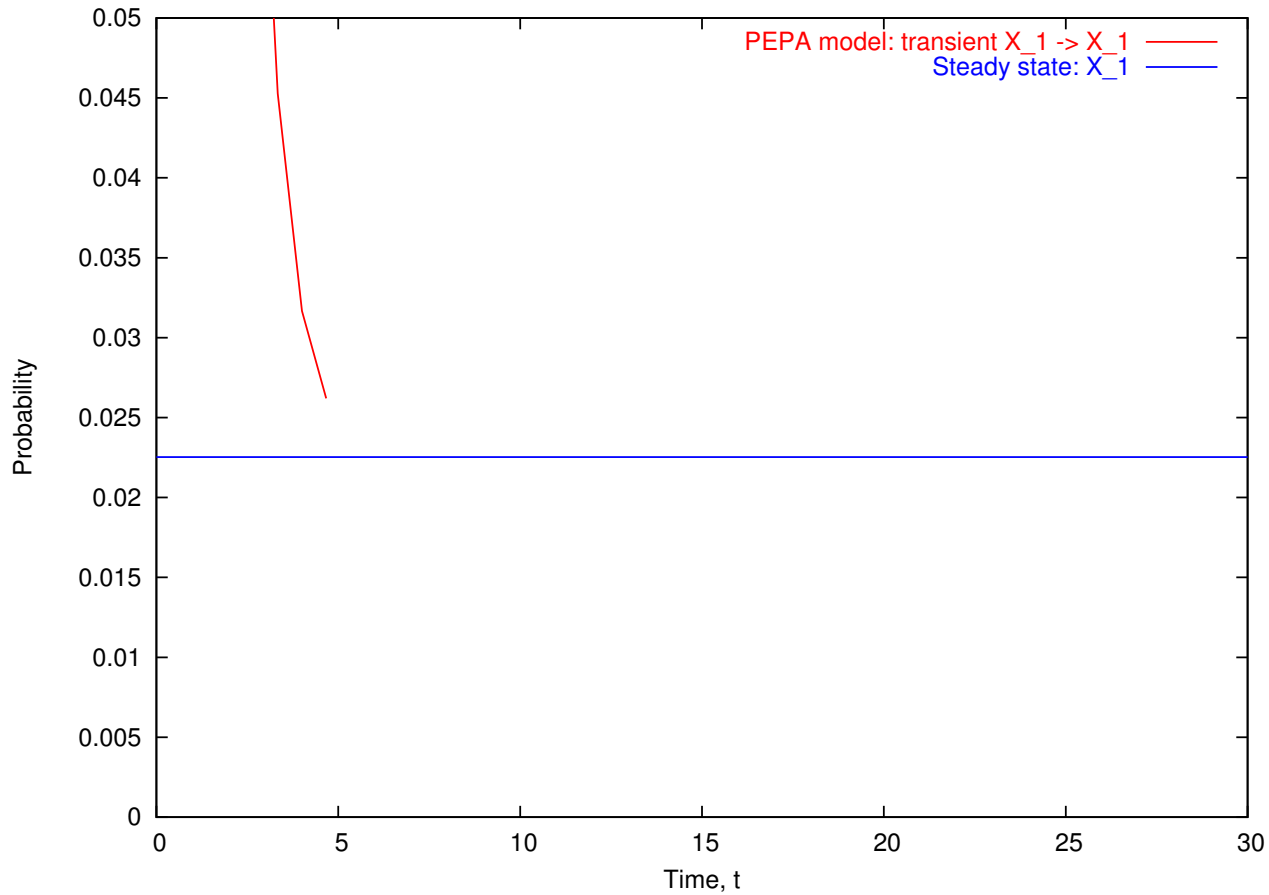


What is the long-run average behaviour of my system?

What can you do with PEPA?

It allows you to answer key performance questions

Transient analysis

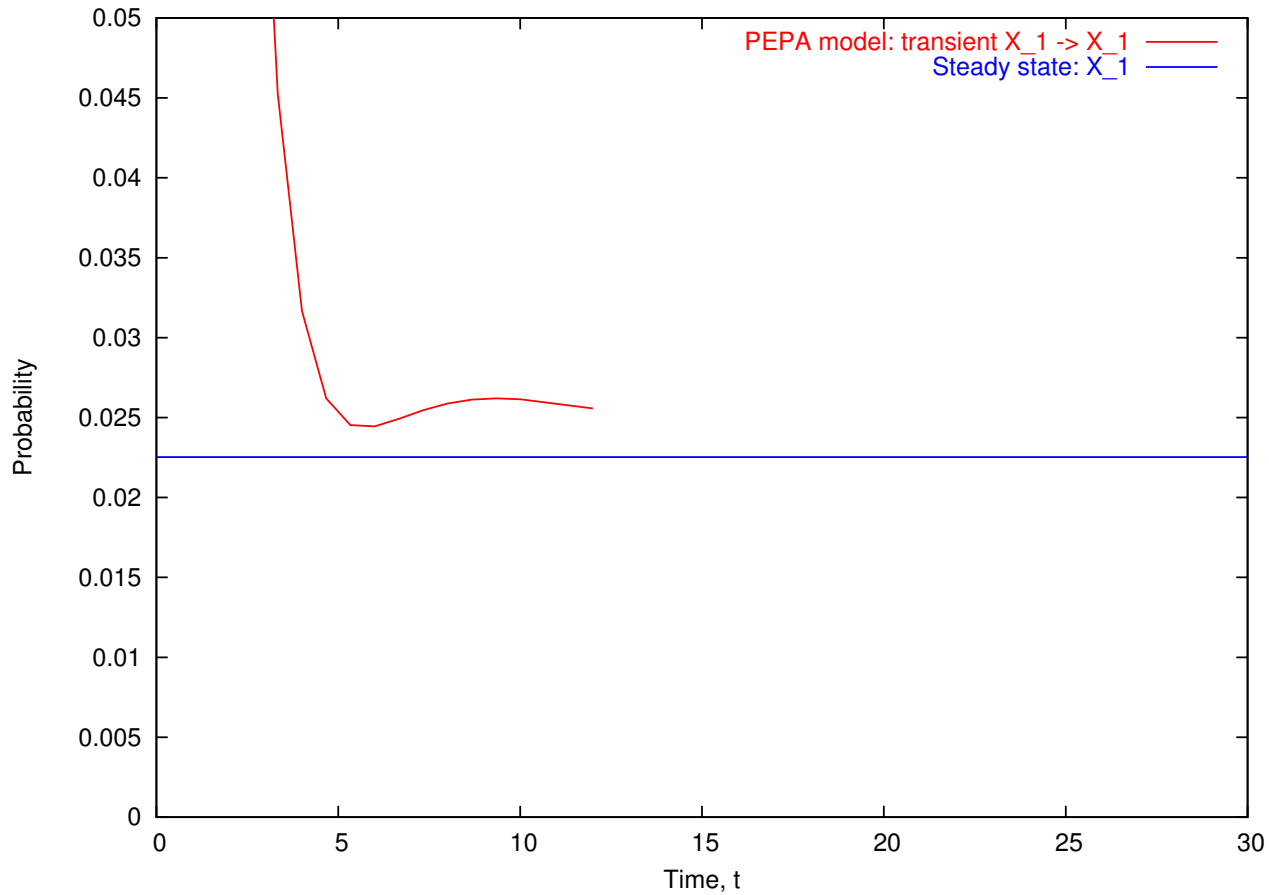


What is the behaviour of my system at time, t?

What can you do with PEPA?

It allows you to answer key performance questions

Transient analysis

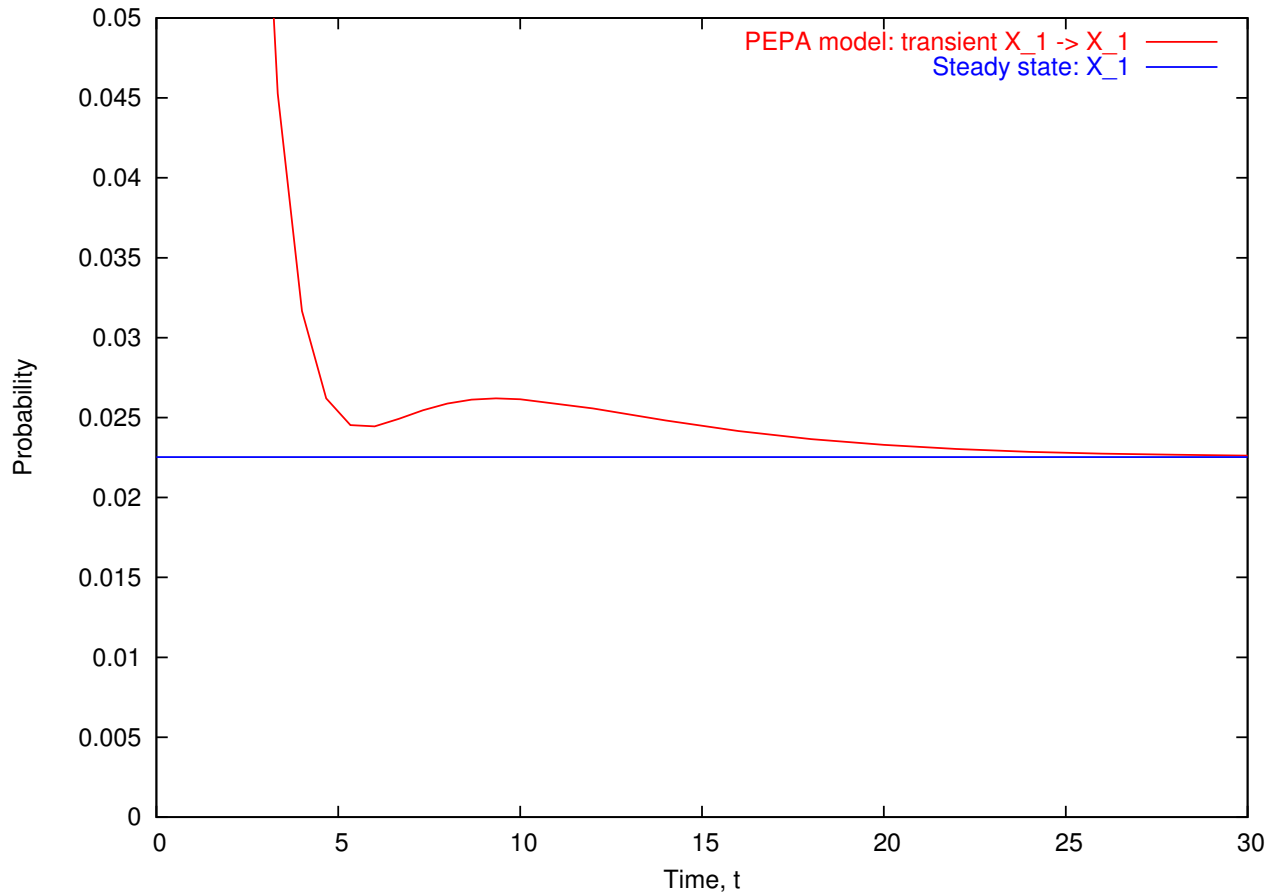


What is the behaviour of my system at time, t?

What can you do with PEPA?

It allows you to answer key performance questions

Transient analysis

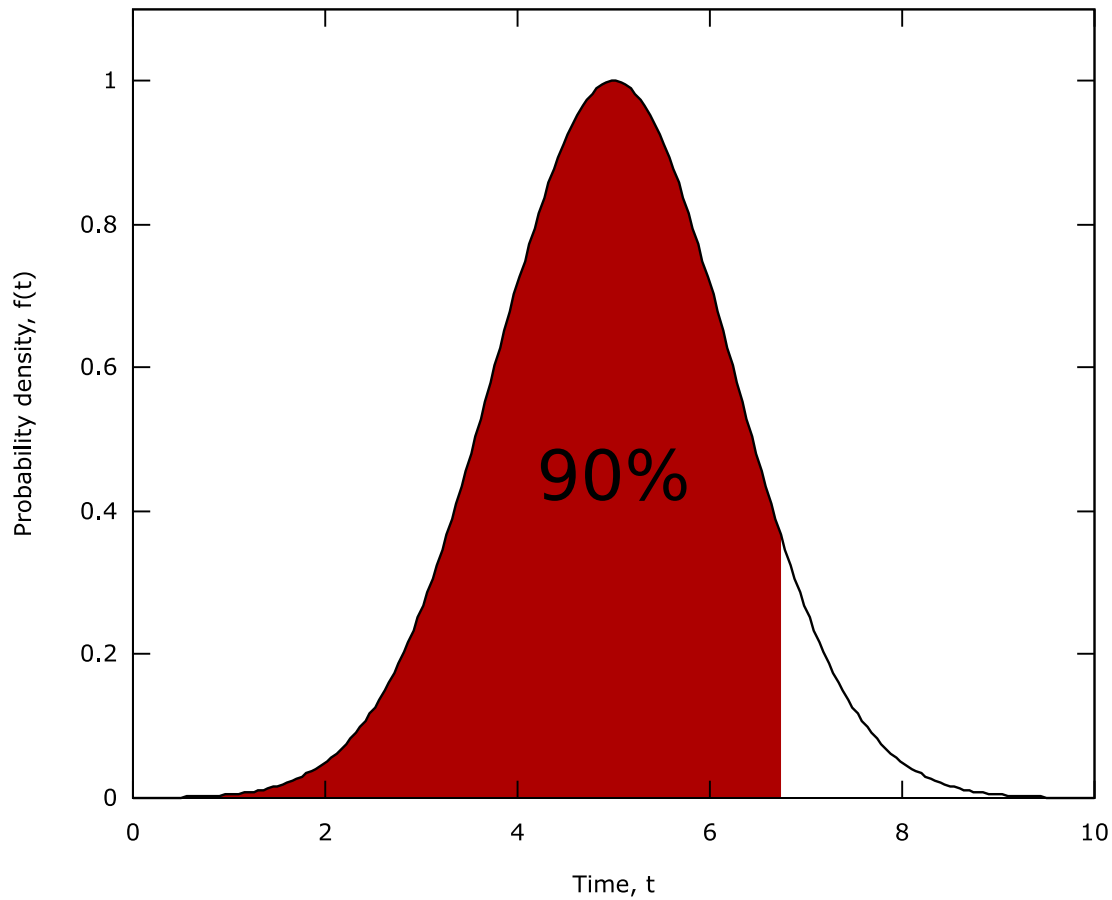


What is the behaviour of my system at time, t?

What can you do with PEPA?

It allows you to answer key performance questions

Passage time analysis



How long does it take my system to complete a key transaction?

Tool Support

- PEPA has several methods of execution and analysis, through comprehensive tool support:
  - PEPA Eclipse plugin: Edinburgh[7]
  - Möbius: Urbana-Champaign, Illinois[8]
  - PRISM: Birmingham[9]
  - ipc: Imperial College London[10]
  - gpa: Imperial College London[11]

[7] Mirco Tribastone. "The PEPA Plug-in Project". In: *QEST'07, Proceedings of the 4th Int. Conference on the Quantitative Evaluation of Systems*. IEEE Computer Society, 2007, pp. 53–54.

[8] Graham Clark et al. "The Möbius Modeling Tool". In: *Proceedings the 9th International Workshop on Petri Nets and Performance Models*. Ed. by B Haverkort and R German. IEEE Computer Society Press, 2001, pp. 241–250.

[9] Marta Kwiatkowska, Gethin Norman, and David Parker. "PRISM: Probabilistic Symbolic Model Checker". In: *TOOLS'02, Proceedings of the 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Ed. by A J Field et al. Vol. 2324. Lecture Notes in Computer Science. London: Springer-Verlag, 2002, pp. 200–204.

[10] Jeremy T Bradley and William J Knottenbelt. "The ipc/HYDRA Tool Chain for the Analysis of PEPA Models". In: *QEST'04, Proceedings of the 1st IEEE Conference on the Quantitative Evaluation of Systems*. Ed. by Boudewijn Haverkort et al. University of Twente, Enschede: IEEE Computer Society, 2004, pp. 334–335.

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "A new tool for the performance analysis of massively parallel computer systems". In: *Eighth Workshop on Quantitative Aspects of Programming Languages (QAPL 2010), March 27-28, 2010, Paphos, Cyprus*. Electronic Proceedings in Theoretical Computer Science. Mar. 2010. URL: `http://pubs.doc.ic.ac.uk/pepa-ode-moments-tool/`.

PEPA Syntax

Syntax:

$$P \quad ::= \quad (a, \lambda).P \;\Big|\; P + P \;\Big|\; P \underset{L}{\bowtie} P \;\Big|\; P/L \;\Big|\; A \stackrel{def}{=} P$$

- Action prefix: $(a, \lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: $A \stackrel{def}{=} P$

PEPA Syntax

Syntax:

$$P \quad ::= \quad (a, \lambda).P \;\Big|\; P + P \;\Big|\; P \underset{L}{\bowtie} P \;\Big|\; P/L \;\Big|\; A \stackrel{def}{=} P$$

- Action prefix: $(a, \lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: $A \stackrel{def}{=} P$

PEPA Syntax

Syntax:

$$P \quad ::= \quad (a, \lambda).P \;\Big|\; P + P \;\Big|\; P \underset{L}{\bowtie} P \;\Big|\; P/L \;\Big|\; A \stackrel{def}{=} P$$

- Action prefix: $(a, \lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: $A \stackrel{def}{=} P$

PEPA Syntax

Syntax:

$$P \quad ::= \quad (a, \lambda).P \;\Big|\; P + P \;\Big|\; P \underset{L}{\bowtie} P \;\Big|\; P/L \;\Big|\; A \stackrel{def}{=} P$$

- Action prefix: $(a, \lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: $A \stackrel{def}{=} P$

PEPA Syntax

Syntax:

$$P \quad ::= \quad (a, \lambda).P \;\Big|\; P + P \;\Big|\; P \underset{L}{\bowtie} P \;\Big|\; P/L \;\Big|\; A \stackrel{def}{=} P$$

- Action prefix: $(a, \lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: $A \stackrel{def}{=} P$

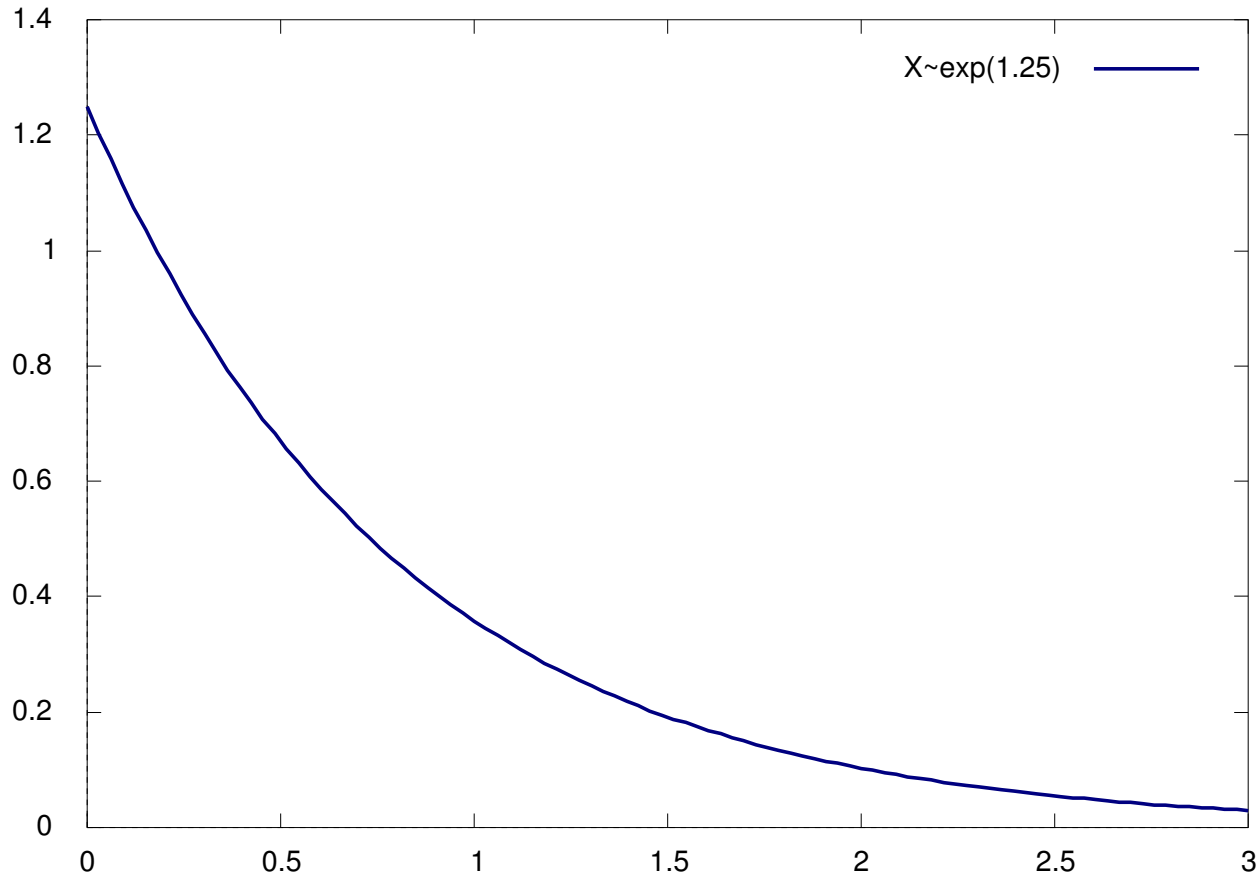
PEPA Syntax

Syntax:

$$P \quad ::= \quad (a, \lambda).P \;\Big|\; P + P \;\Big|\; P \underset{L}{\bowtie} P \;\Big|\; P/L \;\Big|\; A \stackrel{def}{=} P$$

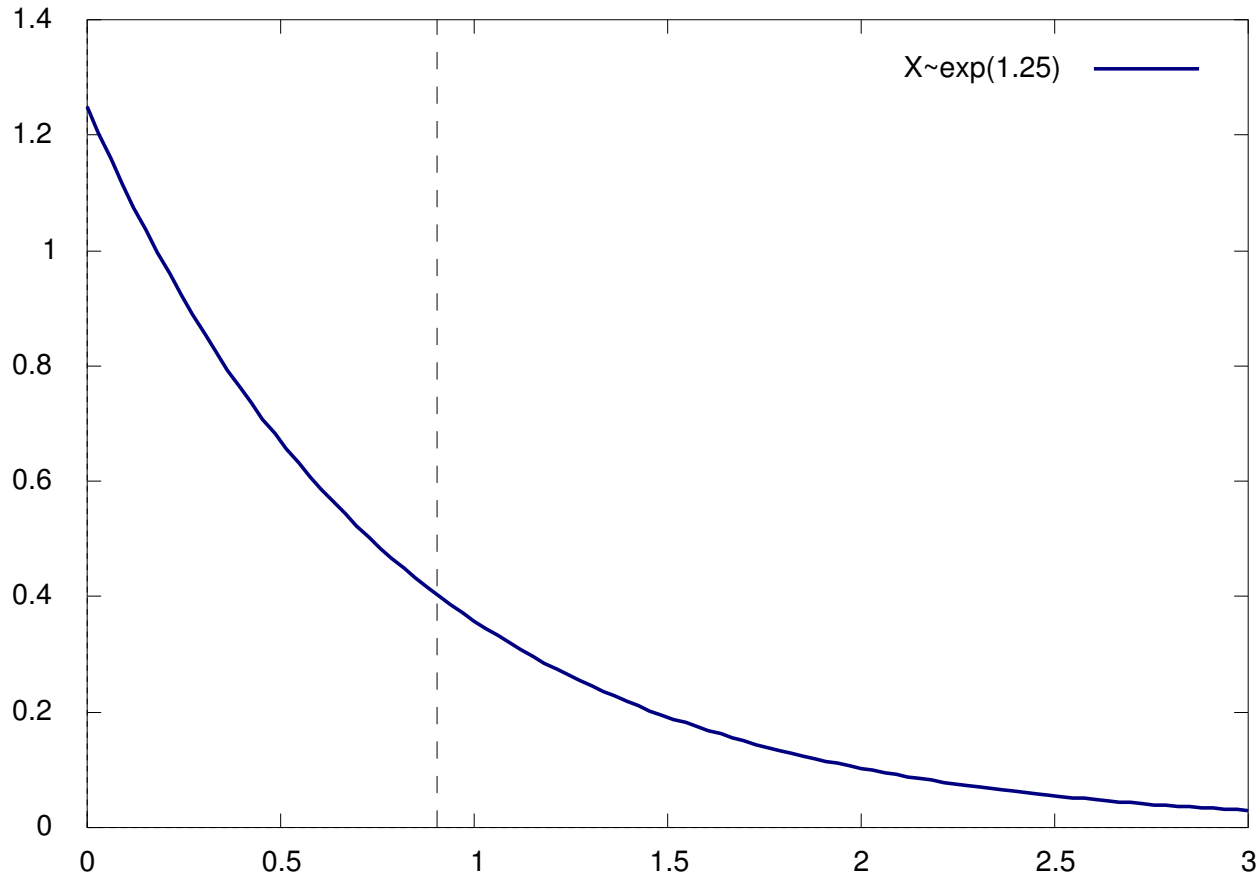
- Action prefix: $(a, \lambda).P$
- Competitive choice: $P_1 + P_2$
- Cooperation: $P_1 \underset{L}{\bowtie} P_2$
- Action hiding: P/L
- Constant label: $A \stackrel{def}{=} P$

Why exponential?



Why exponential?

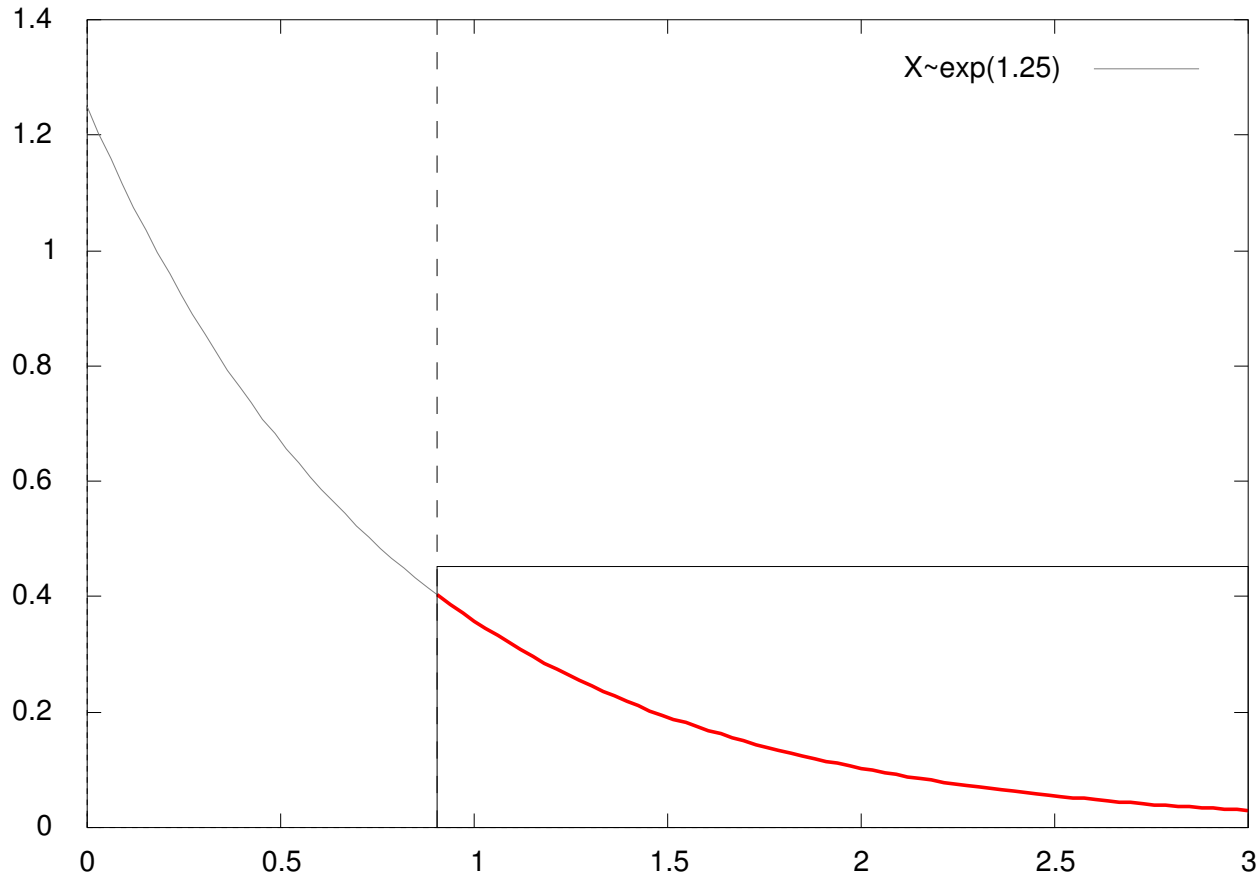
Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

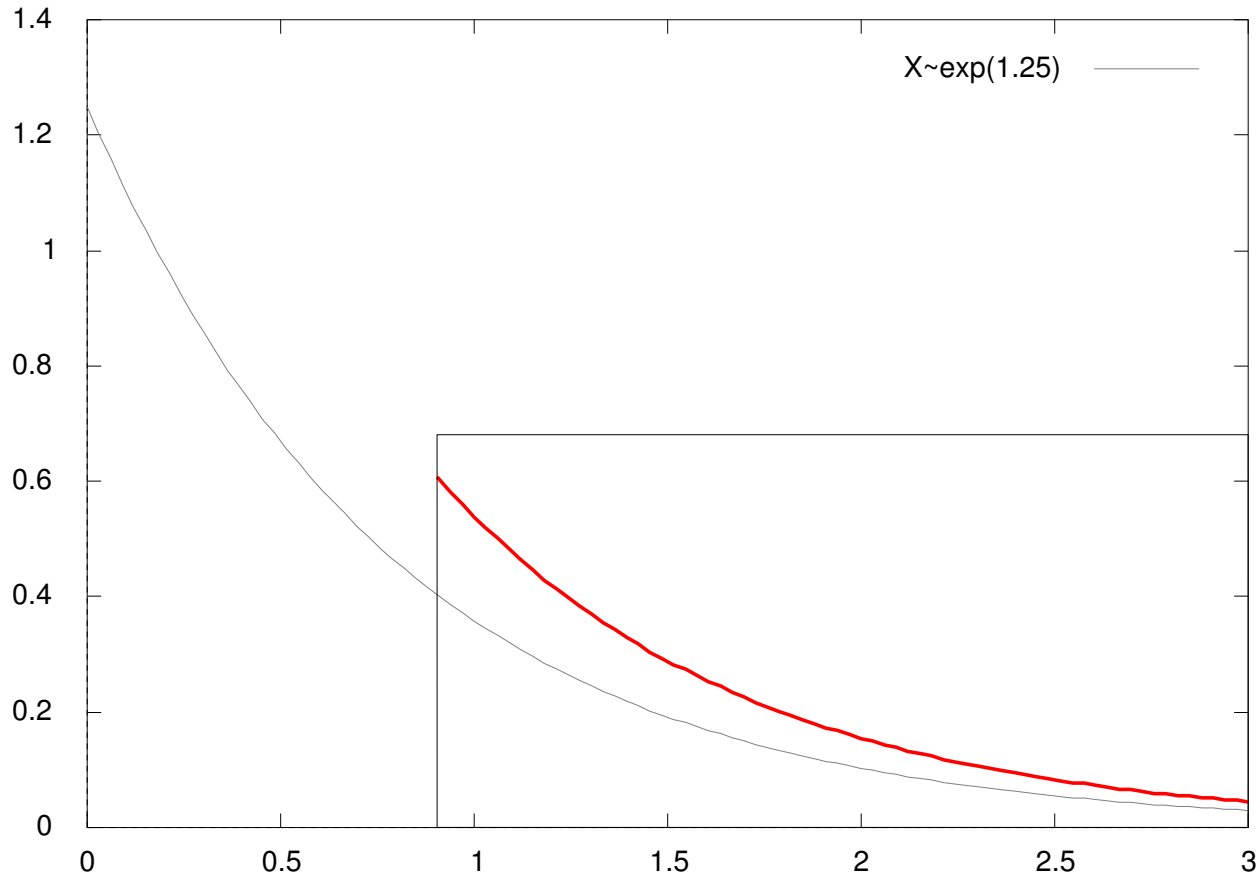
Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

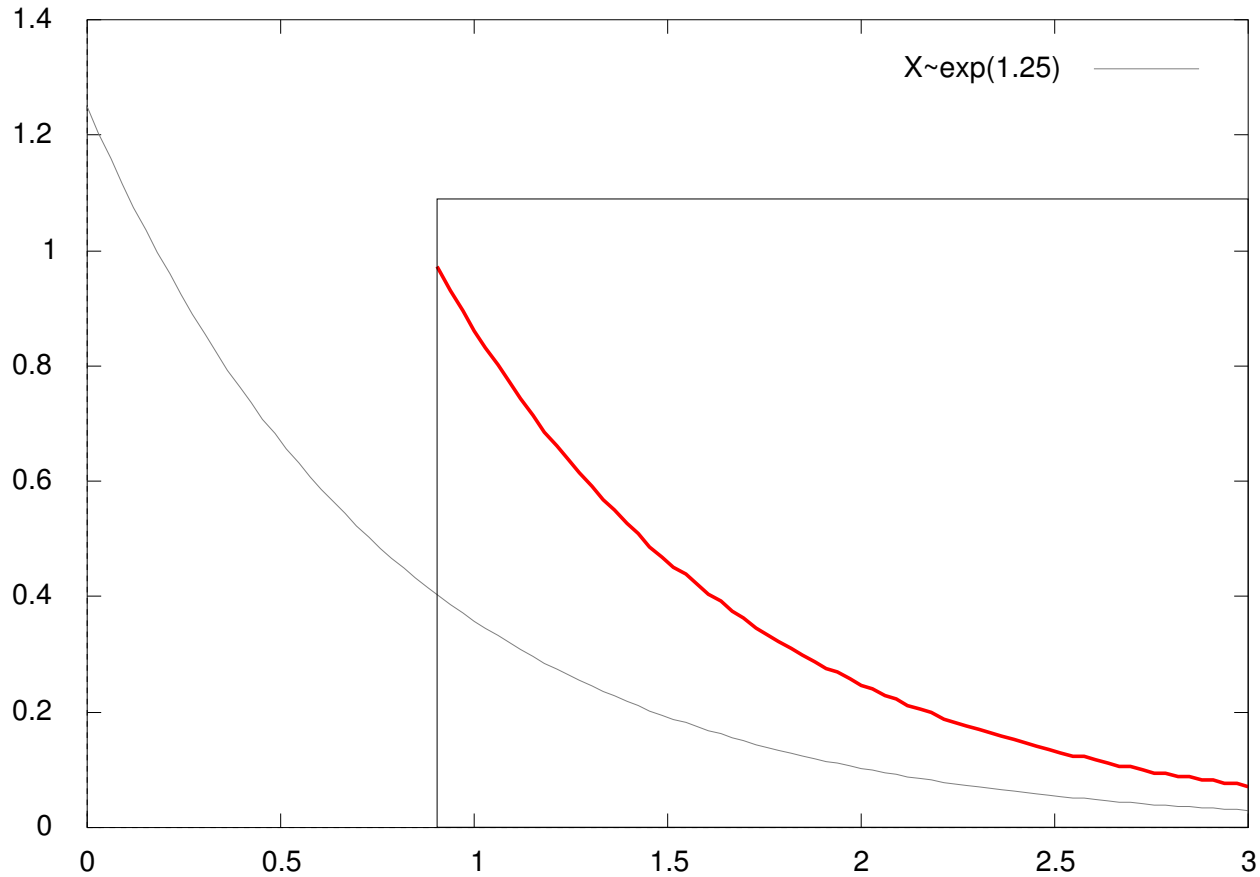
Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

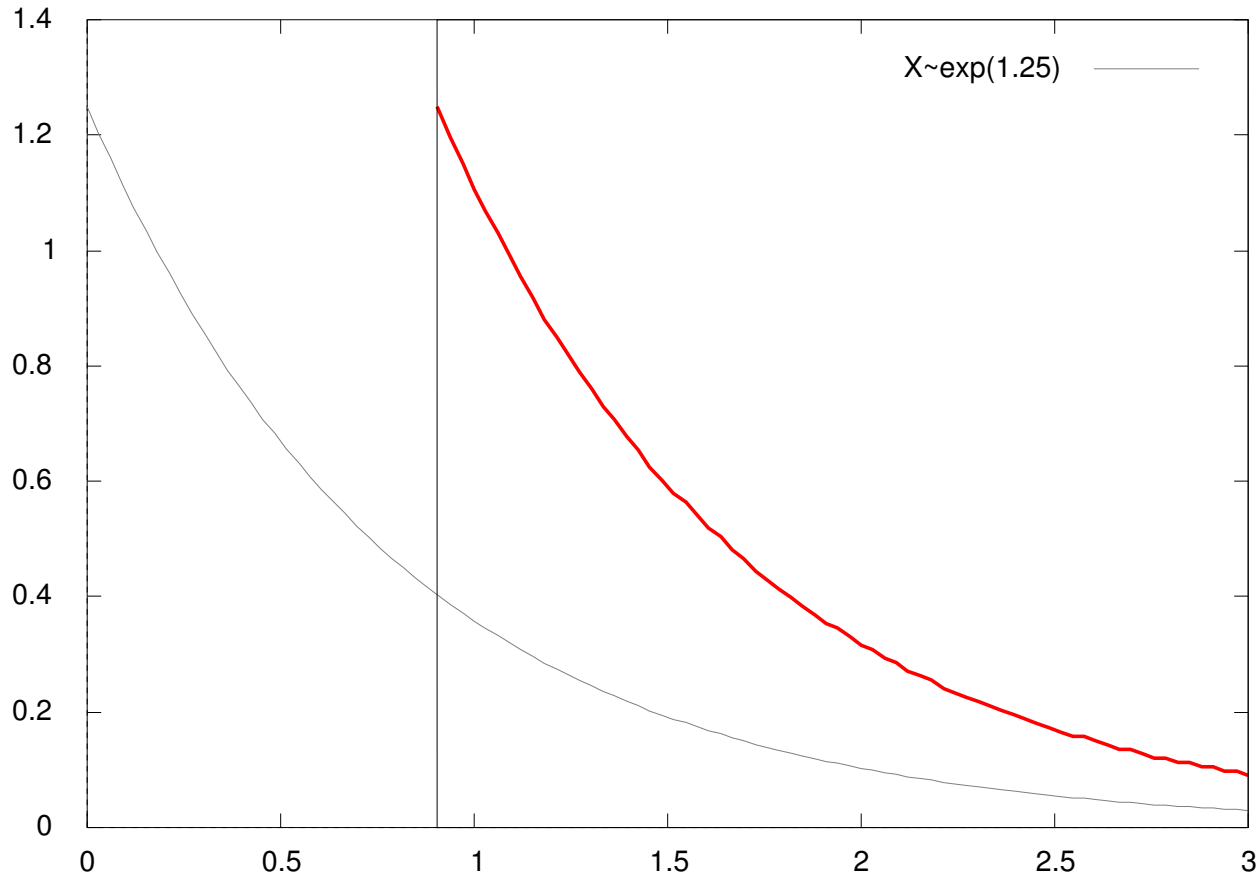
Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

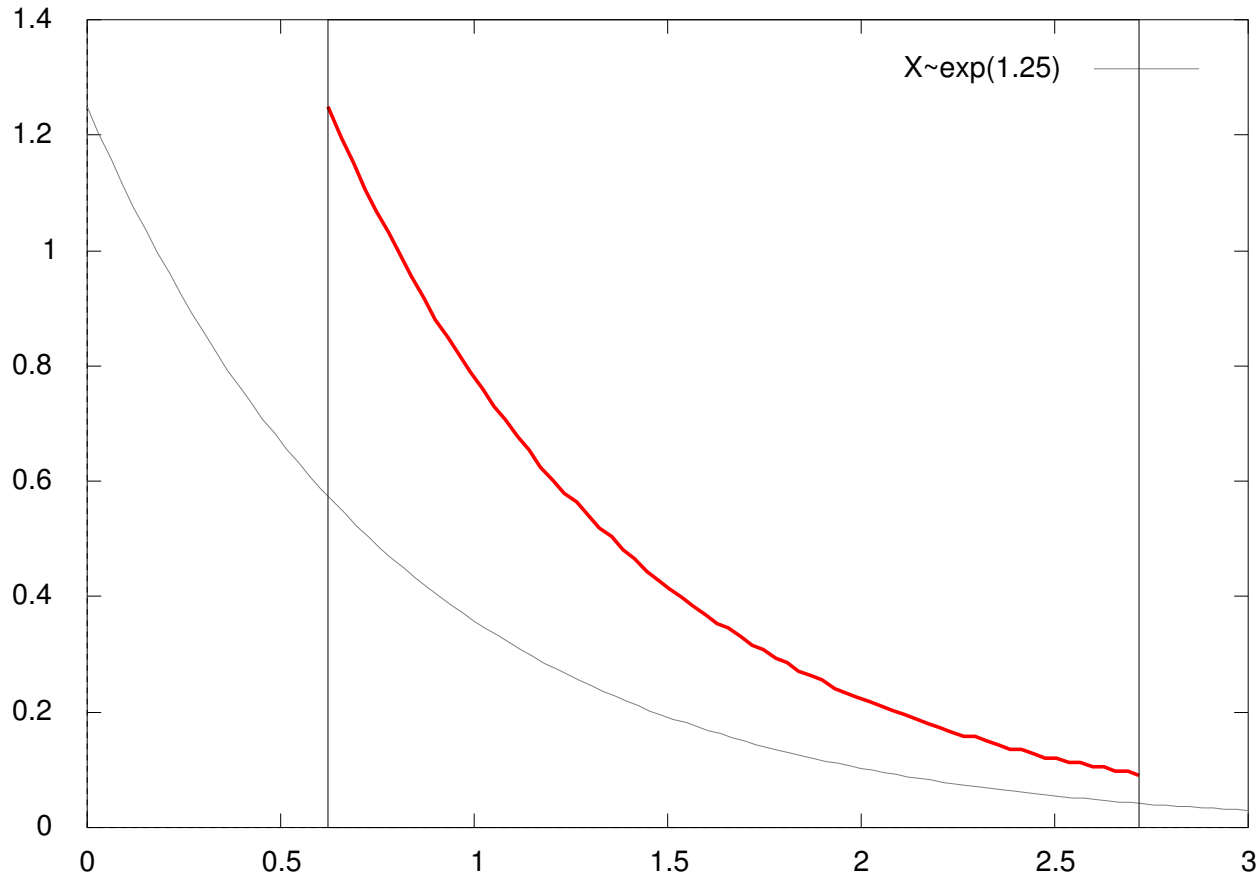
Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

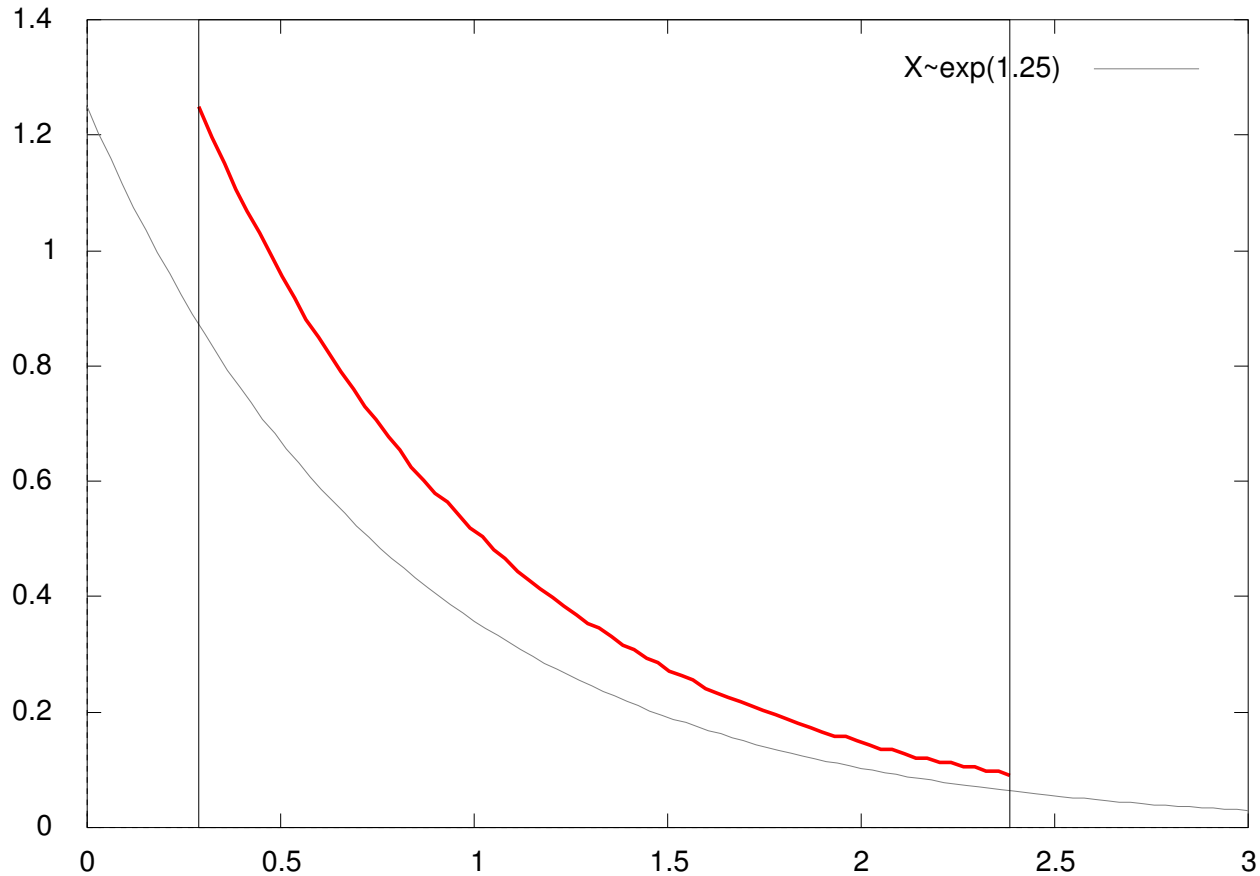
Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

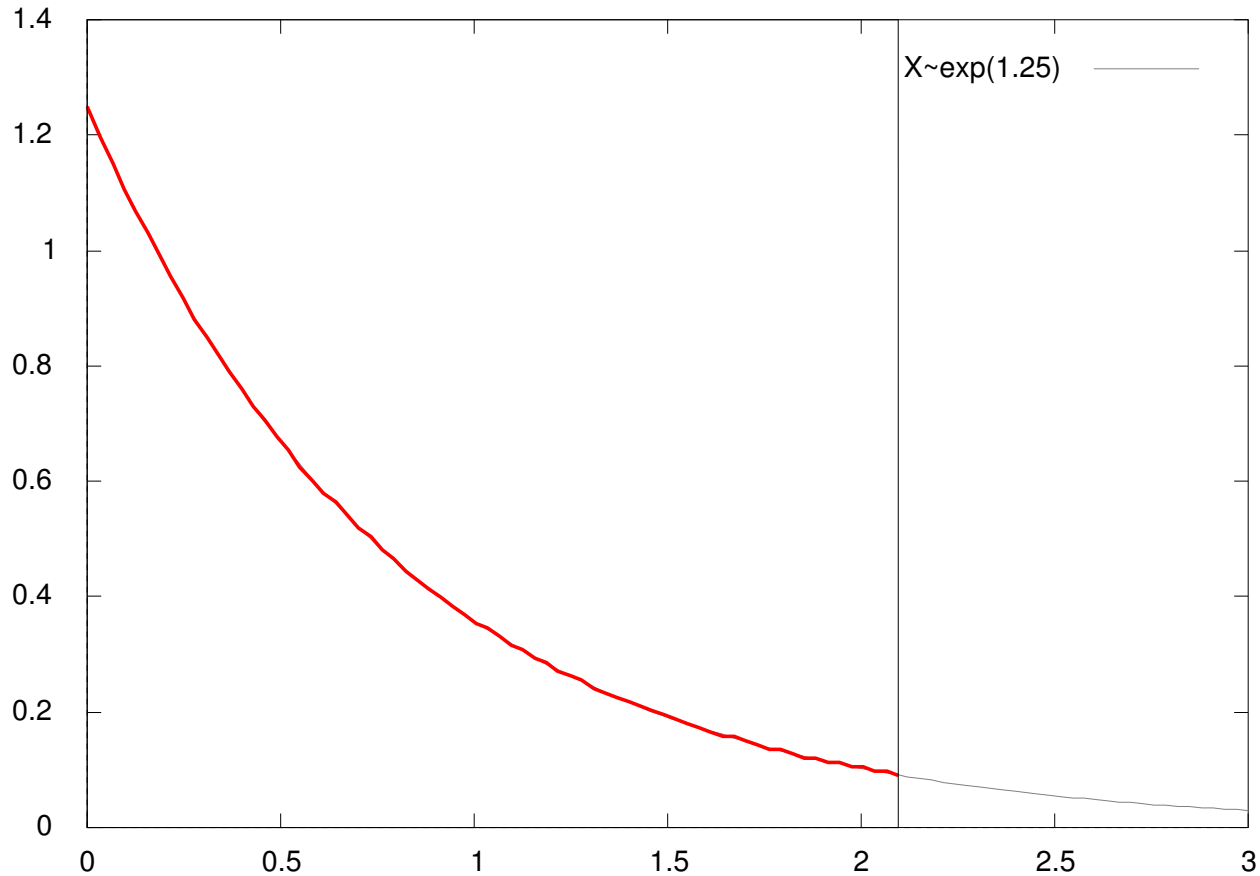
Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

Memorylessness



What is $\mathbb{P}(X \leq t \mid X > u)$ if $X \sim \exp(\lambda)$?

Why exponential?

1. Described by a single parameter

Why exponential?

1. Described by a single parameter
2. Memorylessness

Why exponential?

1. Described by a single parameter
2. Memorylessness
3. Ability to describe other distributions using phase-type combinations

Prefix: $(a, \lambda).A$

- Prefix is used to describe a process that evolves from one state to another by *emitting* or *performing* an action

Prefix: $(a, \lambda).A$

- Prefix is used to describe a process that evolves from one state to another by *emitting* or *performing* an action
- Example:

$$P \stackrel{def}{=} (a, \lambda).Q$$

...means that the process P evolves with rate λ to become process Q, by emitting an a-action

Prefix: $(a, \lambda).A$

- Prefix is used to describe a process that evolves from one state to another by *emitting* or *performing* an action
- Example:

$$P \stackrel{def}{=} (a, \lambda).Q$$

...means that the process P evolves with rate λ to become process Q, by emitting an a-action

- λ is an exponential rate parameter

Prefix: $(a, \lambda).A$

- Prefix is used to describe a process that evolves from one state to another by *emitting* or *performing* an action
- Example:

$$P \stackrel{def}{=} (a, \lambda).Q$$

...means that the process P evolves with rate λ to become process Q, by emitting an a-action

- λ is an exponential rate parameter
- As a labelled transition system, this becomes:

Prefix : $\quad P \xrightarrow{(a,\lambda)} Q$

Choice: $P_1 + P_2$

- PEPA uses a type of choice known as *competitive choice*

Choice: $P_1 + P_2$

- PEPA uses a type of choice known as *competitive choice*
- Example:

$$P \stackrel{def}{=} (a, \lambda).P_1 + (b, \mu).P_2$$

means that P can evolve *either* to produce an a-action with rate λ *or* to produce a b-action with rate μ

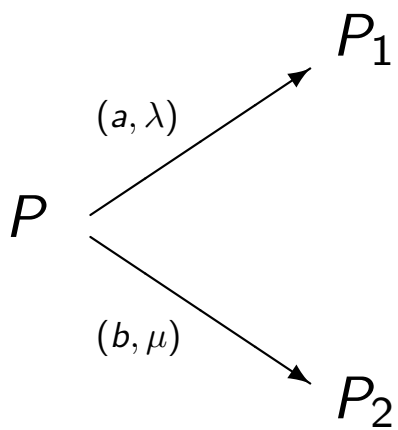
Choice: $P_1 + P_2$

- PEPA uses a type of choice known as *competitive choice*
- Example:

$$P \stackrel{def}{=} (a, \lambda).P_1 + (b, \mu).P_2$$

means that P can evolve *either* to produce an a-action with rate λ *or* to produce a b-action with rate μ

- As a labelled transition system:



Choice: $P_1 + P_2$

- $P \stackrel{def}{=} (a, \lambda).P_1 + (b, \mu).P_2$
- This is competitive choice since:
  - P_1 and P_2 are in a *race condition* – the first one to perform an a or a b will dictate the direction of choice for $P_1 + P_2$
- What is the probability that we see an a-action?

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- $P_1 \underset{L}{\bowtie} P_2$ defines concurrency and communication within PEPA

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- $P_1 \underset{L}{\bowtie} P_2$ defines concurrency and communication within PEPA
- The L in $P_1 \underset{L}{\bowtie} P_2$ defines the set of actions over which two components are to cooperate

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- $P_1 \underset{L}{\bowtie} P_2$ defines concurrency and communication within PEPA
- The L in $P_1 \underset{L}{\bowtie} P_2$ defines the set of actions over which two components are to cooperate
- Any other actions that P_1 and P_2 can do, not mentioned in L, can happen independently

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- $P_1 \underset{L}{\bowtie} P_2$ defines concurrency and communication within PEPA
- The L in $P_1 \underset{L}{\bowtie} P_2$ defines the set of actions over which two components are to cooperate
- Any other actions that P_1 and P_2 can do, not mentioned in L, can happen independently
- If $a \in L$ and P_1 enables an a, then P_1 has to wait for P_2 to enable an a before the cooperation can proceed

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- $P_1 \underset{L}{\bowtie} P_2$ defines concurrency and communication within PEPA
- The L in $P_1 \underset{L}{\bowtie} P_2$ defines the set of actions over which two components are to cooperate
- Any other actions that P_1 and P_2 can do, not mentioned in L, can happen independently
- If $a \in L$ and P_1 enables an a, then P_1 has to wait for P_2 to enable an a before the cooperation can proceed
- Easy source of deadlock!

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- If $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\top)} P_2'$ then:

$$P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,\lambda)} P_1' \underset{\{a\}}{\bowtie} P_2'$$

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- If $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\top)} P_2'$ then:

$$P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,\lambda)} P_1' \underset{\{a\}}{\bowtie} P_2'$$

- $\top$ represents a passive rate which, in the cooperation, inherits the λ-rate of from P_1

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- If $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\top)} P_2'$ then:

$$P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,\lambda)} P_1' \underset{\{a\}}{\bowtie} P_2'$$

- $\top$ represents a passive rate which, in the cooperation, inherits the λ-rate of from P_1
- If both rates are specified and the only a-evolutions allowed from P_1 and P_2 are, $P_1 \xrightarrow{(a,\lambda)} P_1'$ and $P_2 \xrightarrow{(a,\mu)} P_2'$ then:

$$P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,\min(\lambda,\mu))} P_1' \underset{\{a\}}{\bowtie} P_2'$$

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- The general cooperation case is where:
  - P_1 enables m a-actions
  - P_2 enables n a-actions

  at the moment of cooperation
- ...in which case there are $m \times n$ possible transitions for $P_1 \underset{\{a\}}{\bowtie} P_2$

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- The general cooperation case is where:
  - P_1 enables m a-actions
  - P_2 enables n a-actions

  at the moment of cooperation
- ...in which case there are $m \times n$ possible transitions for $P_1 \underset{\{a\}}{\bowtie} P_2$
- with mn a-actions having cumulative rate $P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,R)}$ where $R = \min(r_a(P_1), r_a(P_2))$

Cooperation: $P_1 \underset{L}{\bowtie} P_2$

- The general cooperation case is where:
  - P_1 enables m a-actions
  - P_2 enables n a-actions

  at the moment of cooperation
- ...in which case there are $m \times n$ possible transitions for $P_1 \underset{\{a\}}{\bowtie} P_2$
- with mn a-actions having cumulative rate $P_1 \underset{\{a\}}{\bowtie} P_2 \xrightarrow{(a,R)}$ where $R = \min(r_a(P_1), r_a(P_2))$
- $r_a(P) = \sum_{i:P \xrightarrow{(a,\, r_i)}} r_i$ is the apparent rate of an action a – the total rate at which P can do a

Hiding: P/L

- Used to turn observable actions in P into hidden or silent actions in P/L
- L defines the set of actions to hide
- If $P \xrightarrow{(a,\lambda)} P'$:

$$P/\{a\} \xrightarrow{(\tau,\lambda)} P'/\{a\}$$

- τ is the *silent* action
- Used to hide complexity and create a component interface
- Cooperation on τ not allowed

PEPA: A Transmitter-Receiver

$$System \stackrel{def}{=} (Transmitter \underset{\emptyset}{\bowtie} Receiver) \underset{L}{\bowtie} Network$$

A simple model of a transmitter–receiver over a network

PEPA: A Transmitter-Receiver

$$System \stackrel{def}{=} (Transmitter \parallel Receiver) \underset{L}{\bowtie} Network$$

A simple model of a transmitter–receiver over a network

PEPA: A Transmitter-Receiver

$$System \stackrel{def}{=} (Transmitter \parallel Receiver) \underset{L}{\bowtie} Network$$

$$Transmitter \stackrel{def}{=} (transmit, \lambda_1).(t\_recover, \lambda_2).Transmitter$$

$$Receiver \stackrel{def}{=} (receive, \top).(r\_recover, \mu).Receiver$$

$$Network \stackrel{def}{=} (transmit, \top).(delay, \nu_1).(receive, \nu_2).Network$$

A simple model of a transmitter–receiver over a network

PEPA: A Transmitter-Receiver

$$System \stackrel{def}{=} (Transmitter \parallel Receiver) \underset{L}{\bowtie} Network$$

$$Transmitter \stackrel{def}{=} (transmit, \lambda_1).(t\_recover, \lambda_2).Transmitter$$

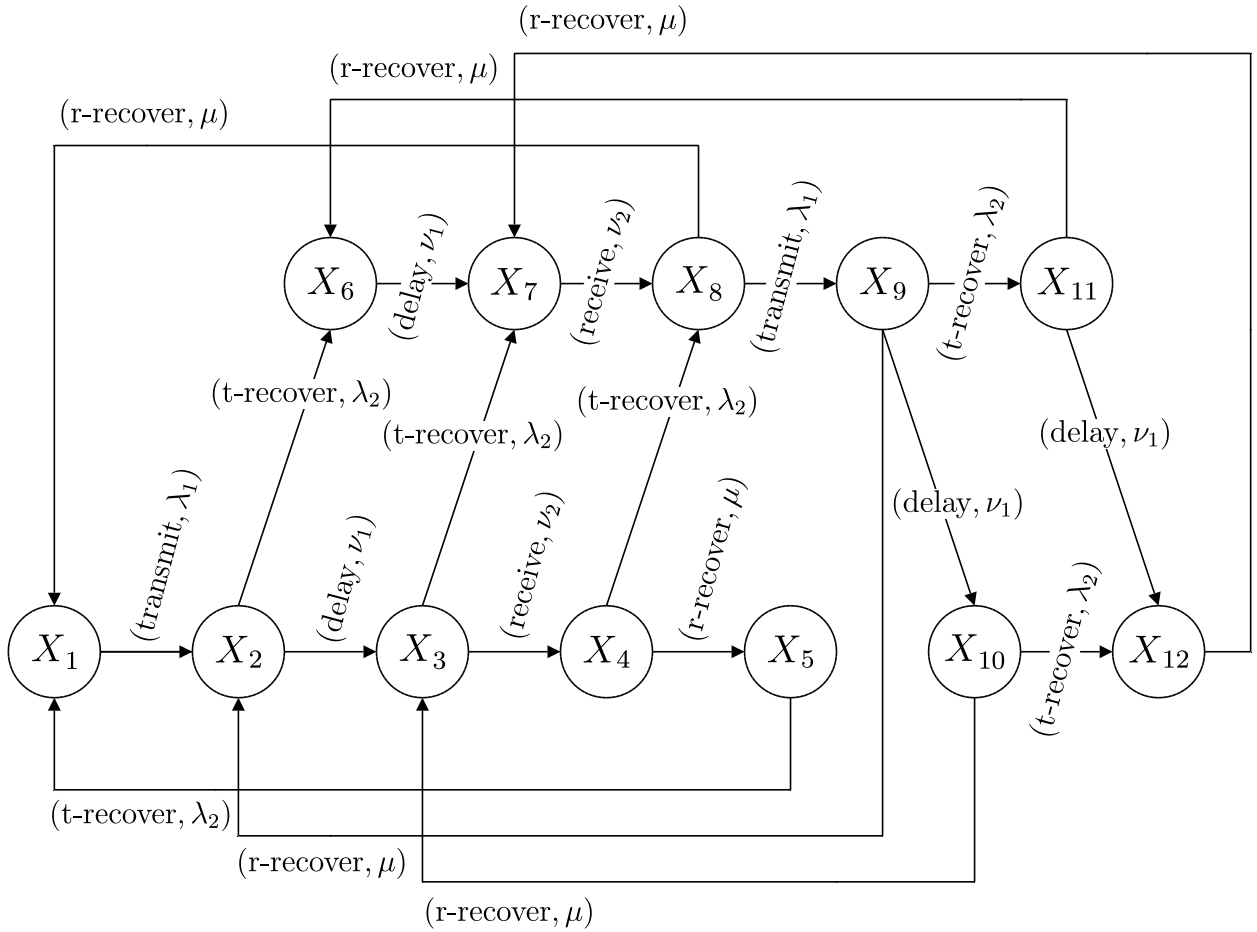
$$Receiver \stackrel{def}{=} (receive, \top).(r\_recover, \mu).Receiver$$

$$Network \stackrel{def}{=} (transmit, \top).(delay, \nu_1).(receive, \nu_2).Network$$

where $L = \{transmit, receive\}$.

A simple model of a transmitter–receiver over a network

TR example: Labelled transition system



with $X_1 \rightarrow (\mathit{Transmitter} \mid\mid \mathit{Receiver}) \underset{L}{\bowtie} \mathit{Network}$

$X_2 \rightarrow (\mathit{Transmitter}' \mid\mid \mathit{Receiver}) \underset{L}{\bowtie} \mathit{Network}'$ and so on.

Voting Example I

Voters vote and Pollers record those votes.

Pollers can break individually and recover individually. If all Pollers break then they are all repaired in unison.

$$System \stackrel{def}{=} (Voter \;||\; Voter \;||\; Voter) \underset{\{vote\}}{\bowtie} ((Poller \underset{L}{\bowtie} Poller) \underset{L'}{\bowtie} Poller\_group\_0)$$

where

- $L = \{recover\_all\}$
- $L' = \{recover, break, recover\_all\}$

Voting Example II

$$Voter \stackrel{def}{=} (vote, \lambda).(pause, \mu).Voter$$

Voting Example II

$$Voter \stackrel{def}{=} (vote, \lambda).(pause, \mu).Voter$$

$$Poller \stackrel{def}{=} (vote, \top).(register, \gamma).Poller$$
$$+ (break, \nu).Poller\_broken$$

Voting Example II

$$
\begin{aligned}
\mathit{Voter} &\stackrel{\mathit{def}}{=} (\mathit{vote}, \lambda).(\mathit{pause}, \mu).\mathit{Voter} \\
\mathit{Poller} &\stackrel{\mathit{def}}{=} (\mathit{vote}, \top).(\mathit{register}, \gamma).\mathit{Poller} \\
&\quad + (\mathit{break}, \nu).\mathit{Poller\_broken} \\
\mathit{Poller\_broken} &\stackrel{\mathit{def}}{=} (\mathit{recover}, \tau).\mathit{Poller} \\
&\quad + (\mathit{recover\_all}, \top).\mathit{Poller}
\end{aligned}
$$

Voting Example III

$$Poller\_group\_0 \stackrel{def}{=} (break, \top).Poller\_group\_1$$

Voting Example III

$$Poller\_group\_0 \stackrel{def}{=} (break, \top).Poller\_group\_1$$

$$Poller\_group\_1 \stackrel{def}{=} (break, \top).Poller\_group\_2 + (recover, \top).Poller\_group\_0$$

Voting Example III

$$
\begin{aligned}
\mathit{Poller\_group\_0} &\stackrel{def}{=} (\mathit{break}, \top).\mathit{Poller\_group\_1} \\
\mathit{Poller\_group\_1} &\stackrel{def}{=} (\mathit{break}, \top).\mathit{Poller\_group\_2} \\
&\quad + (\mathit{recover}, \top).\mathit{Poller\_group\_0} \\
\mathit{Poller\_group\_2} &\stackrel{def}{=} (\mathit{recover\_all}, \delta) \\
&\quad .\mathit{Poller\_group\_0}
\end{aligned}
$$

An Overview of model-based Fluid Analysis

Mean field/fluid analysis

- Addresses the state-space explosion problem for discrete-state Markov models of computer and communication systems

[12] Jane Hillston. "Fluid flow approximation of PEPA models". In: *Second International Conference on the Quantitative Evaluation of Systems (QEST)*. IEEE, Sept. 2005, pp. 33–42. DOI: 10.1109/QEST.2005.12.

[13] Michel Benaïm and Jean-Yves Le Boudec. "A class of mean field interaction models for computer and communication systems". In: *Performance Evaluation* 65.11-12 (Nov. 2008), pp. 823–838. DOI: 10.1016/j.peva.2008.03.005.

[14] Marco Gribaudo. "Analysis of Large Populations of Interacting Objects with Mean Field and Markovian Agents". In: *6th European Performance Engineering Workshop (EPEW)*. Vol. 5652. 2009, pp. 218–219. DOI: 10.1007/978-3-642-02924-0.

Mean field/fluid analysis

- Addresses the state-space explosion problem for discrete-state Markov models of computer and communication systems
- Derives tractable systems of differential equations approximating mean number of components in each local state, for example:
  - *Fluid analysis* of process algebra models[12]
  - *Mean-field* analysis of systems of interacting objects[13,14]

[12] Jane Hillston. "Fluid flow approximation of PEPA models". In: *Second International Conference on the Quantitative Evaluation of Systems (QEST)*. IEEE, Sept. 2005, pp. 33–42. DOI: 10.1109/QEST.2005.12.

[13] Michel Benaïm and Jean-Yves Le Boudec. "A class of mean field interaction models for computer and communication systems". In: *Performance Evaluation* 65.11-12 (Nov. 2008), pp. 823–838. DOI: 10.1016/j.peva.2008.03.005.

[14] Marco Gribaudo. "Analysis of Large Populations of Interacting Objects with Mean Field and Markovian Agents". In: *6th European Performance Engineering Workshop (EPEW)*. Vol. 5652. 2009, pp. 218–219. DOI: 10.1007/978-3-642-02924-0.

Mean field/fluid analysis

- Addresses the state-space explosion problem for discrete-state Markov models of computer and communication systems
- Derives tractable systems of differential equations approximating mean number of components in each local state, for example:
  - *Fluid analysis* of process algebra models[12]
  - *Mean-field* analysis of systems of interacting objects[13,14]
- Can develop these techniques to capture key performance measures of interest from large CTMCs, e.g. passage-time measures, reward-based measures

[12] Jane Hillston. "Fluid flow approximation of PEPA models". In: *Second International Conference on the Quantitative Evaluation of Systems (QEST)*. IEEE, Sept. 2005, pp. 33–42. DOI: 10.1109/QEST.2005.12.

[13] Michel Benaïm and Jean-Yves Le Boudec. "A class of mean field interaction models for computer and communication systems". In: *Performance Evaluation* 65.11-12 (Nov. 2008), pp. 823–838. DOI: 10.1016/j.peva.2008.03.005.

[14] Marco Gribaudo. "Analysis of Large Populations of Interacting Objects with Mean Field and Markovian Agents". In: *6th European Performance Engineering Workshop (EPEW)*. Vol. 5652. 2009, pp. 218–219. DOI: 10.1007/978-3-642-02924-0.

A simple agent

A simple agent – replicated

A simple agent – replicated

A simple agent – replicated

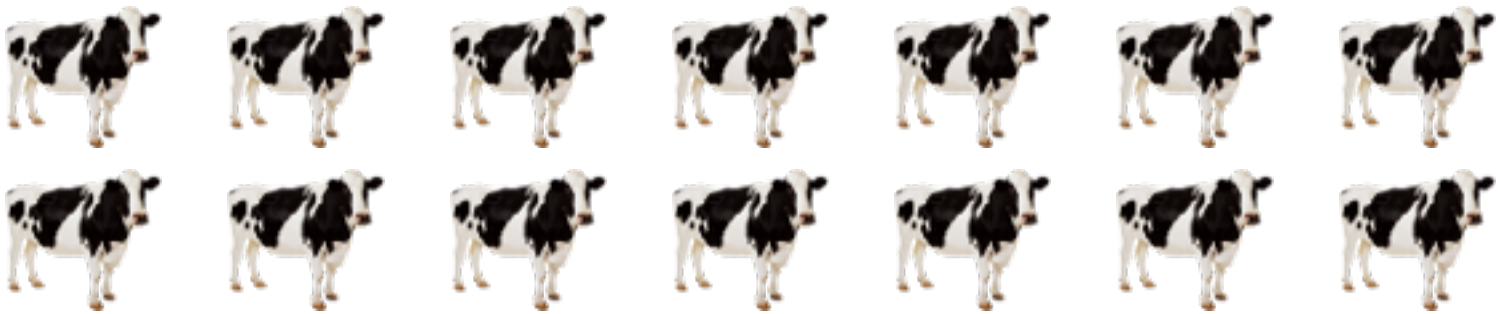
A simple agent – replicated

A simple agent – replicated

A simple agent – replicated

A simple agent – replicated

A simple agent – replicated



A simple agent – replicated

A simple agent – replicated

A simple agent – replicated

Fluid/mean field analysis works best when you have many replicated parallel agents or groups of replicated parallel agents. Agent groups can synchronise.

GPEPA – Syntax

GPEPA or Grouped PEPA as a syntax that is suspiciously similar to that of PEPA.

GPEPA – Syntax

GPEPA or Grouped PEPA as a syntax that is suspiciously similar to that of PEPA.

An sequential agent, P, can have the following syntax:

GPEPA – Syntax

GPEPA or Grouped PEPA as a syntax that is suspiciously similar to that of PEPA.

An sequential agent, P, can have the following syntax:

$P ::= (a, \lambda).P$ SPA Markovian prefix

GPEPA – Syntax

GPEPA or Grouped PEPA as a syntax that is suspiciously similar to that of PEPA.

An sequential agent, P, can have the following syntax:

$$P ::= (a, \lambda).P \qquad \text{SPA Markovian prefix}$$
$$\mid \; P + P \qquad \text{SPA competitive choice}$$

GPEPA – Syntax

GPEPA or Grouped PEPA as a syntax that is suspiciously similar to that of PEPA.

An sequential agent, P, can have the following syntax:

$$
\begin{array}{lr}
P ::= (a, \lambda).P & \text{SPA Markovian prefix} \\
\quad | \quad P + P & \text{SPA competitive choice} \\
\quad | \quad C \stackrel{def}{=} P & \text{Agent name}
\end{array}
$$

GPEPA – Syntax

GPEPA or Grouped PEPA as a syntax that is suspiciously similar to that of PEPA.

An sequential agent, P, can have the following syntax:

$$
\begin{array}{lr}
P ::= (a, \lambda).P & \text{SPA Markovian prefix} \\
\quad | \quad P + P & \text{SPA competitive choice} \\
\quad | \quad C \stackrel{def}{=} P & \text{Agent name}
\end{array}
$$

Sequential agents allow a modeller to define behaviour with associated exponential delays.

GPEPA – Syntax

For parallelism and communication between sequential agents, we need compositional agents.

GPEPA – Syntax

For parallelism and communication between sequential agents, we need compositional agents.

A compositional agent, Q, can have the following syntax:

GPEPA – Syntax

For parallelism and communication between sequential agents, we need compositional agents.

A compositional agent, Q, can have the following syntax:

$$
\begin{array}{lll}
Q ::= Q \underset{L}{\bowtie} Q & & \text{Group cooperation} \\
\quad | \quad \mathbf{Group}\{P[n]\} & & \text{Parallel grouping}
\end{array}
$$

where $P[n]$ represents a parallel group of n sequential agents P. **Group** represents a group label used to identify the parts of the model that are going to be approximated using fluid analysis.

GPEPA Example

$$Client \stackrel{def}{=} (req, r_{req}).Client\_waiting$$

$$Client\_waiting \stackrel{def}{=} (data, r_{data}).Client\_think$$

$$Client\_think \stackrel{def}{=} (think, r_{think}).Client$$

GPEPA Example

$$Client \stackrel{def}{=} (req, r_{req}).Client\_waiting$$

$$Client\_waiting \stackrel{def}{=} (data, r_{data}).Client\_think$$

$$Client\_think \stackrel{def}{=} (think, r_{think}).Client$$

$$Server \stackrel{def}{=} (req, r_{req}).Server\_get + (break, r_{break}).Server\_broken$$

$$Server\_get \stackrel{def}{=} (data, r_{data}).Server$$

$$Server\_broken \stackrel{def}{=} (reset, r_{reset}).Server$$

GPEPA Example

$$Client \stackrel{def}{=} (req, r_{req}).Client\_waiting$$

$$Client\_waiting \stackrel{def}{=} (data, r_{data}).Client\_think$$

$$Client\_think \stackrel{def}{=} (think, r_{think}).Client$$

$$Server \stackrel{def}{=} (req, r_{req}).Server\_get + (break, r_{break}).Server\_broken$$

$$Server\_get \stackrel{def}{=} (data, r_{data}).Server$$

$$Server\_broken \stackrel{def}{=} (reset, r_{reset}).Server$$

$$CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}$$

GPEPA Example

$$Client \stackrel{def}{=} (req, r_{req}).Client\_waiting \qquad C(t)$$

$$Client\_waiting \stackrel{def}{=} (data, r_{data}).Client\_think$$

$$Client\_think \stackrel{def}{=} (think, r_{think}).Client$$

$$Server \stackrel{def}{=} (req, r_{req}).Server\_get + (break, r_{break}).Server\_broken$$

$$Server\_get \stackrel{def}{=} (data, r_{data}).Server$$

$$Server\_broken \stackrel{def}{=} (reset, r_{reset}).Server$$

$$CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}$$

GPEPA Example

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get\\
&\qquad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

GPEPA Example

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get\\
&\qquad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

GPEPA Example

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get && S(t)\\
&\quad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

GPEPA Example

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get && S(t)\\
&\quad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server && S_g(t)\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

GPEPA Example

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get && S(t)\\
&\quad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server && S_g(t)\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server && S_b(t)
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

ODEs – Means

Ideally, we want the distribution of say $C(t)$ for each t

ODEs – Means

Ideally, we want the distribution of say $C(t)$ for each t
This can be too expensive

ODEs – Means

Ideally, we want the distribution of say $C(t)$ for each t
This can be too expensive
Can derive ODEs approximating the means

$$\begin{array}{lcllcl} \mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t & = & \cdots \\ \mathrm{d}\mathbb{E}[C_w(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_g(t)]/\mathrm{d}t & = & \cdots \\ \mathrm{d}\mathbb{E}[C_t(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_b(t)]/\mathrm{d}t & = & \cdots \end{array}$$

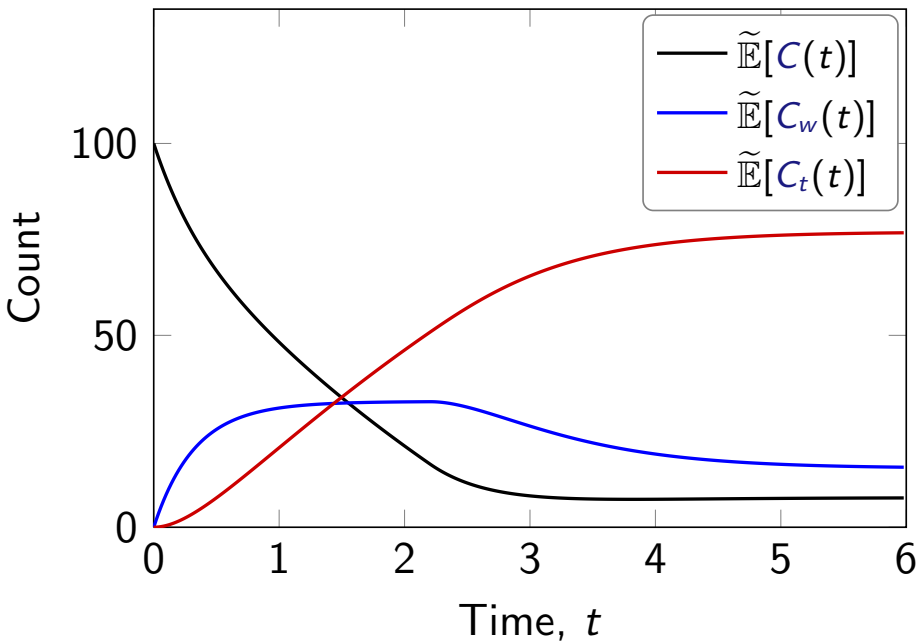
ODEs – Means

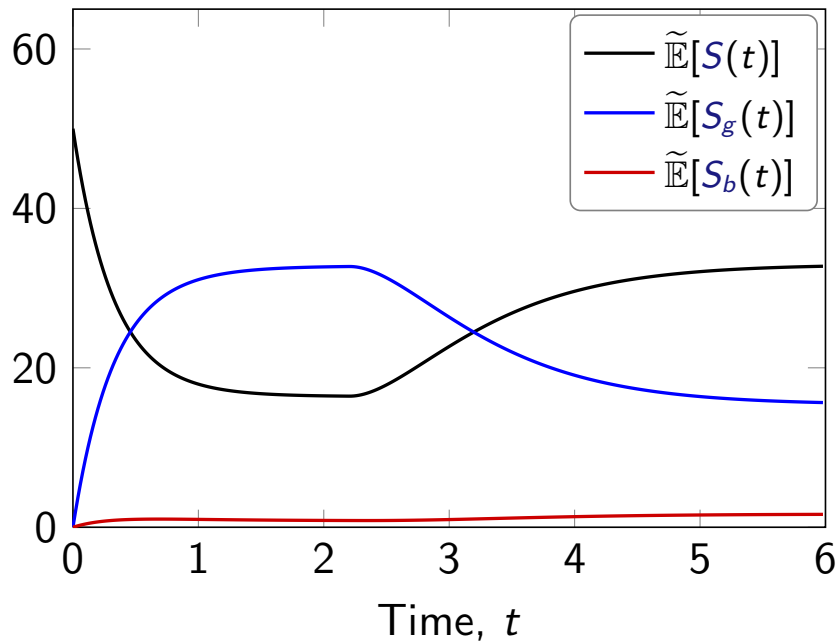
Ideally, we want the distribution of say $C(t)$ for each t
This can be too expensive
Can derive ODEs approximating the means

$$\begin{array}{lcllcl} \mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t & = & \cdots \\ \mathrm{d}\mathbb{E}[C_w(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_g(t)]/\mathrm{d}t & = & \cdots \\ \mathrm{d}\mathbb{E}[C_t(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_b(t)]/\mathrm{d}t & = & \cdots \end{array}$$

These can be numerically solved, cheaper than simulation

ODEs – Means


$\widetilde{\mathbb{E}}[C(t)]$
$\widetilde{\mathbb{E}}[C_w(t)]$
$\widetilde{\mathbb{E}}[C_t(t)]$
Count
100
50
0
0
1
2
3
4
5
6
Time, t


$\widetilde{\mathbb{E}}[S(t)]$
$\widetilde{\mathbb{E}}[S_g(t)]$
$\widetilde{\mathbb{E}}[S_b(t)]$
60
40
20
0
0
1
2
3
4
5
6
Time, t

ODEs – Higher moments

Can extend the ODEs

$$
\begin{array}{llll}
\mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t & = & \cdots \\
\mathrm{d}\mathbb{E}[C_w(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_g(t)]/\mathrm{d}t & = & \cdots \\
\mathrm{d}\mathbb{E}[C_t(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_b(t)]/\mathrm{d}t & = & \cdots
\end{array}
$$

ODEs – Higher moments

Can extend the ODEs

$$\begin{aligned} \mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t &= \cdots & \mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t &= \cdots \\ \mathrm{d}\mathbb{E}[C_w(t)]/\mathrm{d}t &= \cdots & \mathrm{d}\mathbb{E}[S_g(t)]/\mathrm{d}t &= \cdots \\ \mathrm{d}\mathbb{E}[C_t(t)]/\mathrm{d}t &= \cdots & \mathrm{d}\mathbb{E}[S_b(t)]/\mathrm{d}t &= \cdots \end{aligned}$$

with ODEs for higher moments

$$\begin{aligned} \mathrm{d}\mathbb{E}[S_g(t)^2]/\mathrm{d}t &= \cdots \\ \mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t &= \cdots \\ &\vdots \end{aligned}$$

ODEs – Higher moments

Can extend the ODEs

$$\begin{aligned} \mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t &= \cdots & \mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t &= \cdots \\ \mathrm{d}\mathbb{E}[C_w(t)]/\mathrm{d}t &= \cdots & \mathrm{d}\mathbb{E}[S_g(t)]/\mathrm{d}t &= \cdots \\ \mathrm{d}\mathbb{E}[C_t(t)]/\mathrm{d}t &= \cdots & \mathrm{d}\mathbb{E}[S_b(t)]/\mathrm{d}t &= \cdots \end{aligned}$$

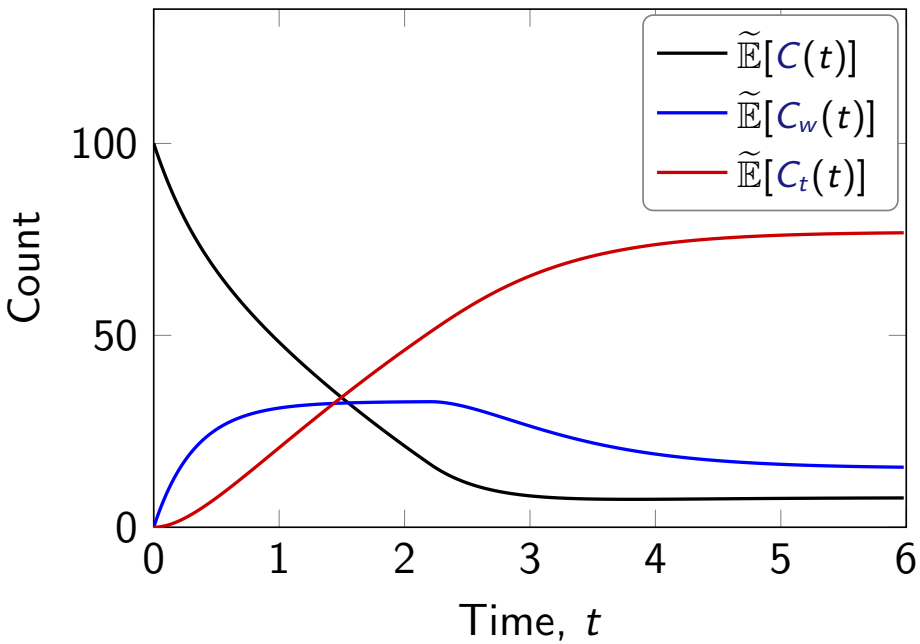
with ODEs for higher moments

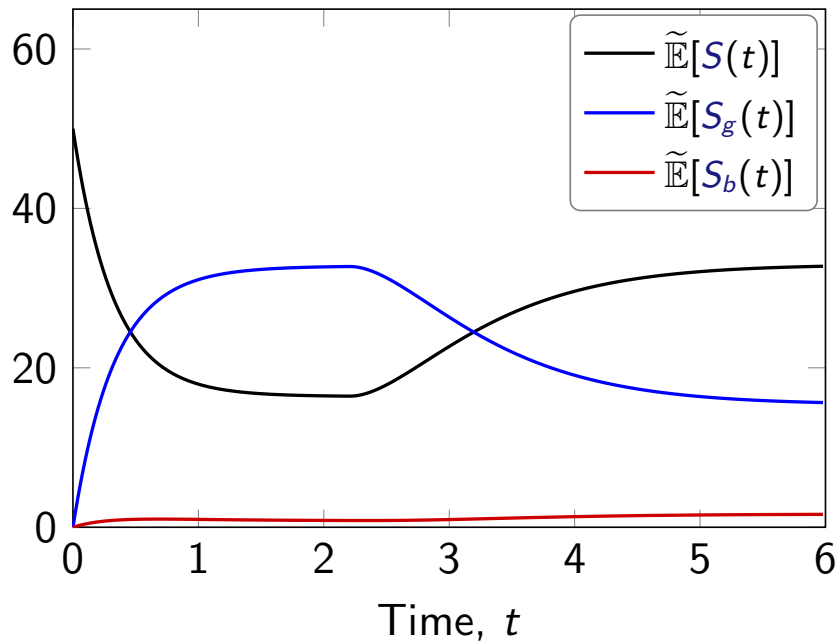
$$\begin{aligned} \mathrm{d}\mathbb{E}[S_g(t)^2]/\mathrm{d}t &= \cdots \\ \mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t &= \cdots \\ &\vdots \end{aligned}$$

E.g. can get variance as

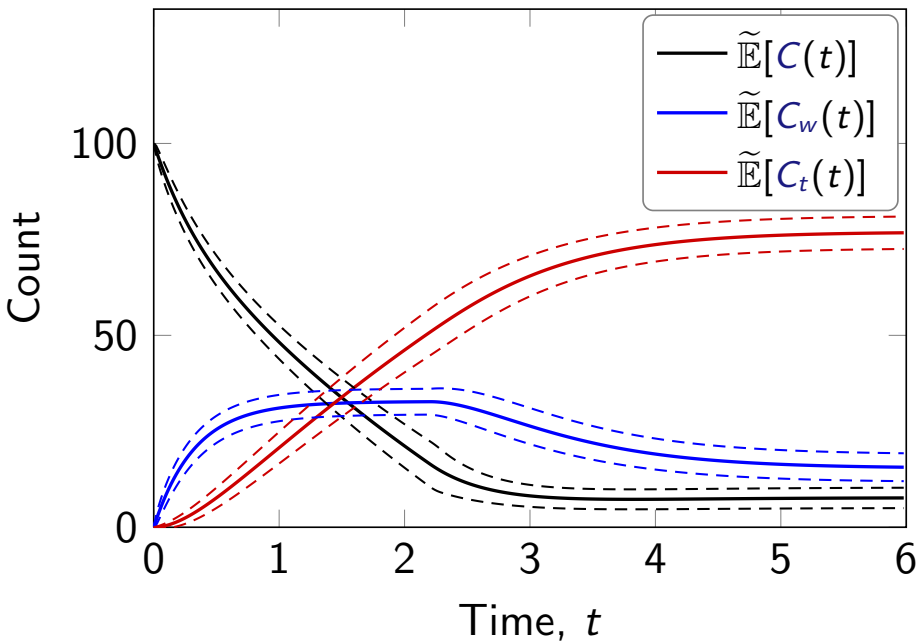
$$\mathrm{Var}[C(t)] = \mathbb{E}[C(t)^2] - \mathbb{E}[C(t)]^2$$

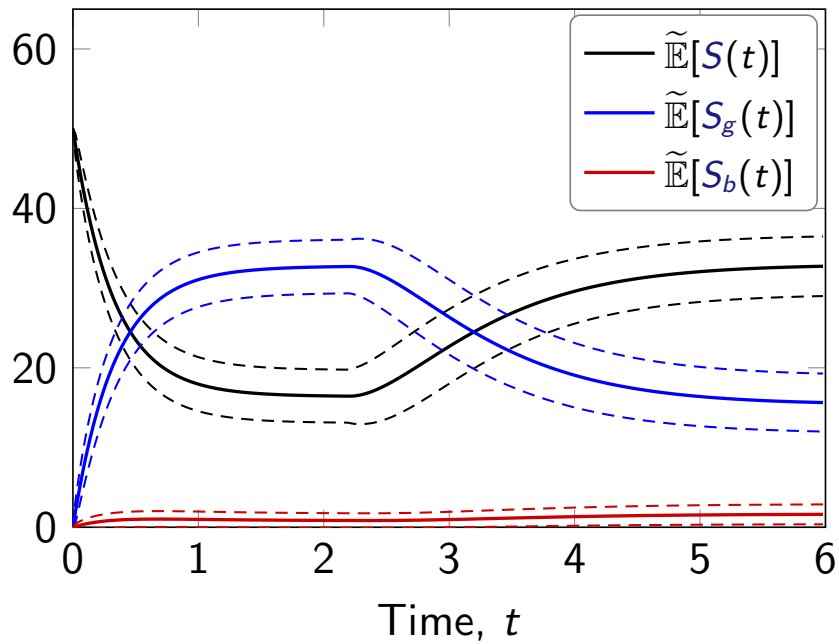
ODEs – Higher moments





ODEs – Higher moments





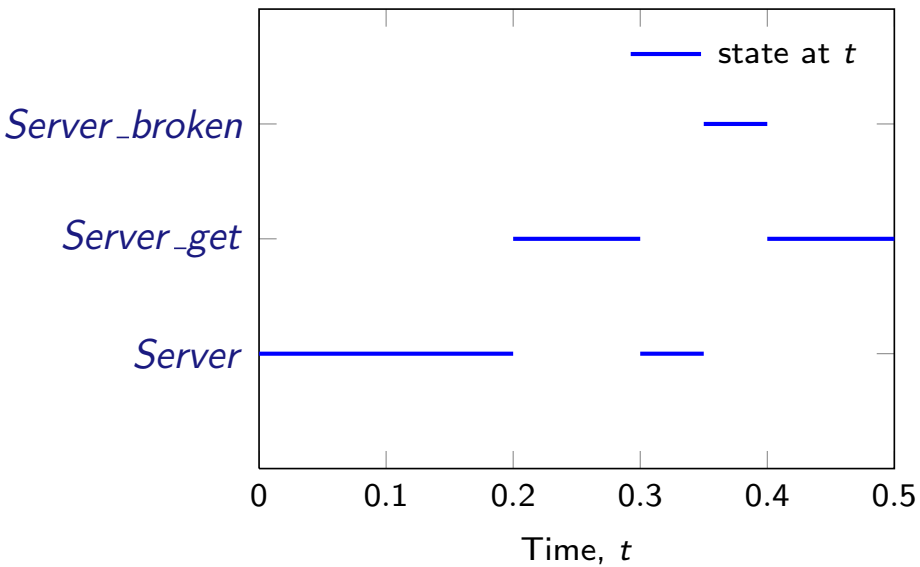
Accumulated rewards

How to analyse quantities accumulated over time, e.g. energy consumption?

Accumulated rewards

How to analyse quantities accumulated over time, e.g. energy consumption?

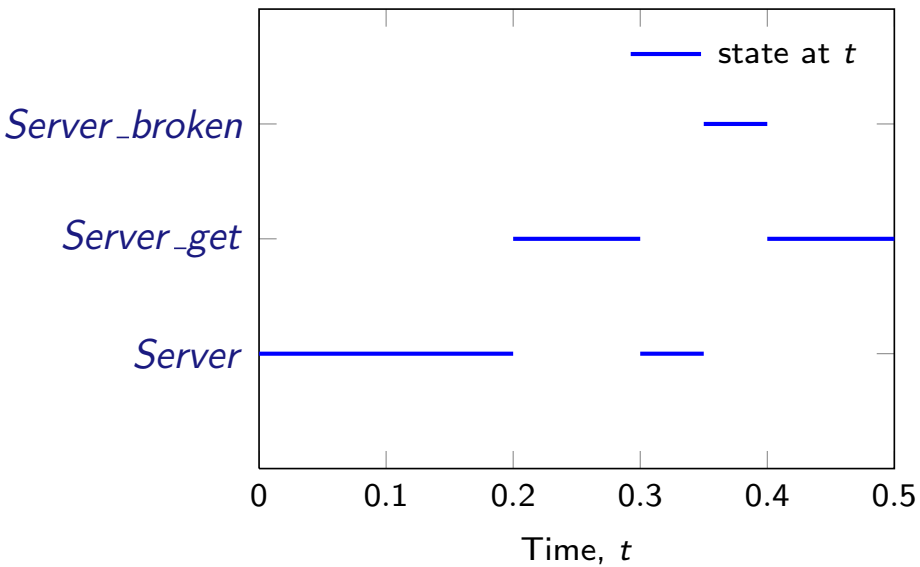
Servers consume energy in the *Server_get* state



Accumulated rewards

How to analyse quantities accumulated over time, e.g. energy consumption?

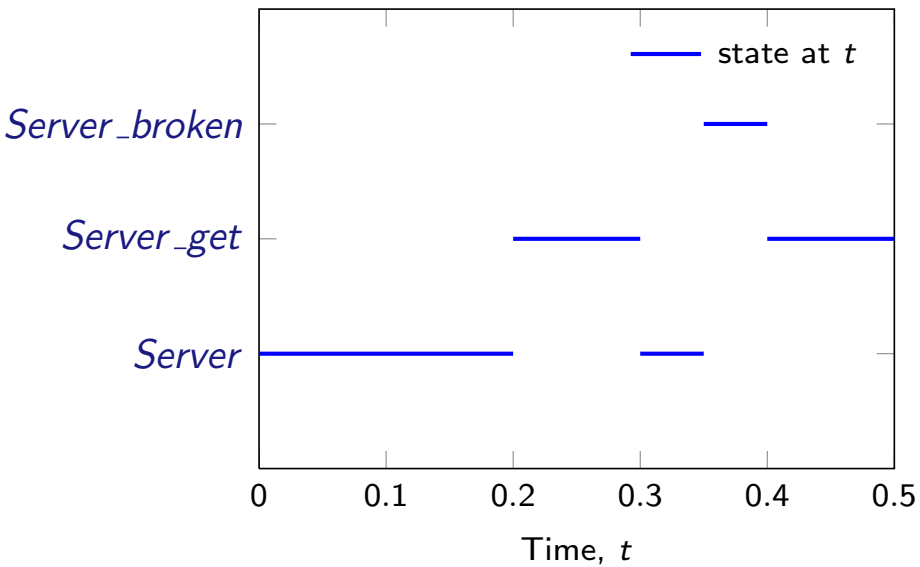
Servers consume energy in the *Server_get* state

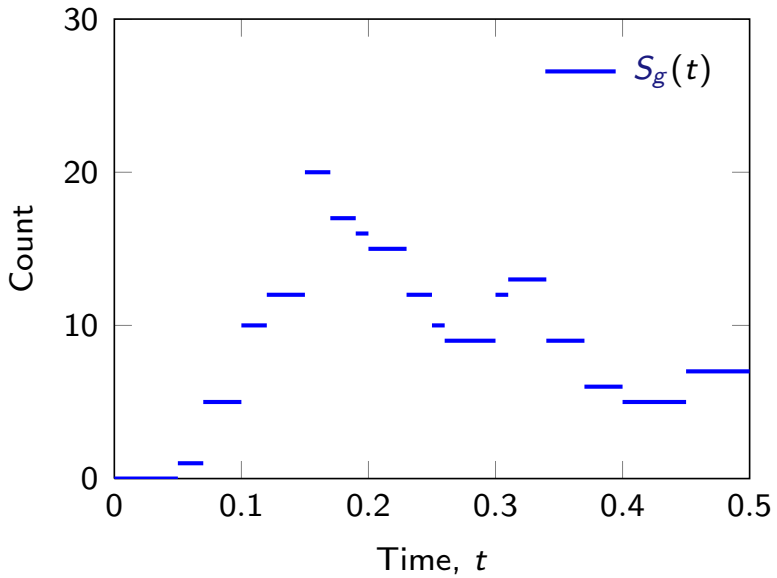


Accumulated rewards

How to analyse quantities accumulated over time, e.g. energy consumption?

Servers consume energy in the *Server_get* state

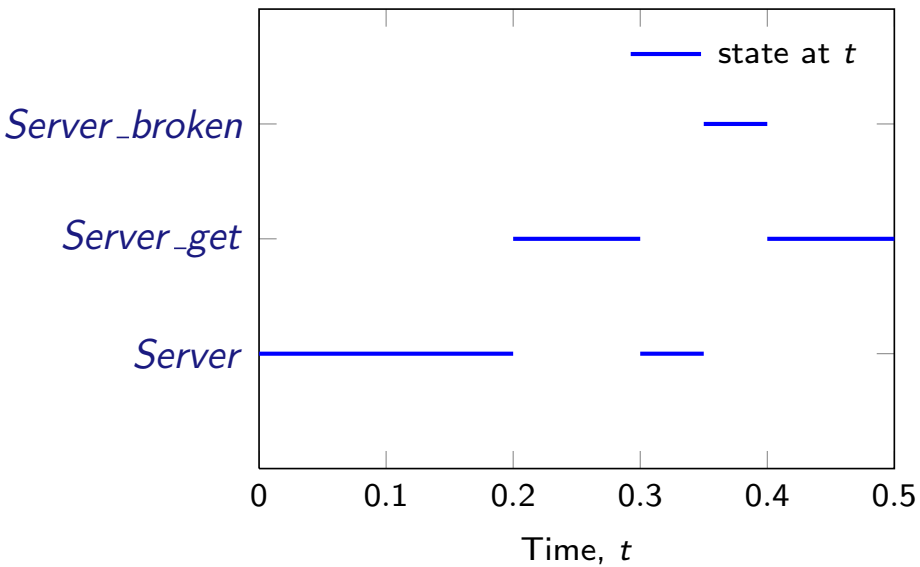


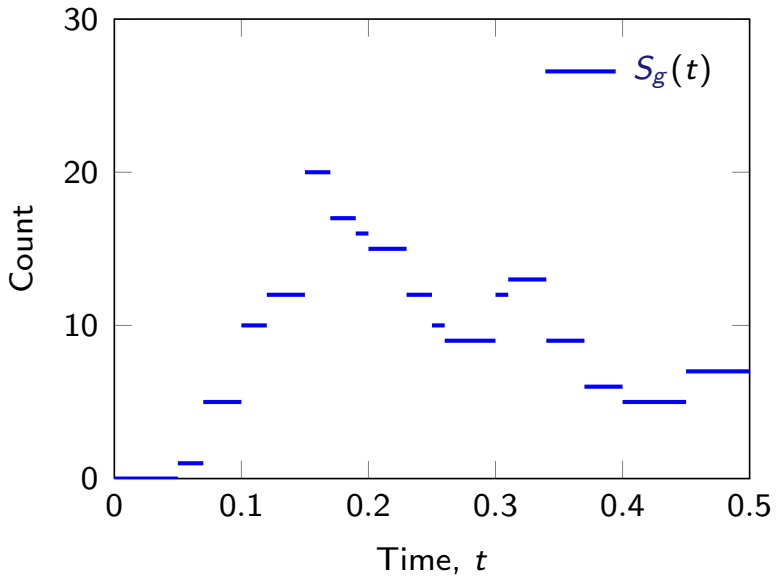


Accumulated rewards

How to analyse quantities accumulated over time, e.g. energy consumption?

Servers consume energy in the *Server_get* state

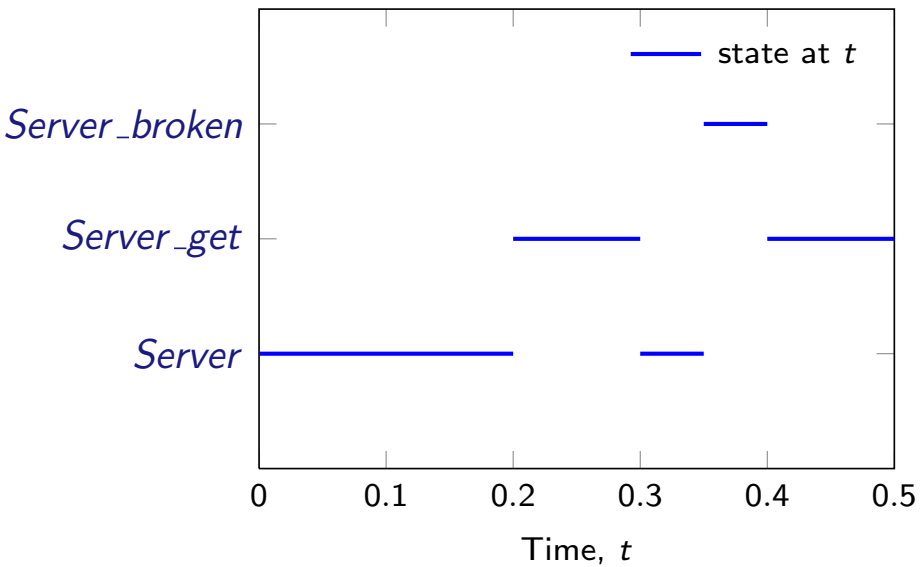


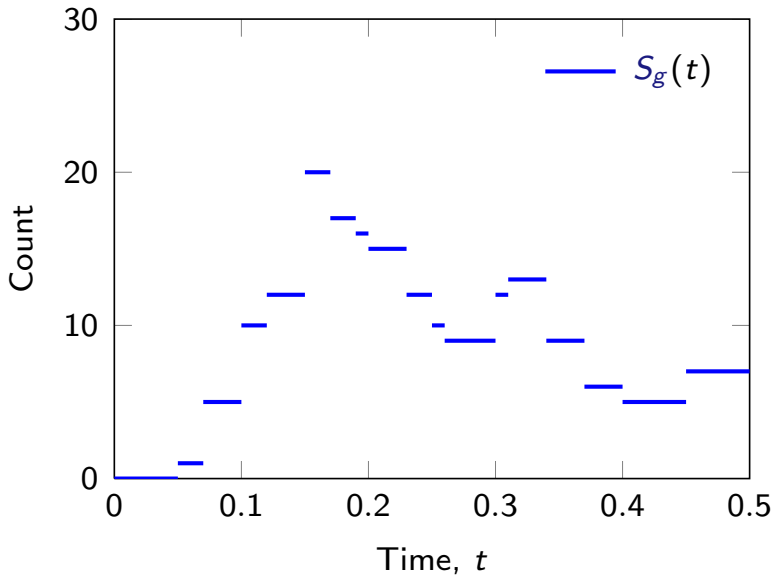


Accumulated rewards

How to analyse quantities accumulated over time, e.g. energy consumption?

Servers consume energy in the *Server_get* state





The total energy consumption is the process

$$\int_0^t S_g(u)\mathrm{d}u$$

ODEs – moments of rewards

Can extend the ODEs for moments of counts

$$\begin{array}{llll} \mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t & = & \cdots \\ \mathrm{d}\mathbb{E}[C_w(t)S_g(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_g(t)^3]/\mathrm{d}t & = & \cdots \\ & & \vdots & & & \vdots \end{array}$$

ODEs – moments of rewards

Can extend the ODEs for moments of counts

$$\begin{array}{llll} \mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t & = & \cdots \\ \mathrm{d}\mathbb{E}[C_w(t)S_g(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_g(t)^3]/\mathrm{d}t & = & \cdots \\ & \vdots & & & \vdots & \end{array}$$

with ODEs for the mean accumulated rewards

$$\begin{array}{lll} \mathrm{d}\mathbb{E}\left[\int_0^t S_g(u)\mathrm{d}u\right]/\mathrm{d}t & = & \mathbb{E}[S_g(t)] \\ \mathrm{d}\mathbb{E}\left[\int_0^t S(u)C(u)\mathrm{d}u\right]/\mathrm{d}t & = & \mathbb{E}[S(t)C(t)] \\ \vdots & & \end{array}$$

ODEs – moments of rewards

Can extend the ODEs for moments of counts

$$\begin{array}{lcllcl} \mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t & = & \cdots \\ \mathrm{d}\mathbb{E}[C_w(t)S_g(t)]/\mathrm{d}t & = & \cdots & \mathrm{d}\mathbb{E}[S_g(t)^3]/\mathrm{d}t & = & \cdots \\ & \vdots & & & \vdots & \end{array}$$

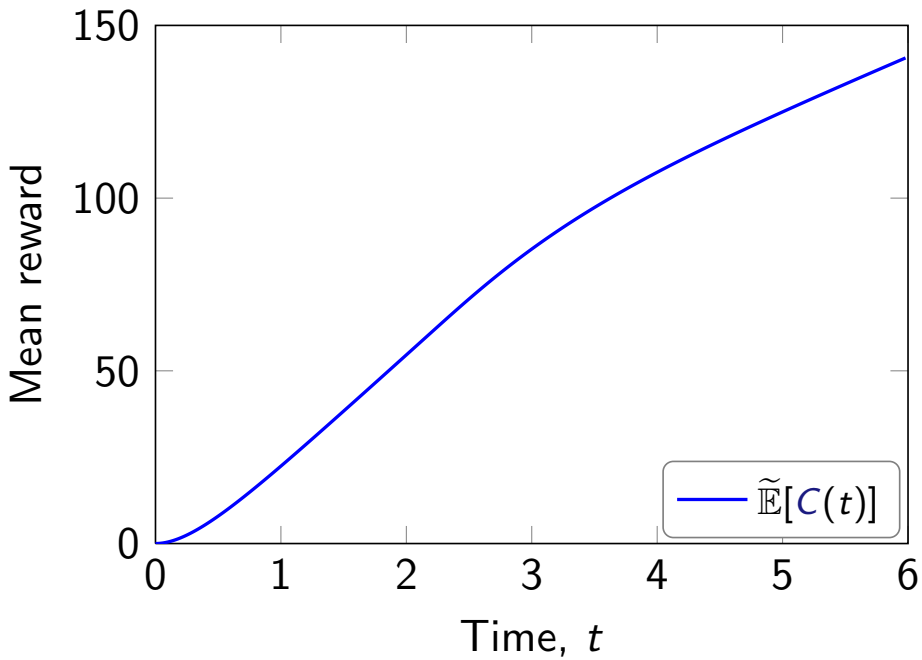
with ODEs for the mean accumulated rewards

$$\begin{array}{lcl} \mathrm{d}\mathbb{E}\left[\int_0^t S_g(u)\mathrm{d}u\right]/\mathrm{d}t & = & \mathbb{E}[S_g(t)] \\ \mathrm{d}\mathbb{E}\left[\int_0^t S(u)C(u)\mathrm{d}u\right]/\mathrm{d}t & = & \mathbb{E}[S(t)C(t)] \\ \vdots & & \end{array}$$

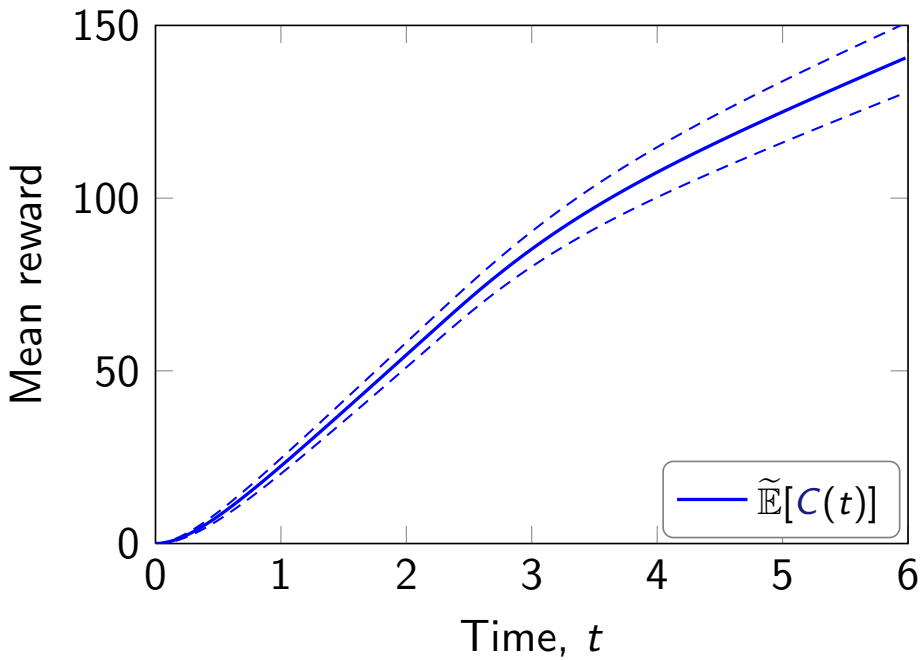
and ODEs for higher moments of accumulated rewards

$$\begin{array}{lcl} \mathrm{d}\mathbb{E}\left[\left(\int_0^t S_g(u)\mathrm{d}u\right)^2\right]/\mathrm{d}t & = & \cdots \\ \vdots & & \end{array}$$

ODEs – moments of rewards



ODEs – moments of rewards



GPA – Grouped PEPA Analyser

Why tool?

$$\mathrm{d}\mathbb{E}[C(t)S_g(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[C(t)S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C(t)C_w(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[C(t)S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C(t)C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[C(t)S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)^2])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)^2]/\mathrm{d}t+ = (-2.0)\cdot(\min((\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)^2])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_w(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)^2])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)^2])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S(t)C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)C_t(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_t(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_t(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)C_t(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_t(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)C_t(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)C_t(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_t(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_t(t)^2]/\mathrm{d}t+ = (2.0)\cdot(\min((\mathbb{E}[C_w(t)C_t(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_t(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)^2])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_w(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)^2])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)^2]/\mathrm{d}t+ = (-2.0)\cdot(\min((\mathbb{E}[C_w(t)^2])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)^2])\cdot(r_{data}),(\mathbb{E}[S_g(t)C_w(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t+ = (2.0)\cdot((\mathbb{E}[S(t)S_b(t)])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[S(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[S(t)S_b(t)])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[S(t)S_b(t)]/\mathrm{d}t+ = (\mathbb{E}[S_b(t)^2])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[S_b(t)^2]/\mathrm{d}t+ = (-2.0)\cdot((\mathbb{E}[S_b(t)^2])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t+ = (\mathbb{E}[S_b(t)])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[S_b(t)])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t+ = (\mathbb{E}[S_b(t)])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[S(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[S_b(t)])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[S_b(t)^2]/\mathrm{d}t+ = (\mathbb{E}[S_b(t)])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t+ = (\mathbb{E}[C(t)S_b(t)])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[C(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[C(t)S_b(t)])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = (\mathbb{E}[S_g(t)S_b(t)])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[S_g(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[S_g(t)S_b(t)])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[S(t)C_t(t)]/\mathrm{d}t+ = (\mathbb{E}[C_t(t)S_b(t)])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[C_t(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[C_t(t)S_b(t)])\cdot(r_{reset}))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = (\mathbb{E}[C_w(t)S_b(t)])\cdot(r_{reset})$$
$$\mathrm{d}\mathbb{E}[C_w(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[C_w(t)S_b(t)])\cdot(r_{reset}))$$

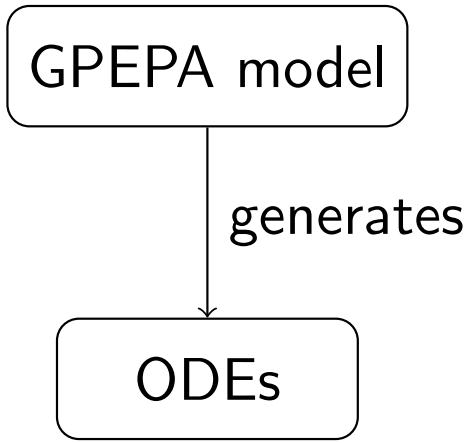
$$\mathrm{d}\mathbb{E}[S(t)C_t(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)C_t(t)])\cdot(r_{req}),(\mathbb{E}[S(t)C_t(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)C_t(t)])\cdot(r_{req}),(\mathbb{E}[S(t)C_t(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)C_t(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)C_t(t)])\cdot(r_{req}),(\mathbb{E}[S(t)C_t(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)C_t(t)])\cdot(r_{req}),(\mathbb{E}[S(t)C_t(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{req}),(\mathbb{E}[C_w(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_w(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{req}),(\mathbb{E}[C_w(t)S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)C_w(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{req}),(\mathbb{E}[C_w(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)^2]/\mathrm{d}t+ = (2.0)\cdot(\min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{req}),(\mathbb{E}[C_w(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t+ = (2.0)\cdot(\min((\mathbb{E}[C_w(t)S(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)S(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)S(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[S(t)C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)S(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S(t)S_b(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)S_b(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S_b(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)S_b(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S_b(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)S_b(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S_b(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_t(t)S_b(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)S_b(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)S_b(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_t(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)^2]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_w(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_t(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)^2]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[C_w(t)C_t(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data})))$$
$$\mathrm{d}\mathbb{E}[C_t(t)^2]/\mathrm{d}t+ = \min((\mathbb{E}[C_w(t)])\cdot(r_{data}),(\mathbb{E}[S_g(t)])\cdot(r_{data}))$$
$$\mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)C_w(t)])\cdot(r_{data}),(\mathbb{E}[C(t)S_g(t)])\cdot(r_{data}))$$

$$\mathrm{d}\mathbb{E}[C_t(t)^2]/\mathrm{d}t+ = (-2.0)\cdot((\mathbb{E}[C_t(t)^2])\cdot(r_{think}))$$
$$\mathrm{d}\mathbb{E}[C(t)C_w(t)]/\mathrm{d}t+ = (\mathbb{E}[C_w(t)C_t(t)])\cdot(r_{think})$$
$$\mathrm{d}\mathbb{E}[C_w(t)C_t(t)]/\mathrm{d}t+ = (-1.0)\cdot((\mathbb{E}[C_w(t)C_t(t)])\cdot(r_{think}))$$
$$\mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t+ = (-2.0)\cdot(\min((\mathbb{E}[C(t)S(t)])\cdot(r_{req}),(\mathbb{E}[S(t)^2])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)S(t)])\cdot(r_{req}),(\mathbb{E}[S(t)^2])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)S(t)])\cdot(r_{req}),(\mathbb{E}[S(t)^2])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)S(t)])\cdot(r_{req}),(\mathbb{E}[S(t)^2])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[S(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)S_b(t)])\cdot(r_{req}),(\mathbb{E}[S(t)S_b(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S_b(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)S_b(t)])\cdot(r_{req}),(\mathbb{E}[S(t)S_b(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)S_b(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)S_b(t)])\cdot(r_{req}),(\mathbb{E}[S(t)S_b(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S_b(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)S_b(t)])\cdot(r_{req}),(\mathbb{E}[S(t)S_b(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)]/\mathrm{d}t+ = (\min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C_w(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[S(t)^2]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C_w(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)^2]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)S_g(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_w(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)])\cdot(r_{req}),(\mathbb{E}[S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[C(t)^2])\cdot(r_{req}),(\mathbb{E}[C(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C(t)S_g(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)^2])\cdot(r_{req}),(\mathbb{E}[C(t)S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[C(t)^2]/\mathrm{d}t+ = (-2.0)\cdot(\min((\mathbb{E}[C(t)^2])\cdot(r_{req}),(\mathbb{E}[C(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C(t)C_w(t)]/\mathrm{d}t+ = \min((\mathbb{E}[C(t)^2])\cdot(r_{req}),(\mathbb{E}[C(t)S(t)])\cdot(r_{req}))$$
$$\mathrm{d}\mathbb{E}[S_g(t)S(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[S_g(t)C(t)])\cdot(r_{req}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)^2]/\mathrm{d}t+ = (2.0)\cdot(\min((\mathbb{E}[S_g(t)C(t)])\cdot(r_{req}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[C(t)S_g(t)]/\mathrm{d}t+ = (-1.0)\cdot(\min((\mathbb{E}[S_g(t)C(t)])\cdot(r_{req}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{req})))$$
$$\mathrm{d}\mathbb{E}[S_g(t)C_w(t)]/\mathrm{d}t+ = \min((\mathbb{E}[S_g(t)C(t)])\cdot(r_{req}),(\mathbb{E}[S_g(t)S(t)])\cdot(r_{req}))$$

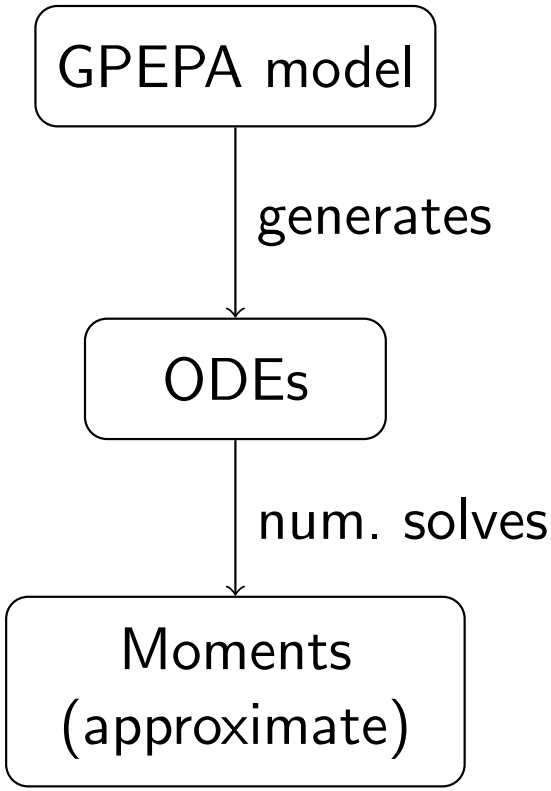
GPA

GPEPA model

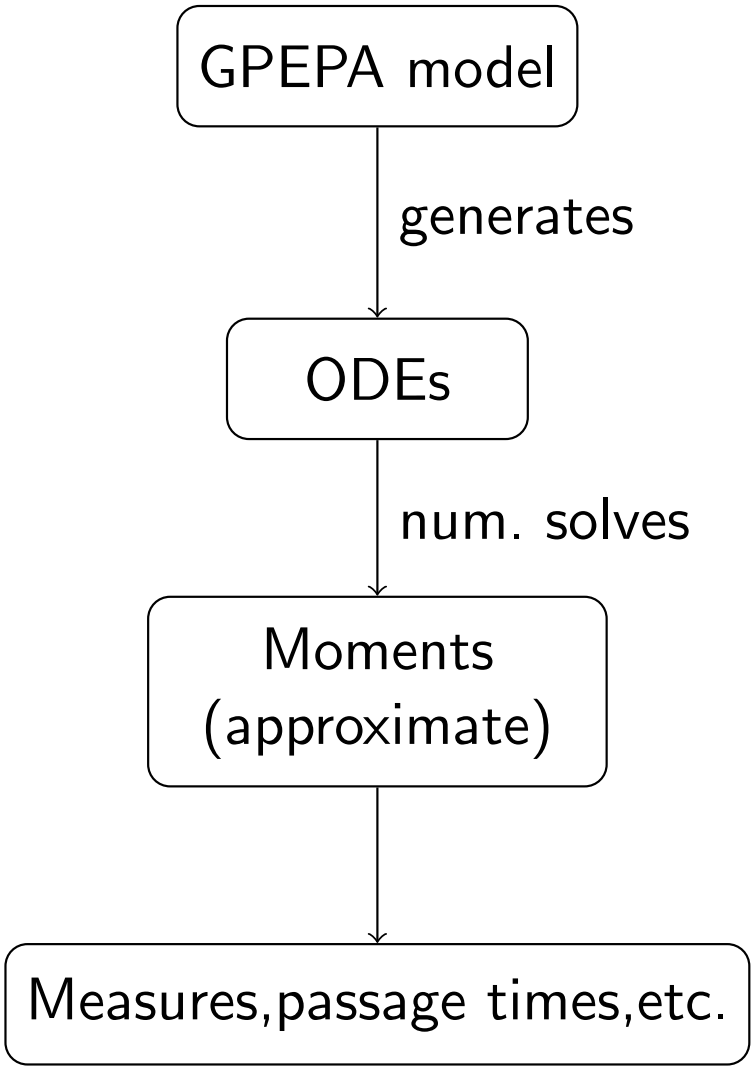
GPA



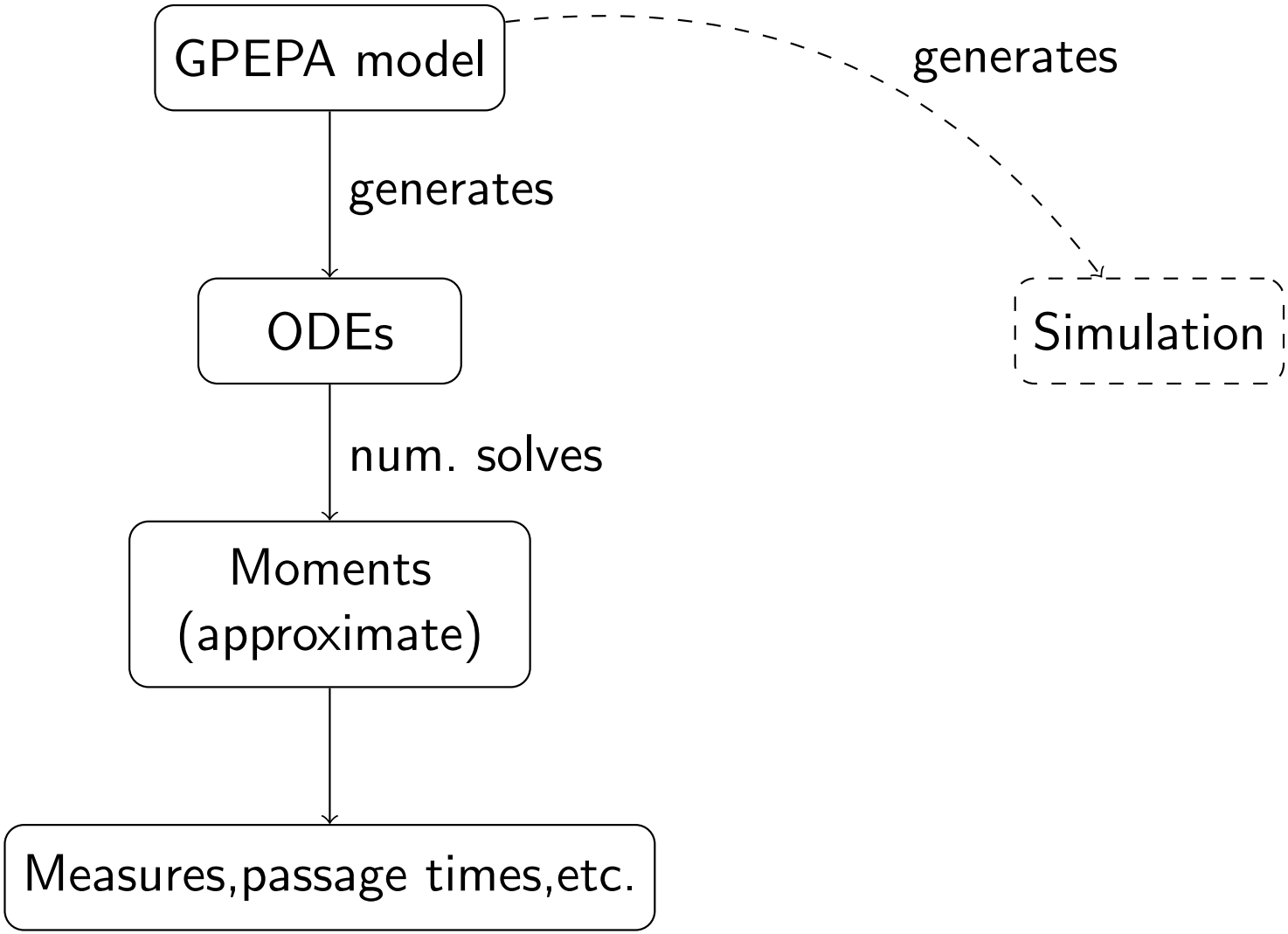
GPA


GPEPA model
generates
ODEs
num. solves
Moments
(approximate)

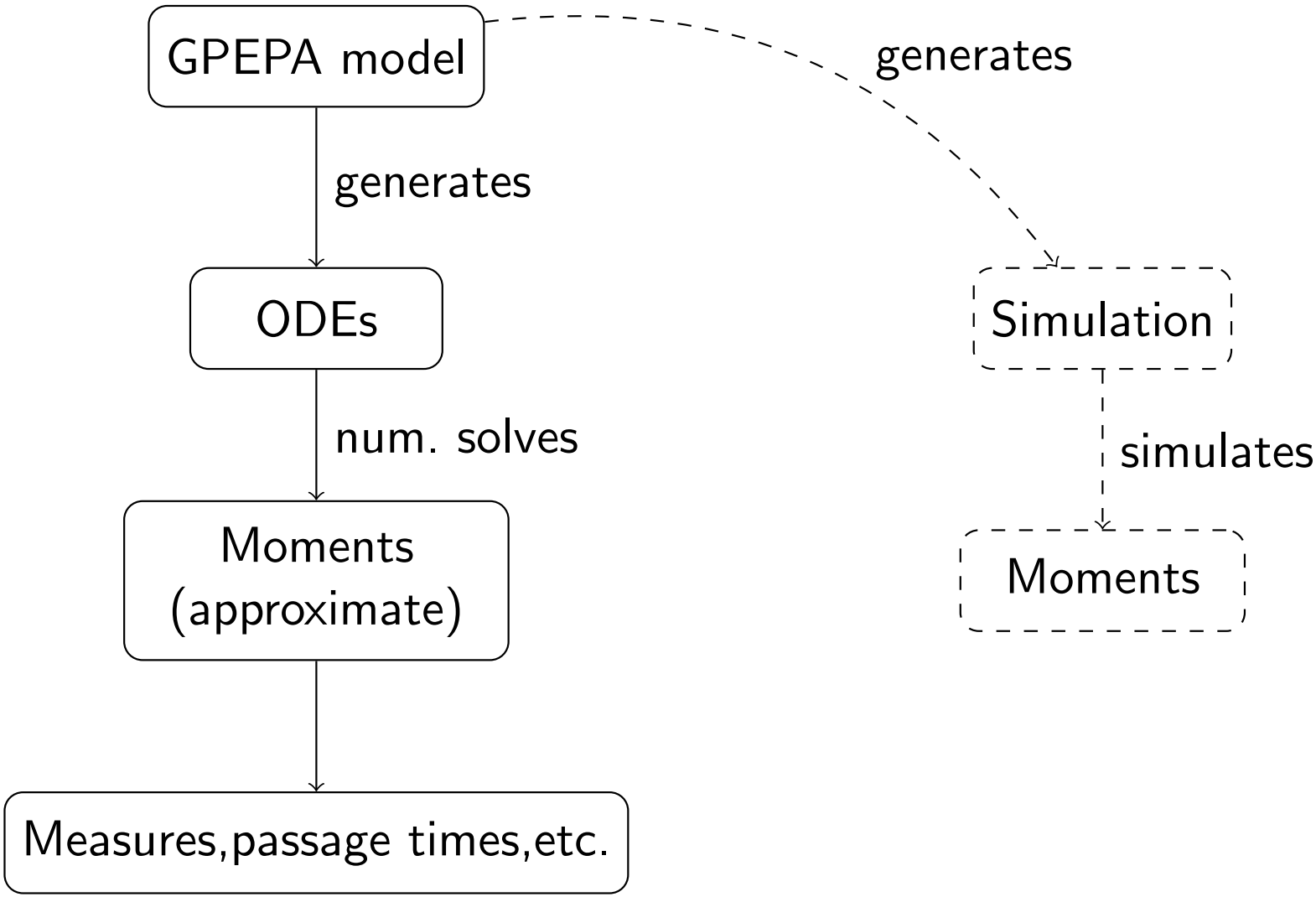
GPA


GPEPA model
generates
ODEs
num. solves
Moments
(approximate)
Measures,passage times,etc.

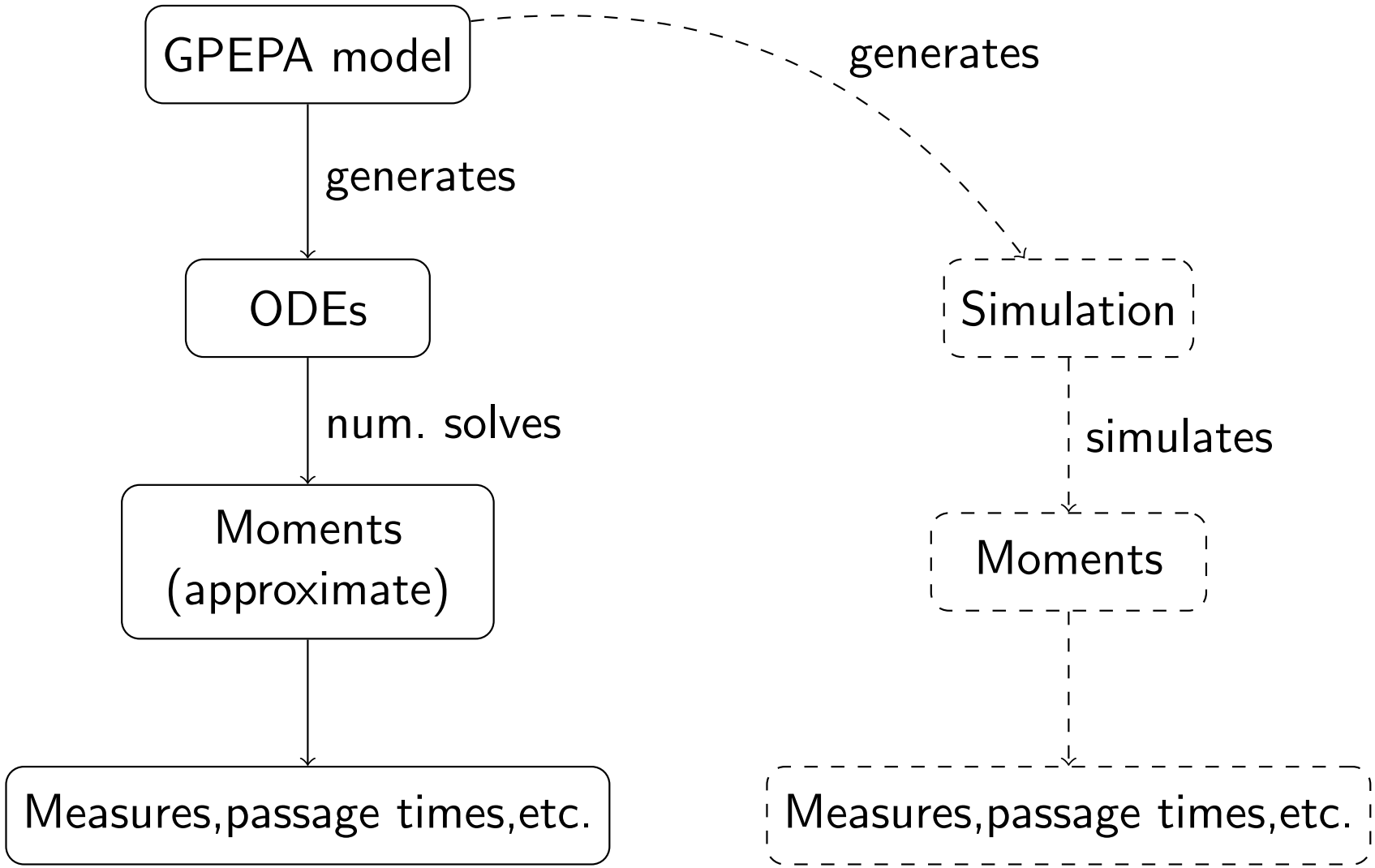
GPA


GPEPA model
generates
generates
ODEs
Simulation
num. solves
Moments
(approximate)
Measures,passage times,etc.

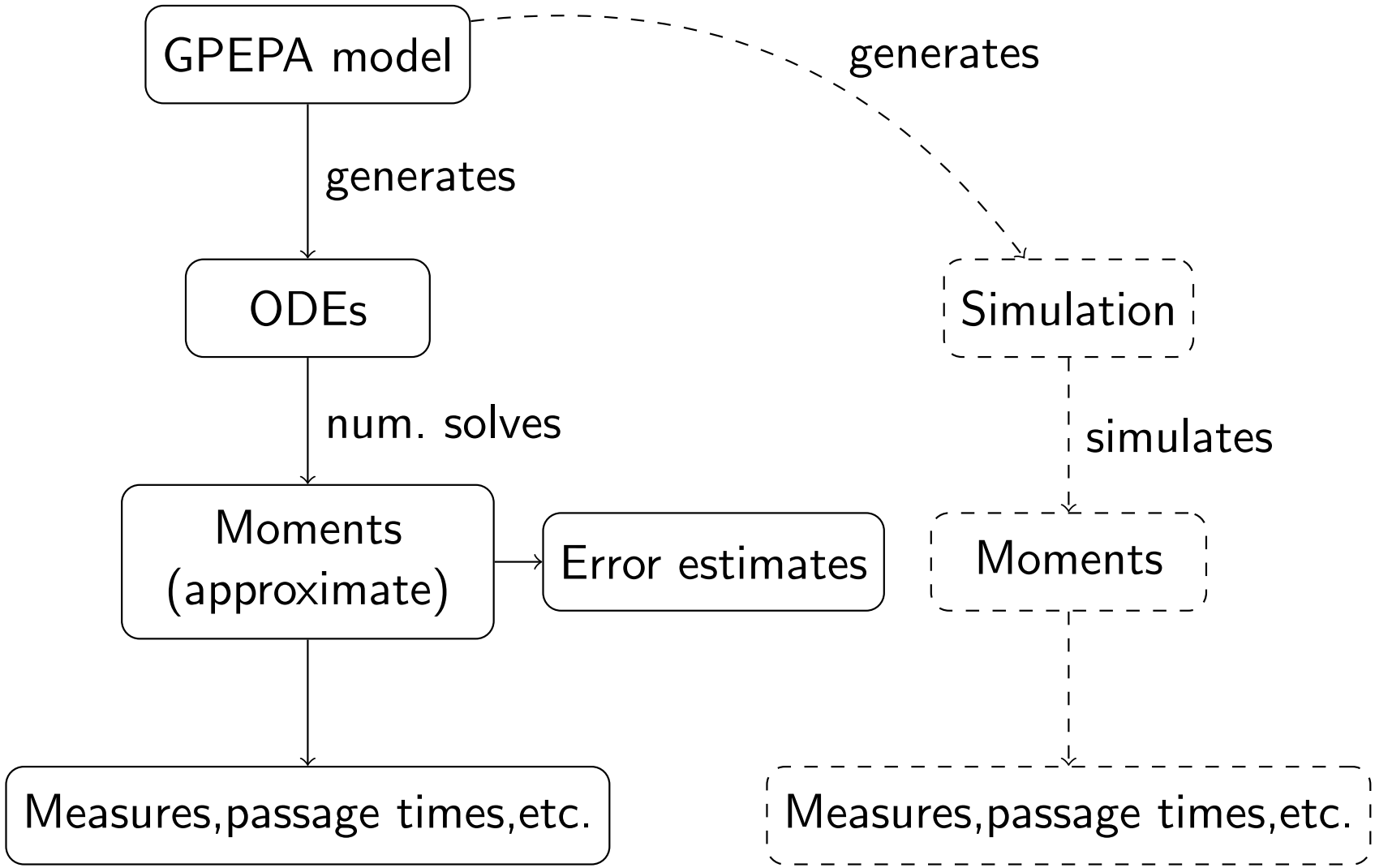
GPA


GPEPA model
generates
generates
ODEs
num. solves
Moments
(approximate)
Measures,passage times,etc.
Simulation
simulates
Moments

GPA


GPEPA model
generates
generates
ODEs
num. solves
Moments
(approximate)
Measures,passage times,etc.
Simulation
simulates
Moments
Measures,passage times,etc.

GPA


GPEPA model
generates
generates
ODEs
Simulation
num. solves
simulates
Moments
(approximate)
Error estimates
Moments
Measures,passage times,etc.
Measures,passage times,etc.

Grouped PEPA analyser

Convenient syntax

```
rreq   = 2.0;    rthink = 0.2;   ...
c = 100.0;       s = 50.0;

Client          = (request,rreq).Client_waiting;
Client_waiting  = (data,rdata).Client_think;
Client_think    = (think,rthink).Client;

Server          = (request,rreq).Server_get
                   + (break,rbreak).Server_broken;
Server_get      = (data,rdata).Server
Server_broken   = (reset,rreset).Server;

Clients{Client[c]}<request,data>Servers{Server[s]}
```

GPA – commands

- Analyses

```
odes(stopTime=5.0, stepSize=0.01, density=10){...}
```

```
simulation(stopTime=5.0,stepSize=0.01,repl.=1000){...}
```

```
comparison(odes(...){...},simulation(...){...}){...}
```

GPA – commands

- Analyses

```
odes(stopTime=5.0, stepSize=0.01, density=10){...}
```

```
simulation(stopTime=5.0,stepSize=0.01,repl.=1000){...}
```

```
comparison(odes(...){...},simulation(...){...}){...}
```

- Plot commands, counts specified with **Group**:Component

```
plot(E[Clients:Client],E[acc(Clients:Client)]);
plot(E[acc(Clients:Client) Servers:Server_get^2]);
plot(Var[Clients:Client]);
```

GPA – commands

- Analyses

```
odes(stopTime=5.0, stepSize=0.01, density=10){...}
```

```
simulation(stopTime=5.0,stepSize=0.01,repl.=1000){...}
```

```
comparison(odes(...){...},simulation(...){...}){...}
```

- Plot commands, counts specified with **Group**:Component

```
plot(E[Clients:Client],E[acc(Clients:Client)]);
plot(E[acc(Clients:Client) Servers:Server_get^2]);
plot(Var[Clients:Client]);
plot(E[Clients:Client]^2.0 + Var[Servers:Server]/s);
```

GPA – commands

- Analyses

```
odes(stopTime=5.0, stepSize=0.01, density=10){...}
```

```
simulation(stopTime=5.0,stepSize=0.01,repl.=1000){...}
```

```
comparison(odes(...){...},simulation(...){...}){...}
```

- Plot commands, counts specified with **Group**:Component

```
plot(E[Clients:Client],E[acc(Clients:Client)]);
plot(E[acc(Clients:Client) Servers:Server_get^2]);
plot(Var[Clients:Client]);
plot(E[Clients:Client]^2.0 + Var[Servers:Server]/s);
```

```
plotSwitchpoints(1);
```

GPA – passage times

Allows general PEPA components

```
NotPassed = (think,rthink).Passed;
Passed    = (think,rthink).Passed;

ObservedClient = Client<think>NotPassed;
```

GPA – passage times

Allows general PEPA components

```
NotPassed = (think,rthink).Passed;
Passed    = (think,rthink).Passed;

ObservedClient = Client<think>NotPassed;
```

For the CDF of first passage of a client

$$\mathbb{E}[C \underset{t}{\bowtie} P(t) + C_w \underset{t}{\bowtie} P(t) + C_t \underset{t}{\bowtie} P(t)]/c$$

GPA – passage times

Allows general PEPA components

```
NotPassed = (think,rthink).Passed;
Passed    = (think,rthink).Passed;

ObservedClient = Client<think>NotPassed;
```

For the CDF of first passage of a client

$$\mathbb{E}[C \underset{t}{\bowtie} P(t) + C_w \underset{t}{\bowtie} P(t) + C_t \underset{t}{\bowtie} P(t)]/c$$

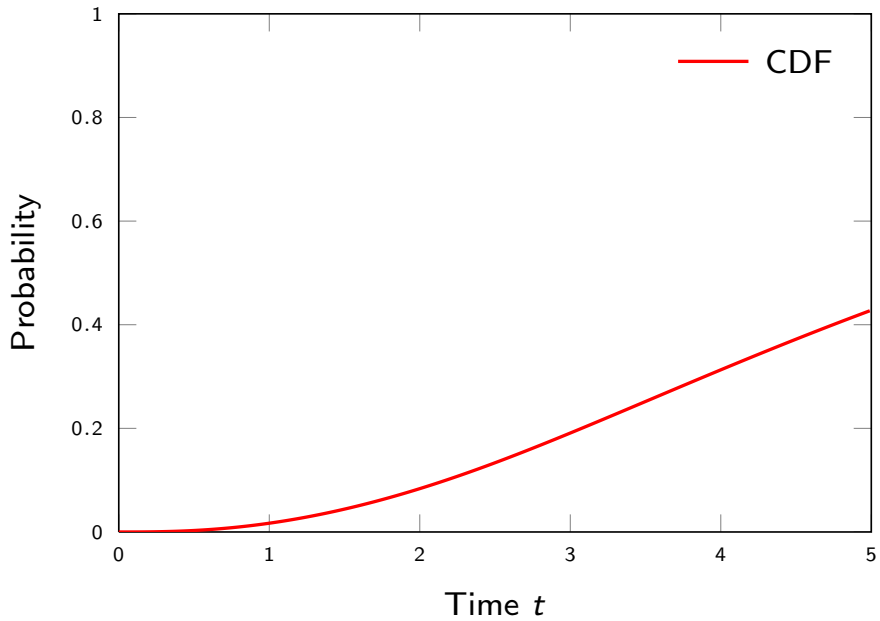
Can use command

```
plot(E[Clients:_<*>Passed]/c);
```

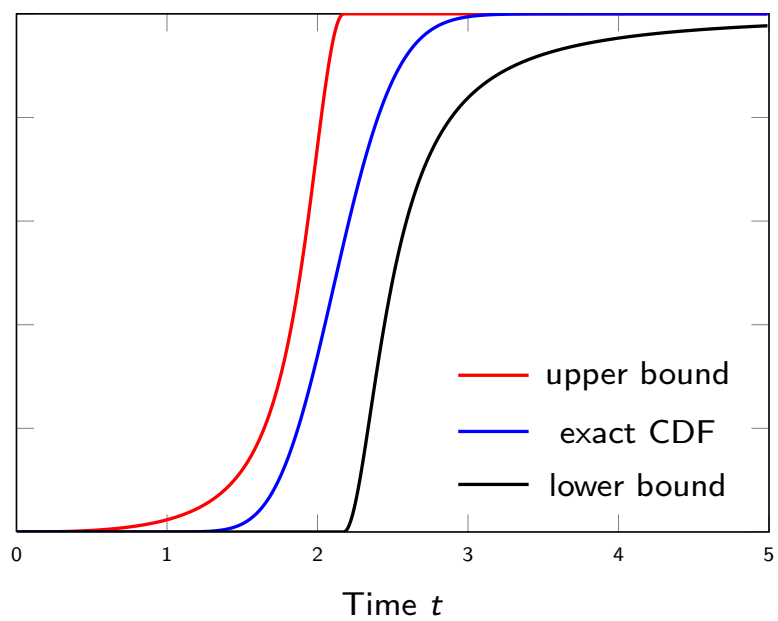
For an upper bound on the CDF of first passage of 1/10-th of clients

```
plot(Var[Clients:_<*>Passed]
/(Var[Clients:_<*>Passed]+(E[Clients:_<*>Passed]-c/10.0)^2.0));
```

GPA – passage times



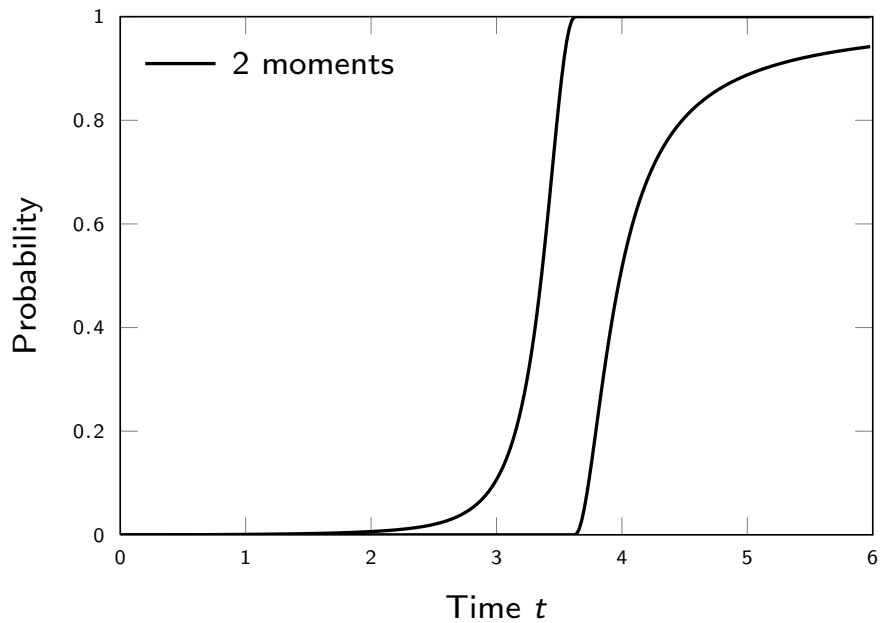
(a) Individual passage time for a client first passage



(b) Global passage time until $c/10$ first passages

GPA – completion times

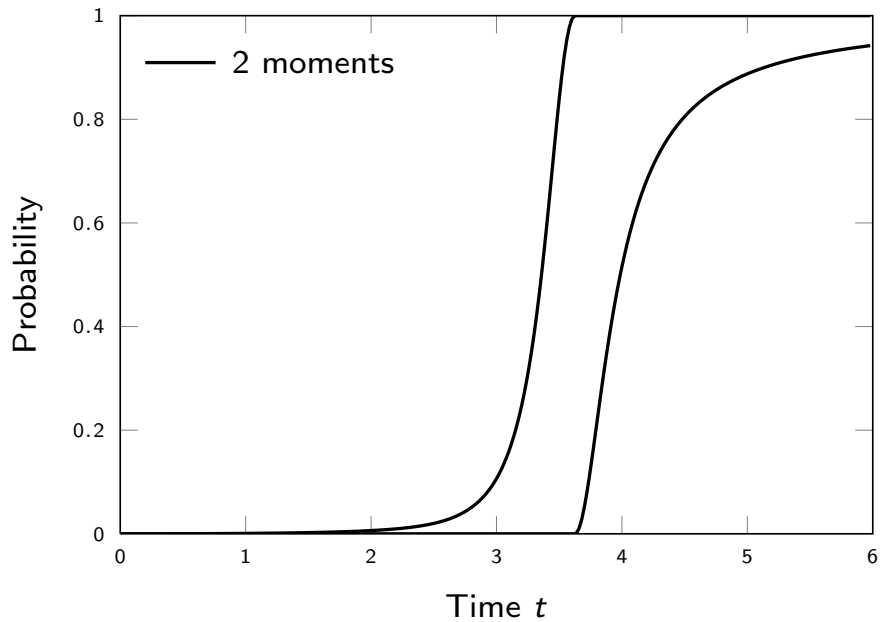
```
bounds(acc(Servers:Server_get),100.0,2);
```



completion time of $\int_0^t S_g(u)\mathrm{d}u$ reaching 100

GPA – completion times

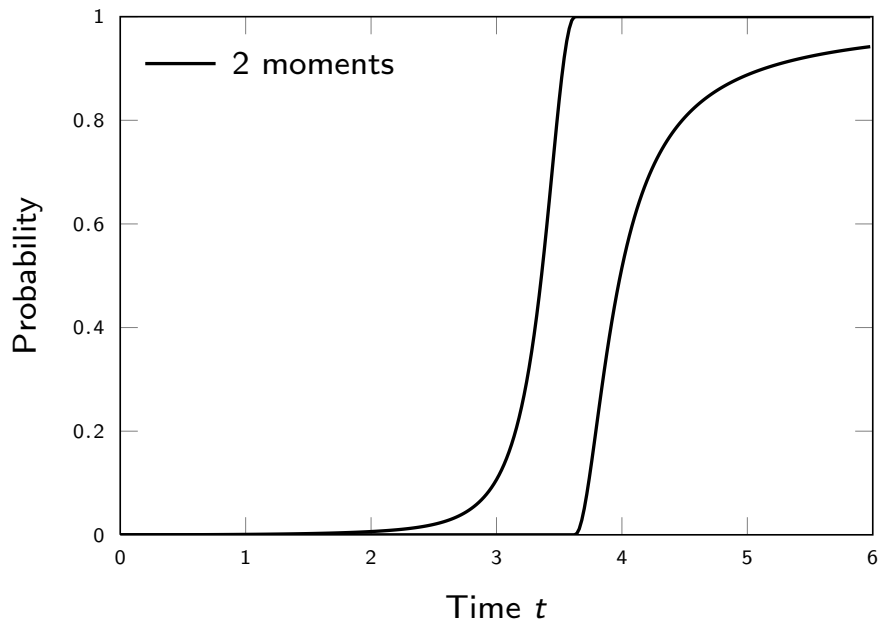
```
bounds(acc(Servers:Server_get),100.0,2,4);
```



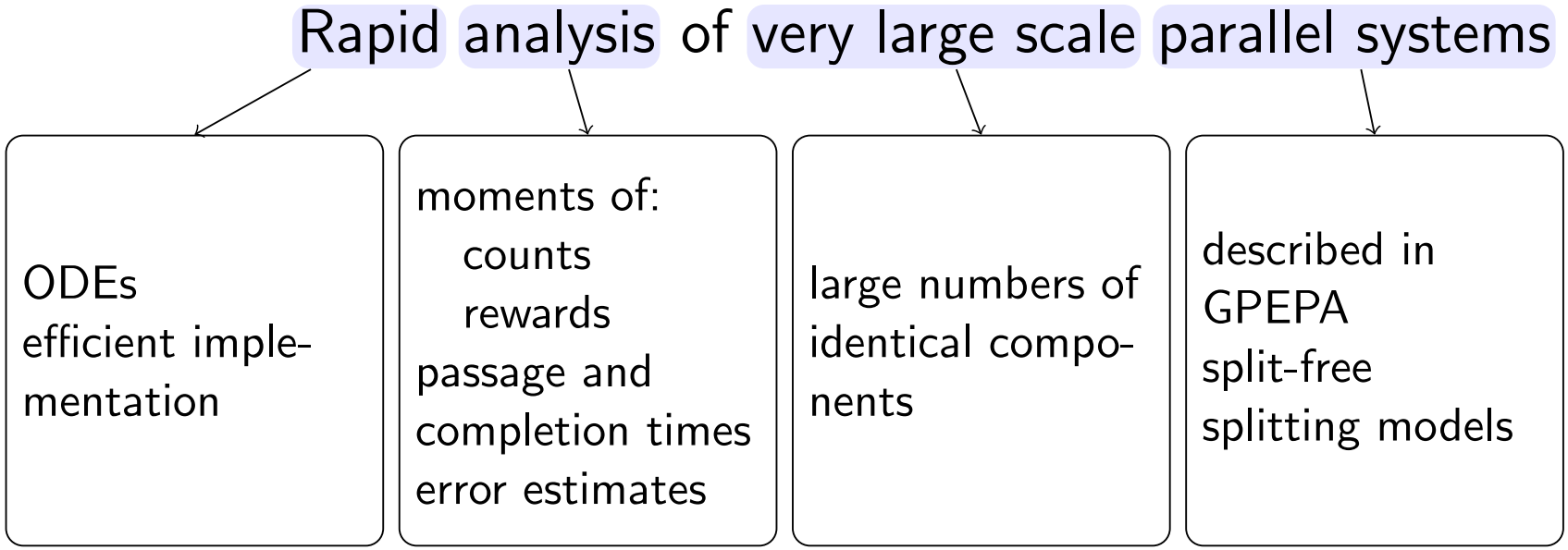
completion time of $\int_0^t S_g(u)\mathrm{d}u$ reaching 100

GPA – completion times

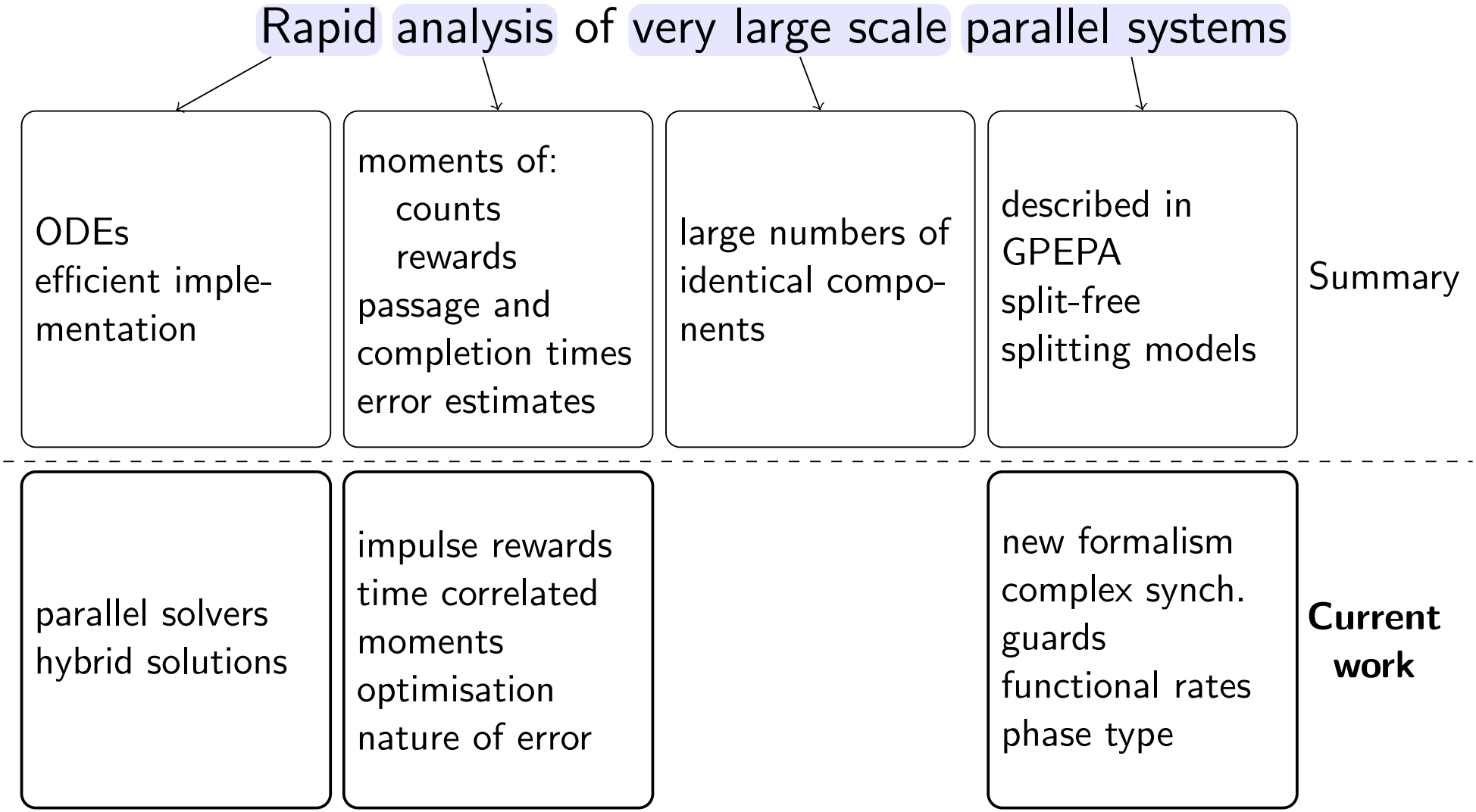
```
bounds(acc(Servers:Server_get),100.0,2,4,6);
```

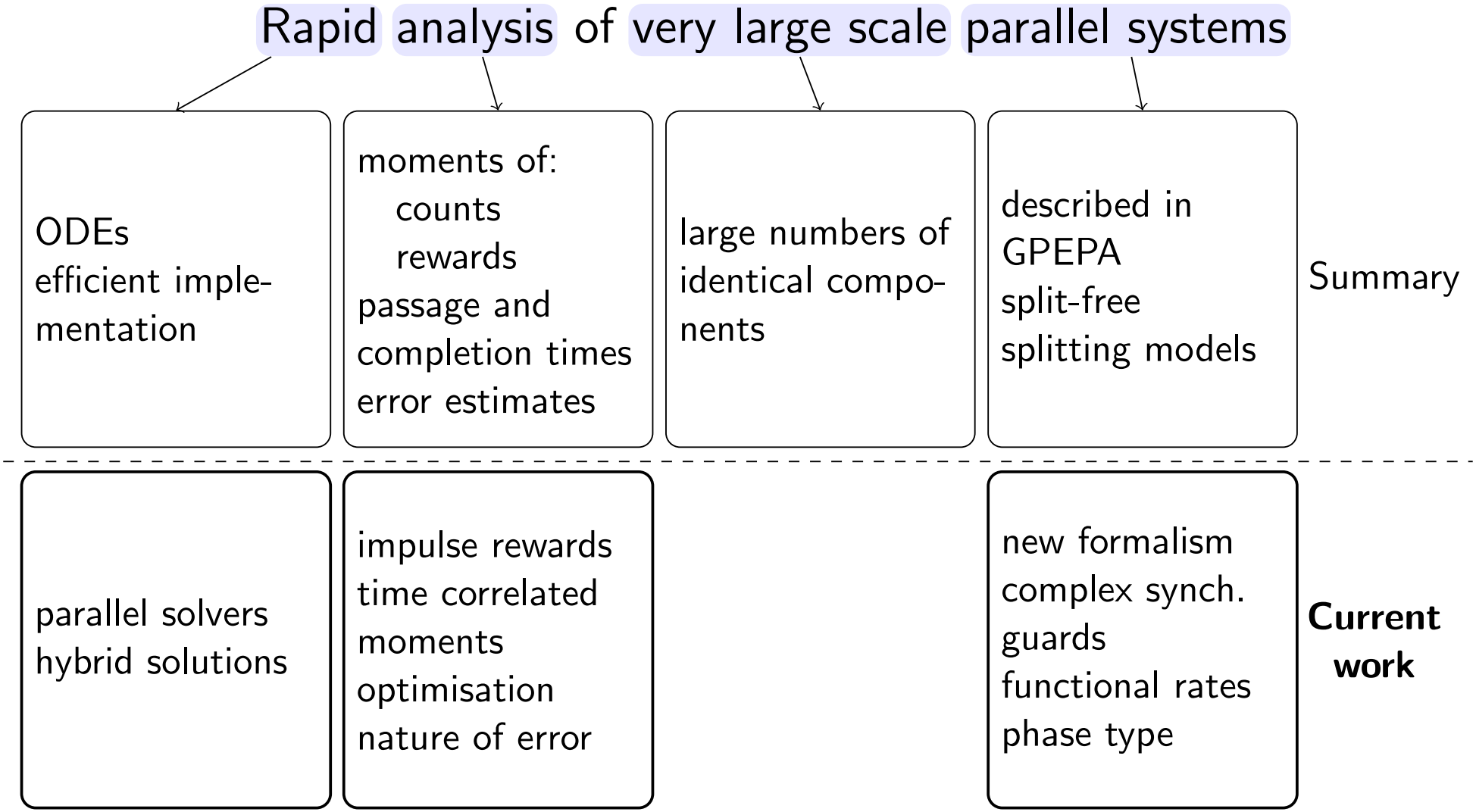


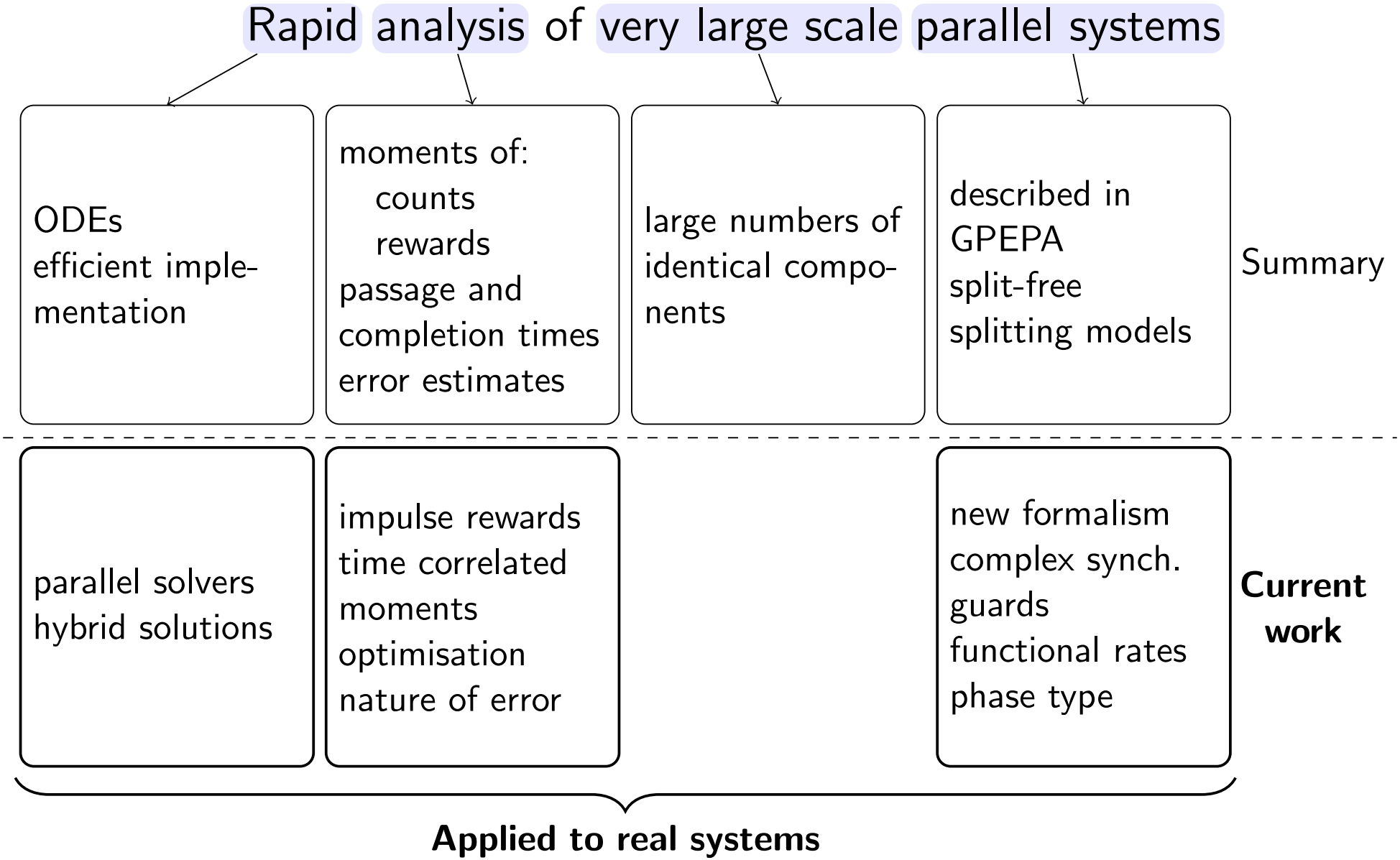
completion time of $\int_0^t S_g(u)\mathrm{d}u$ reaching 100


Rapid analysis of very large scale parallel systems
ODEs
efficient implementation
moments of:
counts
rewards
passage and
completion times
error estimates
large numbers of
identical components
described in
GPEPA
split-free
splitting models

Summary


Rapid analysis of very large scale parallel systems
ODEs
efficient implementation
moments of:
counts
rewards
passage and
completion times
error estimates
large numbers of
identical components
described in
GPEPA
split-free
splitting models
Summary
parallel solvers
hybrid solutions
impulse rewards
time correlated
moments
optimisation
nature of error
new formalism
complex synch.
guards
functional rates
phase type
Current
work


Rapid analysis of very large scale parallel systems
ODEs
efficient implementation
moments of:
counts
rewards
passage and
completion times
error estimates
large numbers of
identical components
described in
GPEPA
split-free
splitting models
Summary
parallel solvers
hybrid solutions
impulse rewards
time correlated
moments
optimisation
nature of error
new formalism
complex synch.
guards
functional rates
phase type
Current
work


Rapid analysis of very large scale parallel systems
ODEs
efficient implementation
moments of:
counts
rewards
passage and
completion times
error estimates
large numbers of
identical components
described in
GPEPA
split-free
splitting models
Summary
parallel solvers
hybrid solutions
impulse rewards
time correlated
moments
optimisation
nature of error
new formalism
complex synch.
guards
functional rates
phase type
Current work
Applied to real systems

GPA: Download for free

GPA tool[11]:

`http://code.google.com/p/gpanalyser/`

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "A new tool for the performance analysis of massively parallel computer systems". In: *Eighth Workshop on Quantitative Aspects of Programming Languages (QAPL 2010), March 27-28, 2010, Paphos, Cyprus*. Electronic Proceedings in Theoretical Computer Science. Mar. 2010. URL: http://pubs.doc.ic.ac.uk/pepa-ode-moments-tool/.

Fluid ODE generation using Population CTMCs

Populations CTMCs

A Population continuous time Markov chain (PCTMC) consists of a finite set of components $\{1, \ldots, N\}$, and a set T of transition classes.

Populations CTMCs

A Population continuous time Markov chain (PCTMC) consists of a finite set of components $\{1, \ldots, N\}$, and a set T of transition classes.

Each state in a PCTMC is expressed as an integer vector $\vec{X} = (X_1, \ldots, X_N) \in Z_N$

Populations CTMCs

A Population continuous time Markov chain (PCTMC) consists of a finite set of components $\{1, \ldots, N\}$, and a set T of transition classes.

Each state in a PCTMC is expressed as an integer vector $\vec{X} = (X_1, \ldots, X_N) \in Z_N$

X_i represents the current population level of a component i.

PCTMCs: Transition classes

A transition class $c = (r_c, \vec{e}_c) \in T$ describes a stochastic event

Event c: $\vec{X}(t) \to \vec{X}(t')$ at rate r_c

PCTMCs: Transition classes

A transition class $c = (r_c, \vec{e}_c) \in T$ describes a stochastic event

Event c: $\vec{X}(t) \to \vec{X}(t')$ at rate r_c

1. with **exponentially distributed duration** D at rate $r_c(\vec{X}(t))$ where $r_c : Z_N \to \mathbb{R}$ is a rate function

PCTMCs: Transition classes

A transition class $c = (r_c, \vec{e}_c) \in T$ describes a stochastic event

Event c: $\vec{X}(t) \to \vec{X}(t')$ at rate r_c

1. with exponentially distributed duration D at rate $r_c(\vec{X}(t))$ where $r_c : Z_N \to \mathbb{R}$ is a rate function
2. which changes the current population vector according to the **change vector** $\vec{e}_c$

PCTMCs: Transition classes

A transition class $c = (r_c, \vec{e}_c) \in T$ describes a stochastic event

Event c: $\vec{X}(t) \to \vec{X}(t')$ at rate r_c

1. with exponentially distributed duration D at rate $r_c(\vec{X}(t))$ where $r_c : Z_N \to \mathbb{R}$ is a rate function
2. which changes the current population vector according to the change vector $\vec{e}_c$

This gives us the following population dynamic formula:

Event c: $\vec{X}(t + D) = \vec{X}(t) + \vec{e}_c$ $D \sim \exp(r_c)$

PCTMCs: Chemical reactions

Similar to chemical reaction:

$$s_1 + \cdots + s_k \to t_1 + \cdots + t_l \qquad \text{at rate } r(\vec{X})$$

PCTMCs: Chemical reactions

Similar to chemical reaction:

$$s_1 + \cdots + s_k \to t_1 + \cdots + t_l \qquad \text{at rate } r(\vec{X})$$

Change vector for this reaction would involve:

$$\vec{e}_c = \{\underbrace{-1, \ldots -1}_{k}, \underbrace{1 \ldots, 1}_{l}, 0, \ldots, 0\}$$

PCTMCs: Mean dynamics

An important aspect of PCTMC models is that we can easily generate approximations to the evolution of the underlying stochastic process.[15]

[15] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

PCTMCs: Mean dynamics

An important aspect of PCTMC models is that we can easily generate approximations to the evolution of the underlying stochastic process.[15]

In particular, the equation for a mean of population $X_i(t)$ is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[X_i(t)] = \sum_{(r_j,\vec{e}_j)\in T} e_{ij} r_j(\vec{X}(t))$$

[15] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

ODE-based dynamics

More generally PCTMCs permit the derivation of moments of the underlying stochastic process, i.e. moments of population levels

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[M(\vec{X}(t))] = \mathbb{E}[f_M(\vec{X}(t))]$$

where $M(\vec{X})$ defines the moment to be calculated.

ODE-based dynamics

More generally PCTMCs permit the derivation of moments of the underlying stochastic process, i.e. moments of population levels

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[M(\vec{X}(t))] = \mathbb{E}[f_M(\vec{X}(t))]$$

where $M(\vec{X})$ defines the moment to be calculated.

- Mean of component 1: $M(\vec{X}) = X_1$
- 2nd moment of component 1: $M(\vec{X}) = X_1^2$
- 2nd joint moment of components 1 and 2: $M(\vec{X}) = X_1 X_2$

Higher moments

The higher moment function is defined as:[16]

$$f_M(\vec{X}(t)) = \sum_{c \in T} (M(\vec{X}(t) + \vec{e}_c) M(\vec{X}(t))) \, r_c(\vec{X}(t))$$

[16] Anton Stefanek. "Efficient Computation of Performance–Energy Trade-offs in Large Scale Markov Models". PhD thesis. Department of Computing, Imperial College London, 2013.

Higher moments

The higher moment function is defined as:[16]

$$f_M(\vec{X}(t)) = \sum_{c \in T} (M(\vec{X}(t) + \vec{e}_c) M(\vec{X}(t))) \, r_c(\vec{X}(t))$$

Key issue: achieving a closed set of equations with each quantity on right hand side of ODEs having a corresponding ODE.

[16] Anton Stefanek. "Efficient Computation of Performance–Energy Trade-offs in Large Scale Markov Models". PhD thesis. Department of Computing, Imperial College London, 2013.

Higher moments

The higher moment function is defined as:[16]

$$f_M(\vec{X}(t)) = \sum_{c \in T} (M(\vec{X}(t) + \vec{e}_c) M(\vec{X}(t))) \, r_c(\vec{X}(t))$$

Key issue: achieving a closed set of equations with each quantity on right hand side of ODEs having a corresponding ODE.

Leads to different dynamics: mean-field, mass action, min-closure, log-normal-closure

[16] Anton Stefanek. "Efficient Computation of Performance–Energy Trade-offs in Large Scale Markov Models". PhD thesis. Department of Computing, Imperial College London, 2013.

Worked example: GPEPA

$$Client \stackrel{def}{=} (req, r_{req}).Client\_waiting$$

$$Client\_waiting \stackrel{def}{=} (data, r_{data}).Client\_think$$

$$Client\_think \stackrel{def}{=} (think, r_{think}).Client$$

Worked example: GPEPA

$$
\begin{aligned}
\mathit{Client} &\stackrel{def}{=} (\mathit{req}, r_{\mathit{req}}).\mathit{Client\_waiting} \\
\mathit{Client\_waiting} &\stackrel{def}{=} (\mathit{data}, r_{\mathit{data}}).\mathit{Client\_think} \\
\mathit{Client\_think} &\stackrel{def}{=} (\mathit{think}, r_{\mathit{think}}).\mathit{Client}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Server} &\stackrel{def}{=} (\mathit{req}, r_{\mathit{req}}).\mathit{Server\_get} \\
&\qquad + (\mathit{break}, r_{\mathit{break}}).\mathit{Server\_broken} \\
\mathit{Server\_get} &\stackrel{def}{=} (\mathit{data}, r_{\mathit{data}}).\mathit{Server} \\
\mathit{Server\_broken} &\stackrel{def}{=} (\mathit{reset}, r_{\mathit{reset}}).\mathit{Server}
\end{aligned}
$$

Worked example: GPEPA

$$
\begin{aligned}
\mathit{Client} &\stackrel{def}{=} (\mathit{req}, r_{\mathit{req}}).\mathit{Client\_waiting} \\
\mathit{Client\_waiting} &\stackrel{def}{=} (\mathit{data}, r_{\mathit{data}}).\mathit{Client\_think} \\
\mathit{Client\_think} &\stackrel{def}{=} (\mathit{think}, r_{\mathit{think}}).\mathit{Client}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Server} &\stackrel{def}{=} (\mathit{req}, r_{\mathit{req}}).\mathit{Server\_get} \\
&\qquad + (\mathit{break}, r_{\mathit{break}}).\mathit{Server\_broken} \\
\mathit{Server\_get} &\stackrel{def}{=} (\mathit{data}, r_{\mathit{data}}).\mathit{Server} \\
\mathit{Server\_broken} &\stackrel{def}{=} (\mathit{reset}, r_{\mathit{reset}}).\mathit{Server}
\end{aligned}
$$

$$
\mathit{CS}(c, s) = \mathbf{Clients}\{\mathit{Client}[c]\} \underset{\{\mathit{req}, \mathit{data}\}}{\bowtie} \mathbf{Servers}\{\mathit{Server}[s]\}
$$

Worked example: GPEPA

$$Client \stackrel{def}{=} (req, r_{req}).Client\_waiting \qquad C(t)$$
$$Client\_waiting \stackrel{def}{=} (data, r_{data}).Client\_think$$
$$Client\_think \stackrel{def}{=} (think, r_{think}).Client$$

$$Server \stackrel{def}{=} (req, r_{req}).Server\_get + (break, r_{break}).Server\_broken$$
$$Server\_get \stackrel{def}{=} (data, r_{data}).Server$$
$$Server\_broken \stackrel{def}{=} (reset, r_{reset}).Server$$

$$CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}$$

Worked example: GPEPA

$$\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client_waiting && C(t)\\
Client_waiting &\stackrel{def}{=} (data, r_{data}).Client_think && C_w(t)\\
Client_think &\stackrel{def}{=} (think, r_{think}).Client
\end{aligned}$$

$$\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server_get\\
&\qquad + (break, r_{break}).Server_broken\\
Server_get &\stackrel{def}{=} (data, r_{data}).Server\\
Server_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}$$

$$CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}$$

Worked example: GPEPA

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get\\
&\qquad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

Worked example: GPEPA

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get && S(t)\\
&\quad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

Worked example: GPEPA

$$
\begin{aligned}
Client &\stackrel{def}{=} (req, r_{req}).Client\_waiting && C(t)\\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think && C_w(t)\\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
Server &\stackrel{def}{=} (req, r_{req}).Server\_get && S(t)\\
&\quad + (break, r_{break}).Server\_broken\\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server && S_g(t)\\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{Client[c]\} \underset{\{req, data\}}{\bowtie} \mathbf{Servers}\{Server[s]\}
$$

Worked example: GPEPA

$$
\begin{aligned}
\mathit{Client} &\stackrel{def}{=} (\mathit{req}, r_{\mathit{req}}).\mathit{Client\_waiting} && C(t)\\
\mathit{Client\_waiting} &\stackrel{def}{=} (\mathit{data}, r_{\mathit{data}}).\mathit{Client\_think} && C_w(t)\\
\mathit{Client\_think} &\stackrel{def}{=} (\mathit{think}, r_{\mathit{think}}).\mathit{Client} && C_t(t)
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Server} &\stackrel{def}{=} (\mathit{req}, r_{\mathit{req}}).\mathit{Server\_get} && S(t)\\
&\quad + (\mathit{break}, r_{\mathit{break}}).\mathit{Server\_broken}\\
\mathit{Server\_get} &\stackrel{def}{=} (\mathit{data}, r_{\mathit{data}}).\mathit{Server} && S_g(t)\\
\mathit{Server\_broken} &\stackrel{def}{=} (\mathit{reset}, r_{\mathit{reset}}).\mathit{Server} && S_b(t)
\end{aligned}
$$

$$
CS(c, s) = \mathbf{Clients}\{\mathit{Client}[c]\} \underset{\{\mathit{req},\mathit{data}\}}{\bowtie} \mathbf{Servers}\{\mathit{Server}[s]\}
$$

Worked example: PCTMC

In total, there are 5 transition classes:

$$
\begin{aligned}
\mathit{req} &: \\
\mathit{data} &: \\
\mathit{think} &: \\
\mathit{break} &: \\
\mathit{reset} &:
\end{aligned}
$$

Worked example: PCTMC

In total, there are 5 transition classes:

$$\begin{aligned} \mathit{req}: &\quad C(t) + S(t) \to C_w(t) + S_g(t) \quad \text{at } r_{\mathit{req}} \cdot \min(C(t), S(t)) \\ \mathit{data}: & \\ \mathit{think}: & \\ \mathit{break}: & \\ \mathit{reset}: & \end{aligned}$$

Worked example: PCTMC

In total, there are 5 transition classes:

$$
\begin{aligned}
\mathit{req}: &\quad C(t) + S(t) \to C_w(t) + S_g(t) \quad \text{at } r_{\mathit{req}} \cdot \min(C(t), S(t)) \\
\mathit{data}: &\quad C_w(t) + S_g(t) \to C_t(t) + S(t) \quad \text{at } r_{\mathit{data}} \cdot \min(C_w(t), S_g(t)) \\
\mathit{think}: & \\
\mathit{break}: & \\
\mathit{reset}: &
\end{aligned}
$$

Worked example: PCTMC

In total, there are 5 transition classes:

$$
\begin{aligned}
\mathit{req}: &\quad C(t) + S(t) \to C_w(t) + S_g(t) \quad \text{at } r_{req} \cdot \min(C(t), S(t)) \\
\mathit{data}: &\quad C_w(t) + S_g(t) \to C_t(t) + S(t) \quad \text{at } r_{data} \cdot \min(C_w(t), S_g(t)) \\
\mathit{think}: &\quad C_t(t) \to C(t) \quad \text{at } r_{think} \cdot C_t(t) \\
\mathit{break}: & \\
\mathit{reset}: &
\end{aligned}
$$

Worked example: PCTMC

In total, there are 5 transition classes:

$$
\begin{aligned}
\mathit{req}: &\quad C(t) + S(t) \to C_w(t) + S_g(t) \quad \text{at } r_{req} \cdot \min(C(t), S(t)) \\
\mathit{data}: &\quad C_w(t) + S_g(t) \to C_t(t) + S(t) \quad \text{at } r_{data} \cdot \min(C_w(t), S_g(t)) \\
\mathit{think}: &\quad C_t(t) \to C(t) \quad \text{at } r_{think} \cdot C_t(t) \\
\mathit{break}: &\quad S(t) \to S_b(t) \quad \text{at } r_{break} \cdot S(t) \\
\mathit{reset}: &
\end{aligned}
$$

Worked example: PCTMC

In total, there are 5 transition classes:

$$
\begin{aligned}
\mathit{req}: &\quad C(t) + S(t) \to C_w(t) + S_g(t) \quad \text{at } r_{req} \cdot \min(C(t), S(t)) \\
\mathit{data}: &\quad C_w(t) + S_g(t) \to C_t(t) + S(t) \quad \text{at } r_{data} \cdot \min(C_w(t), S_g(t)) \\
\mathit{think}: &\quad C_t(t) \to C(t) \quad \text{at } r_{think} \cdot C_t(t) \\
\mathit{break}: &\quad S(t) \to S_b(t) \quad \text{at } r_{break} \cdot S(t) \\
\mathit{reset}: &\quad S_b(t) \to S(t) \quad \text{at } r_{reset} \cdot S_b(t)
\end{aligned}
$$

Worked example: PCTMC

In total, there are 5 transition classes:

$$
\begin{aligned}
\mathit{req}: &\quad C(t) + S(t) \to C_w(t) + S_g(t) \quad \text{at } r_{req} \cdot \min(C(t), S(t)) \\
\mathit{data}: &\quad C_w(t) + S_g(t) \to C_t(t) + S(t) \quad \text{at } r_{data} \cdot \min(C_w(t), S_g(t)) \\
\mathit{think}: &\quad C_t(t) \to C(t) \quad \text{at } r_{think} \cdot C_t(t) \\
\mathit{break}: &\quad S(t) \to S_b(t) \quad \text{at } r_{break} \cdot S(t) \\
\mathit{reset}: &\quad S_b(t) \to S(t) \quad \text{at } r_{reset} \cdot S_b(t)
\end{aligned}
$$

Then apply PCTMC ODE generation rules to get a fluid GPEPA model.

Even more exciting fluid analysis

Scalable passage-time analysis

Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

"file should be transferred within 2 seconds, 95% of the time"

[7] Richard A. Hayden, Anton Stefanek, and Jeremy T. Bradley. "Fluid computation of passage time distributions in large Markov models". In: *Theoretical Computer Science* 413.1 (2012), pp. 106–141. DOI: 10.1016/j.tcs.2011.07.017.

[8] Richard A. Hayden, Jeremy T. Bradley, and Allan Clark. "Performance Specification and Evaluation with Unified Stochastic Probes and Fluid Analysis". In: *IEEE Transactions on Software Engineering* 99.PrePrints (2012). DOI: 10.1109/TSE.2012.1.

Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

"connection should be established within 0.25 seconds, 99% of the time"

Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  "connection should be established within 0.25 seconds, 99% of the time"

- We consider two classes of passage-time query:

Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  "connection should be established within 0.25 seconds, 99% of the time"

- We consider two classes of passage-time query:
  - Individual passage times: track the time taken for an individual to complete a task

Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  "connection should be established within 0.25 seconds, 99% of the time"

- We consider two classes of passage-time query:
  - Individual passage times: track the time taken for an individual to complete a task
  - Global passage times: track the time taken for all of a large number of individuals to complete a task

Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  "connection should be established within 0.25 seconds, 99% of the time"

- We consider two classes of passage-time query:
  - Individual passage times: track the time taken for an individual to complete a task
    - *Direct approximation to the entire CDF*
  - Global passage times: track the time taken for all of a large number of individuals to complete a task

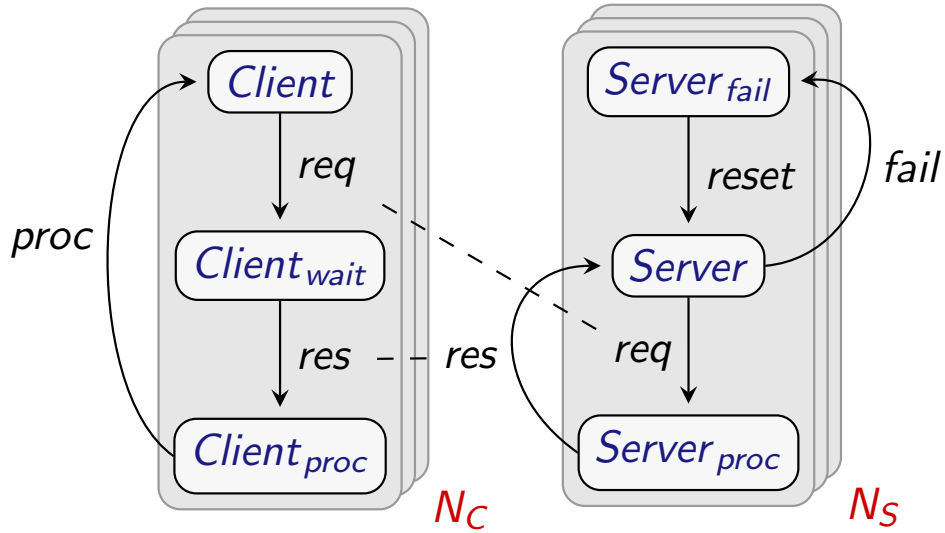
Scalable passage-time analysis

- Passage-time distributions are key for specifying service level agreements (SLAs), e.g.:

  "connection should be established within 0.25 seconds, 99% of the time"

- We consider two classes of passage-time query:
  - Individual passage times: track the time taken for an individual to complete a task
    - *Direct approximation to the entire CDF*
  - Global passage times: track the time taken for all of a large number of individuals to complete a task
    - *Moment-derived bounds on CDF*

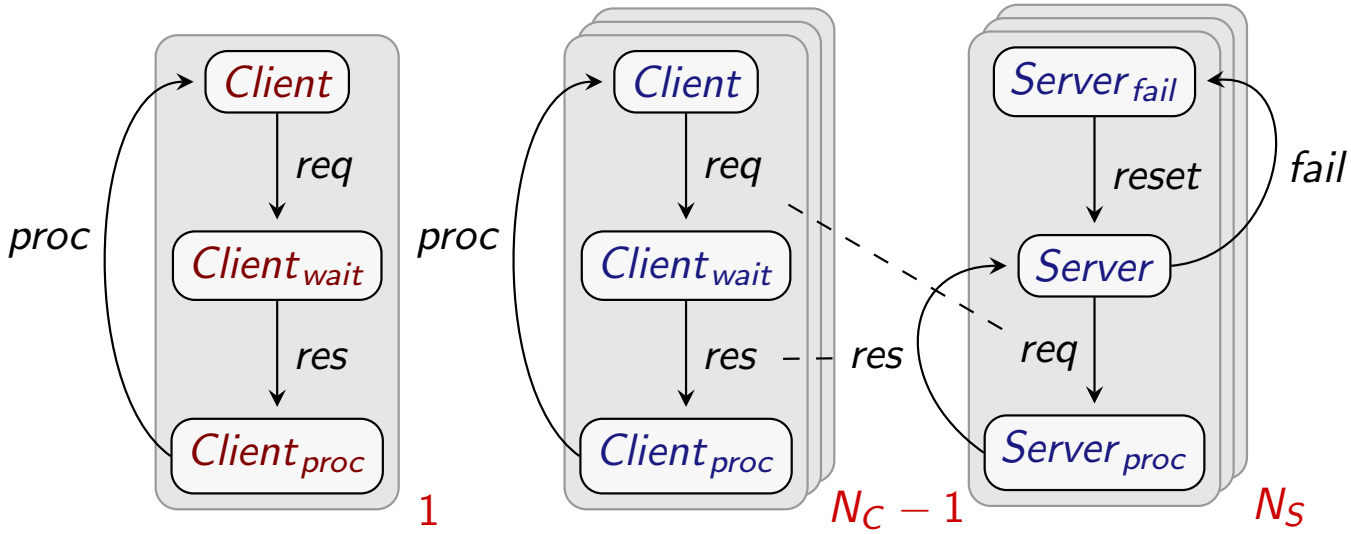
Individual passage times



How long does it take a single client to make a request, receive a response and process it?

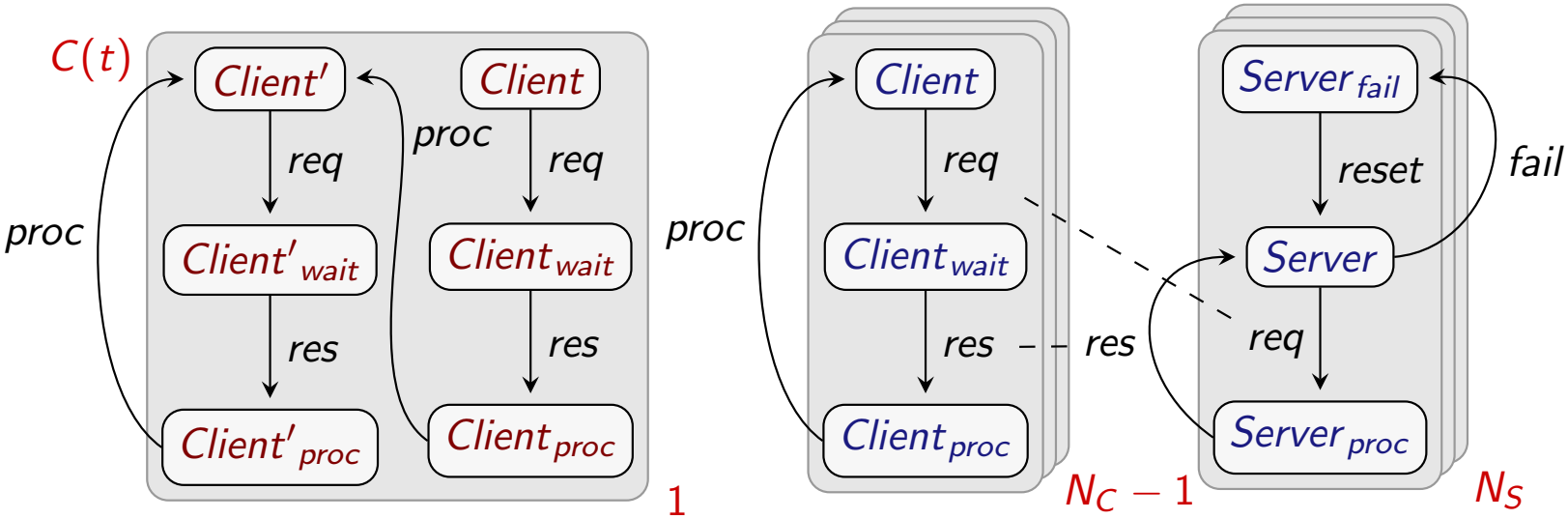
[7] Richard A. Hayden, Anton Stefanek, and Jeremy T. Bradley. "Fluid computation of passage time distributions in large Markov models". In: *Theoretical Computer Science* 413.1 (2012), pp. 106–141. DOI: 10.1016/j.tcs.2011.07.017.

Individual passage times



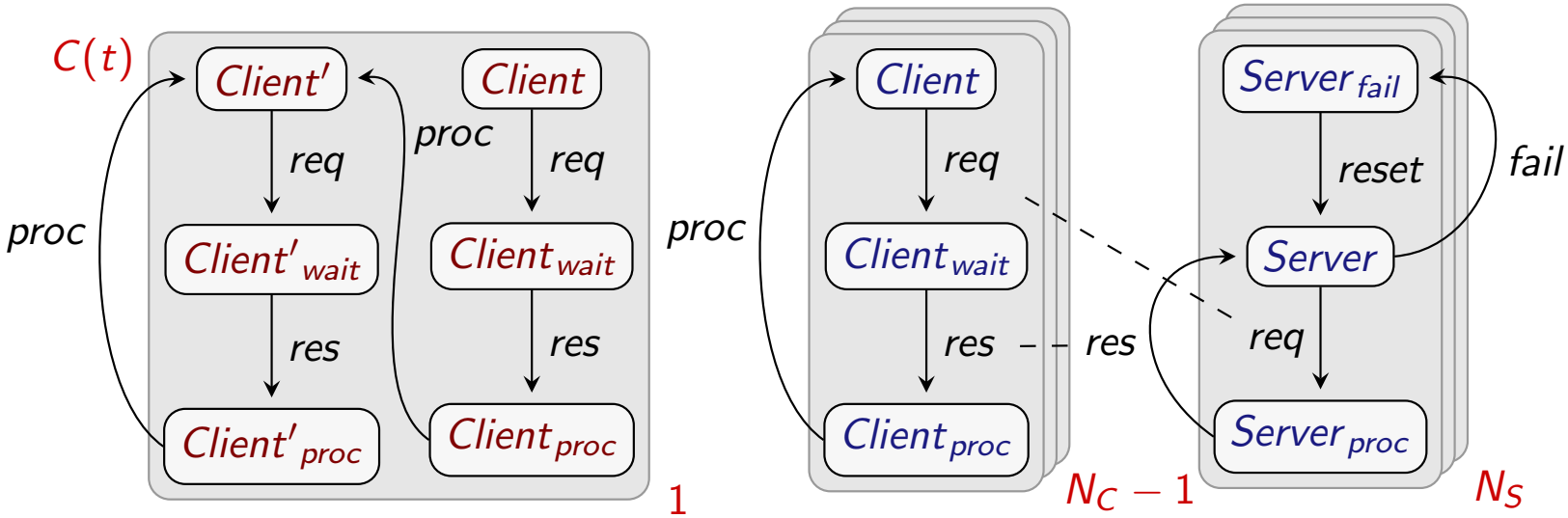
How long does it take a single client to make a request, receive a response and process it?

Individual passage times



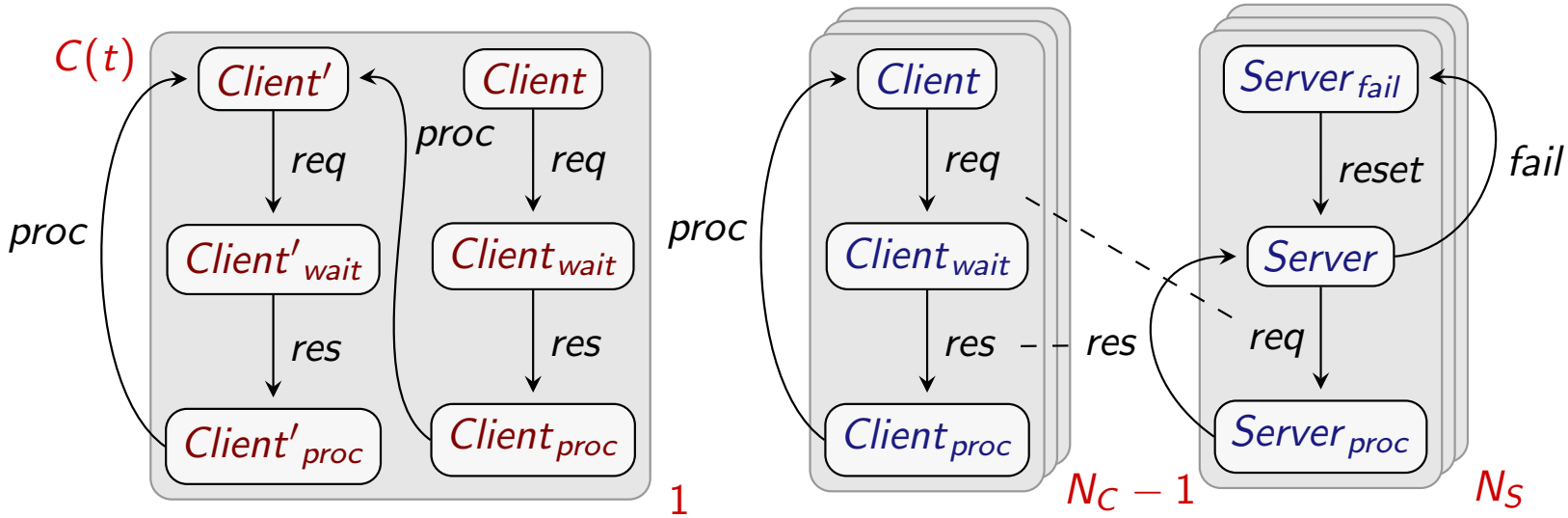
How long does it take a single client to make a request, receive a response and process it?

Individual passage times



$T := \inf\{t \geq 0 : C(t) = \mathit{Client}'\}$, given that $C(0) = \mathit{Client}$

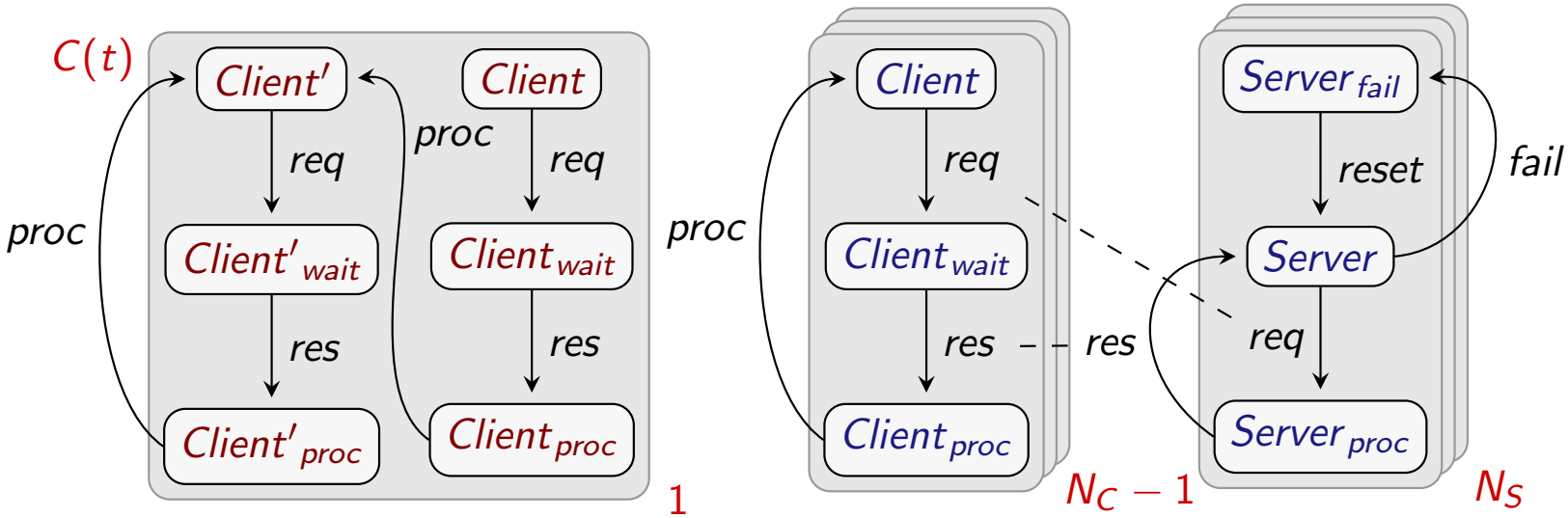
Individual passage times



$T := \inf\{t \geq 0 : C(t) = \mathit{Client}'\}$, given that $C(0) = \mathit{Client}$

$$\mathbb{P}\{T \leq t\} = \mathbb{P}\{C(t) \in \{\mathit{Client}', \mathit{Client}'_{\mathit{wait}}, \mathit{Client}'_{\mathit{proc}}\}\}$$

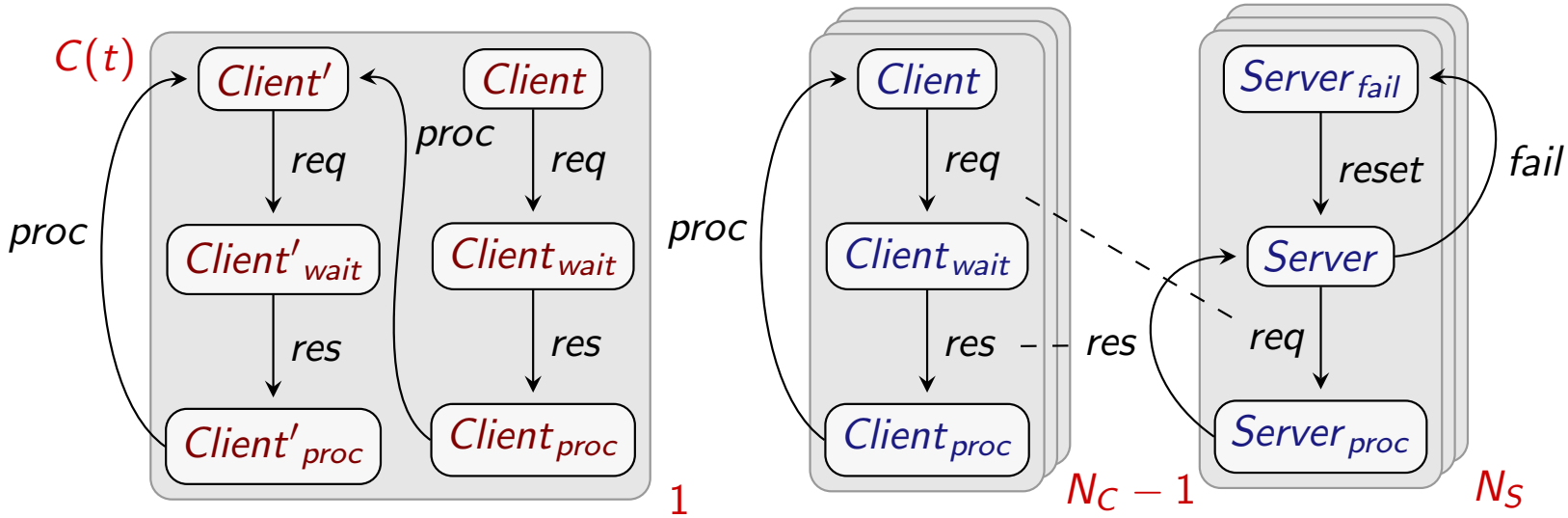
Individual passage times



$T := \inf\{t \geq 0 : C(t) = \mathit{Client}'\}$, given that $C(0) = \mathit{Client}$

$$\begin{aligned}\mathbb{P}\{T \leq t\} &= \mathbb{P}\{C(t) \in \{\mathit{Client}', \mathit{Client}'_{\mathit{wait}}, \mathit{Client}'_{\mathit{proc}}\}\} \\ &= \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'\}}] + \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'_{\mathit{wait}}\}}] + \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'_{\mathit{proc}}\}}]\end{aligned}$$

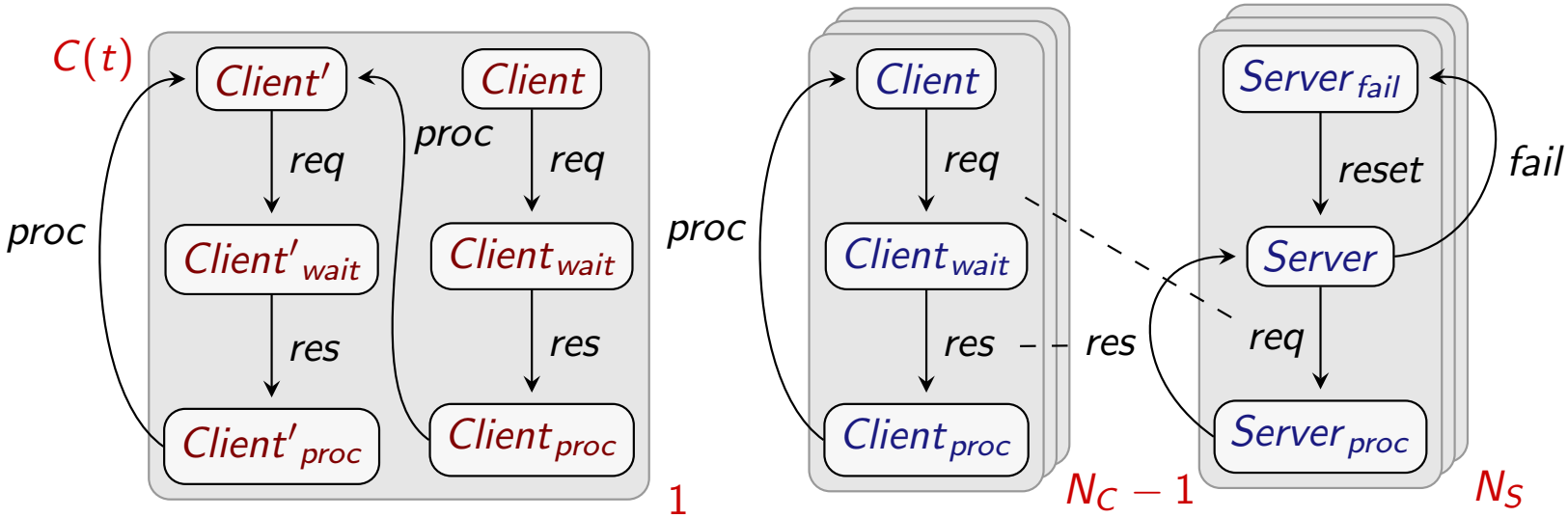
Individual passage times



$T := \inf\{t \geq 0 : C(t) = \mathit{Client}'\}$, given that $C(0) = \mathit{Client}$

$$\begin{aligned}\mathbb{P}\{T \leq t\} &= \mathbb{P}\{C(t) \in \{\mathit{Client}', \mathit{Client}'_{\mathit{wait}}, \mathit{Client}'_{\mathit{proc}}\}\} \\ &= \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'\}}] + \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'_{\mathit{wait}}\}}] + \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'_{\mathit{proc}}\}}] \\ &= \mathbb{E}[N_{\mathit{Client}'}(t)] + \mathbb{E}[N_{\mathit{Client}'_{\mathit{wait}}}(t)] + \mathbb{E}[N_{\mathit{Client}'_{\mathit{proc}}}(t)]\end{aligned}$$

Individual passage times

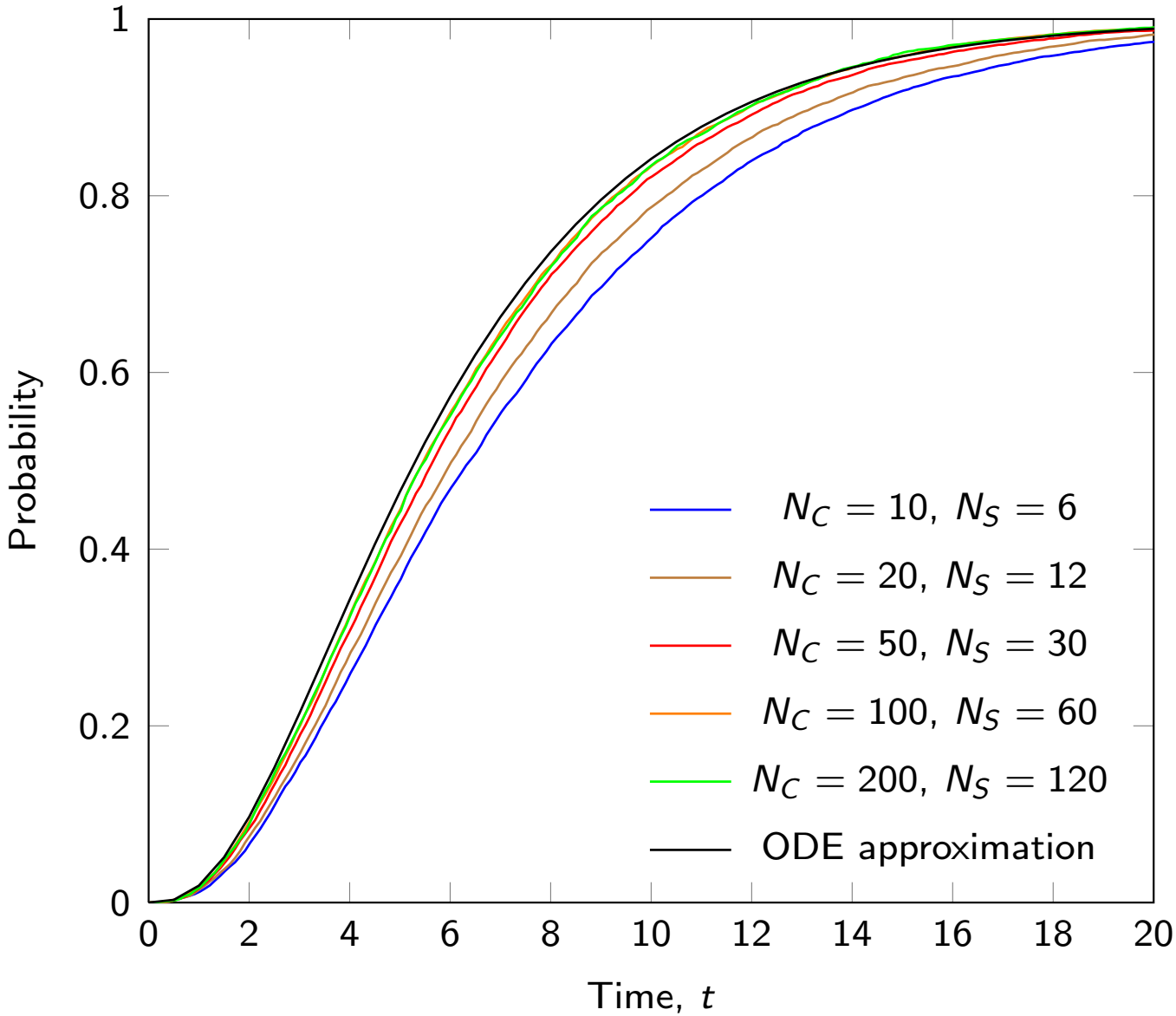


$T := \inf\{t \geq 0 : C(t) = \mathit{Client}'\}$, given that $C(0) = \mathit{Client}$

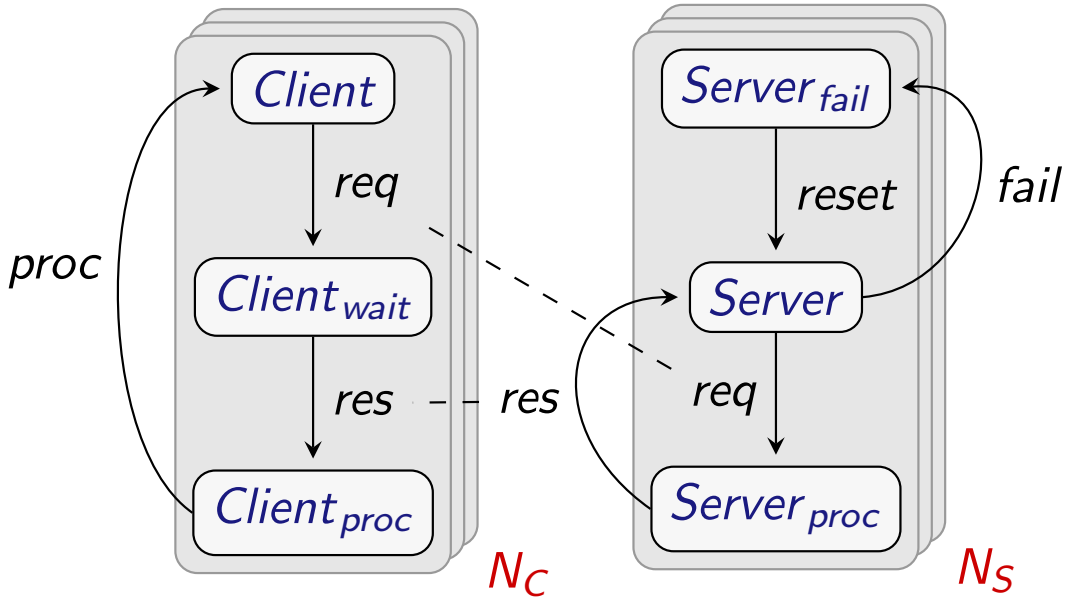
$$\begin{aligned}\mathbb{P}\{T \leq t\} &= \mathbb{P}\{C(t) \in \{\mathit{Client}', \mathit{Client}'_{\mathit{wait}}, \mathit{Client}'_{\mathit{proc}}\}\} \\ &= \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'\}}] + \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'_{\mathit{wait}}\}}] + \mathbb{E}[\mathbf{1}_{\{C(t)=\mathit{Client}'_{\mathit{proc}}\}}] \\ &= \mathbb{E}[N_{\mathit{Client}'}(t)] + \mathbb{E}[N_{\mathit{Client}'_{\mathit{wait}}}(t)] + \mathbb{E}[N_{\mathit{Client}'_{\mathit{proc}}}(t)]\end{aligned}$$

$$\mathbb{P}\{T \leq t\} \approx v_{\mathit{Client}'}(t) + v_{\mathit{Client}'_{\mathit{wait}}}(t) + v_{\mathit{Client}'_{\mathit{proc}}}(t)$$

Example — individual passage time



Global passage times

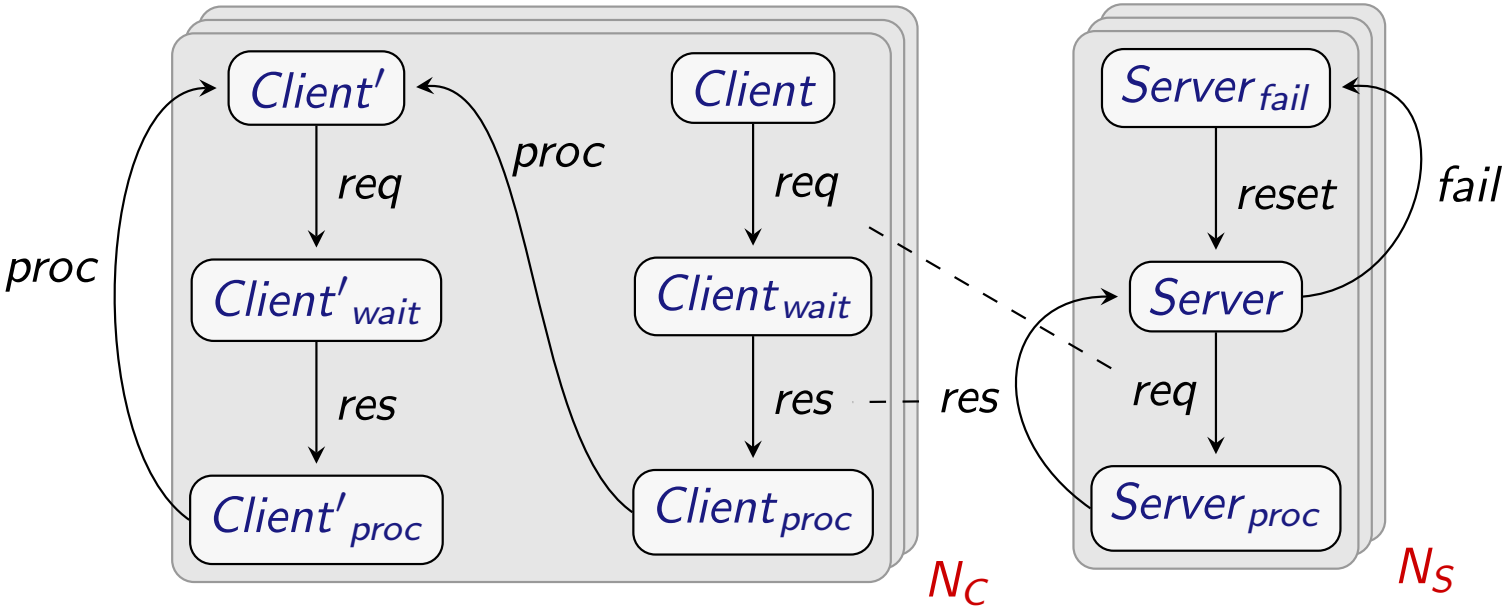


How long does it take for half of the clients to make a request, receive a response and process it?

[7] Richard A. Hayden, Anton Stefanek, and Jeremy T. Bradley. "Fluid computation of passage time distributions in large Markov models". In: *Theoretical Computer Science* 413.1 (2012), pp. 106–141. DOI: 10.1016/j.tcs.2011.07.017.

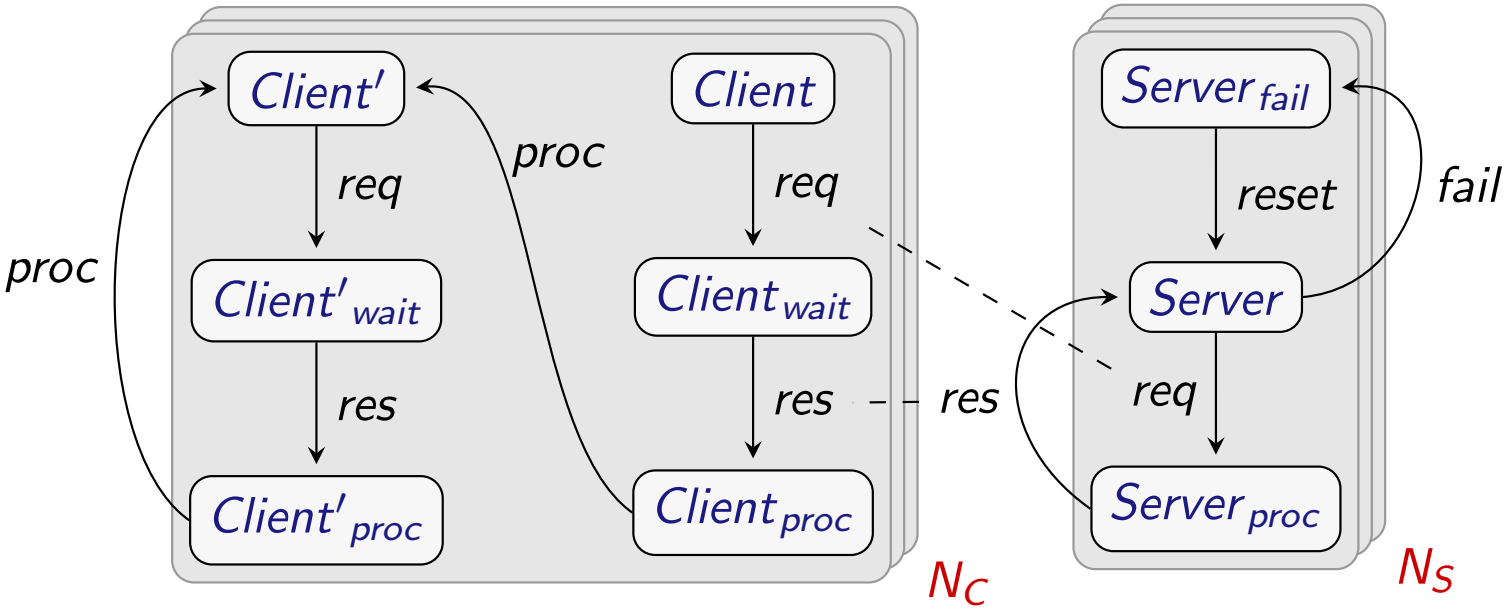
[9] Rena Bakhshi et al. "Mean-Field Analysis for the Evaluation of Gossip Protocols". In: *QEST'09, Proceedings of the 5th IEEE Conference on the Quantitative Evaluation of Systems*. IEEE Computer Society, 2009, pp. 247–256.

Global passage times



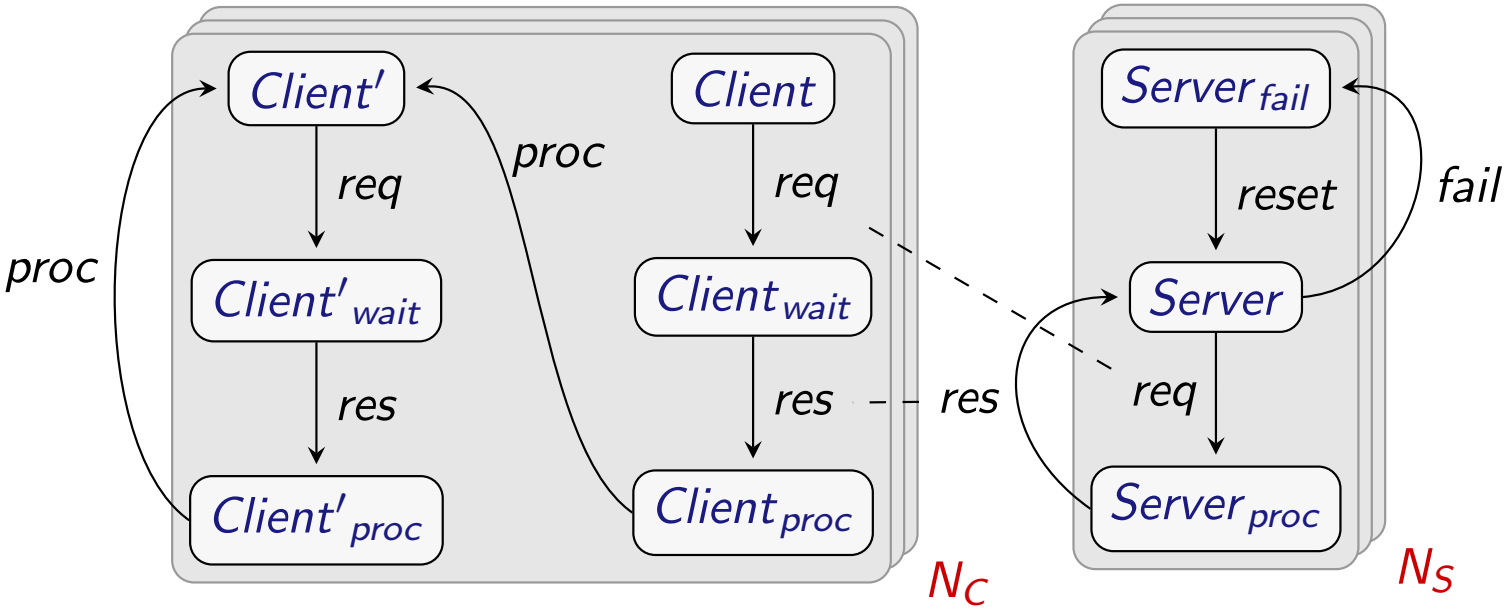
How long does it take for half of the clients to make a request, receive a response and process it?

Global passage times



$$T := \inf\{t \geq 0 : N_{C'}(t) + N_{C'_w}(t) + N_{C'_p}(t) \geq N_C/2\}$$

Global passage times

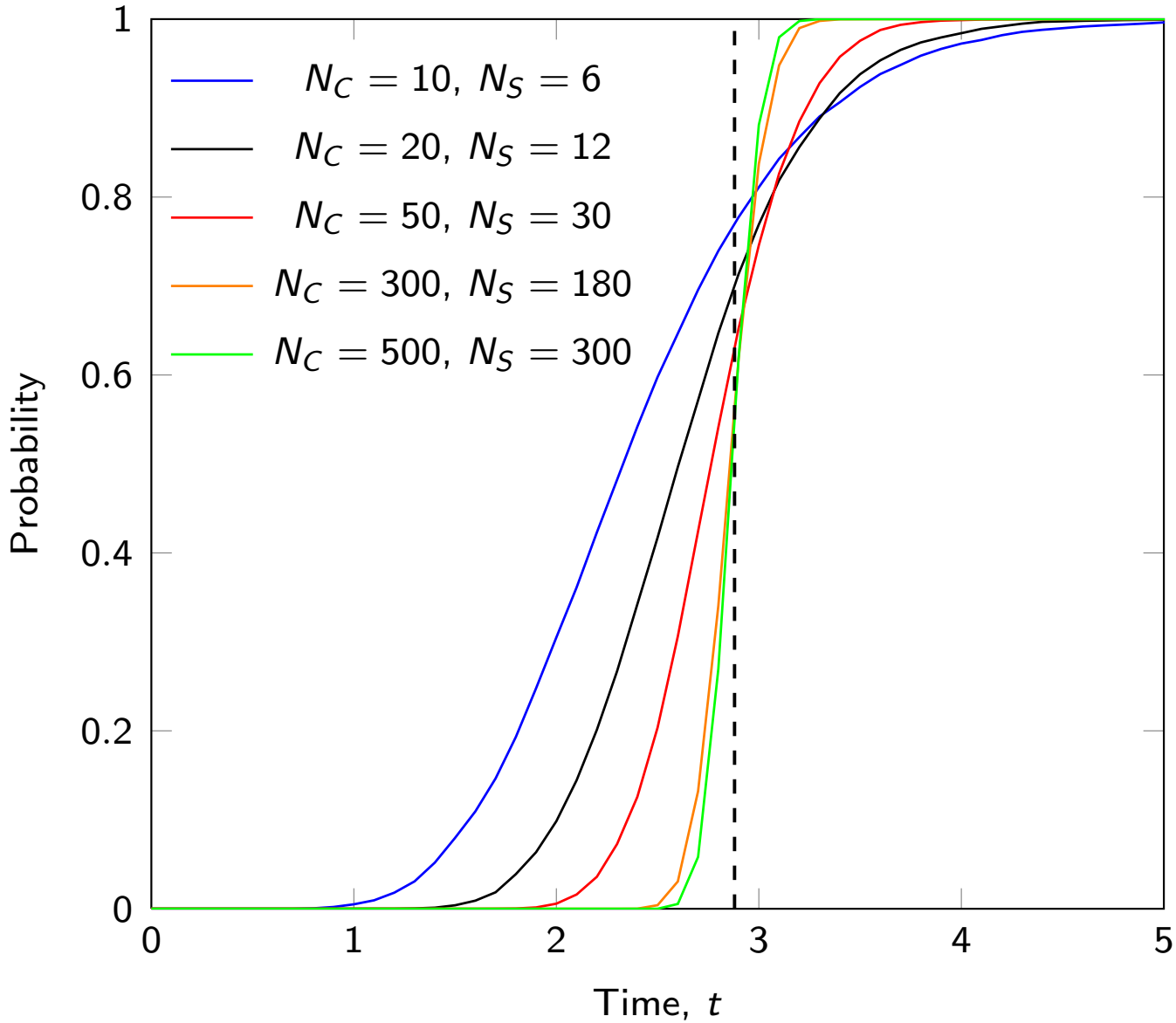


$$T := \inf\{t \geq 0 : N_{C'}(t) + N_{C'_w}(t) + N_{C'_p}(t) \geq N_C/2\}$$

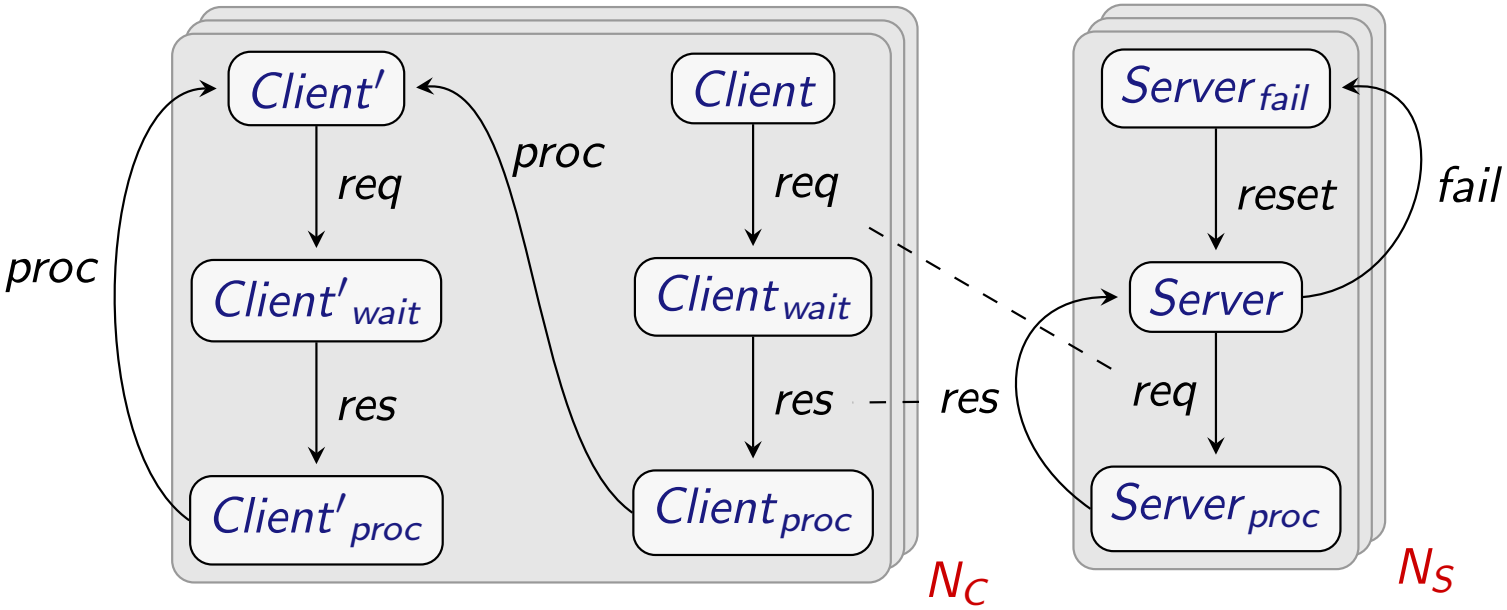
Point-mass approximation:

$$T \approx \inf\{t \geq 0 : v_{C'}(t) + v_{C'_w}(t) + v_{C'_p}(t) \geq N_C/2\}$$

Global passage times



Global passage times

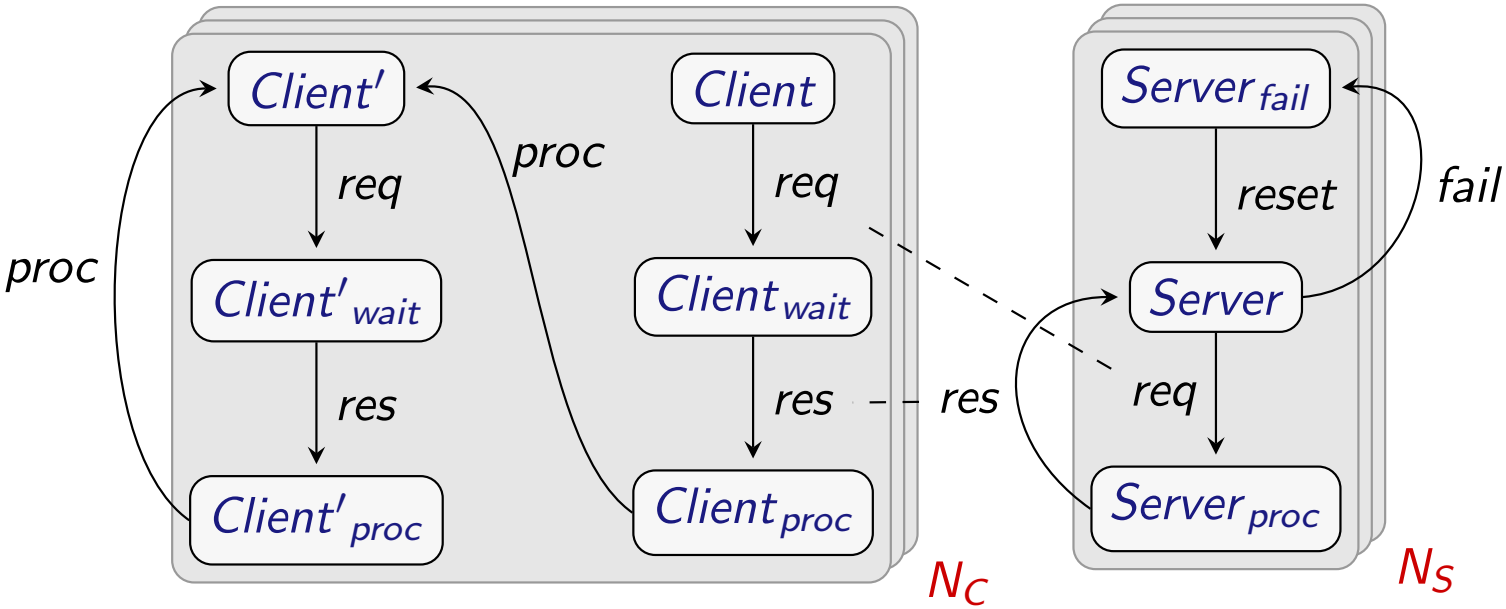


Point-mass approximation:

$$T \approx \inf\{t \geq 0 : v_{C'}(t) + v_{C'_w}(t) + v_{C'_p}(t) \geq N_C/2\}$$

- Approximation is very coarse
- Cannot be applied directly to the same question for *all* clients

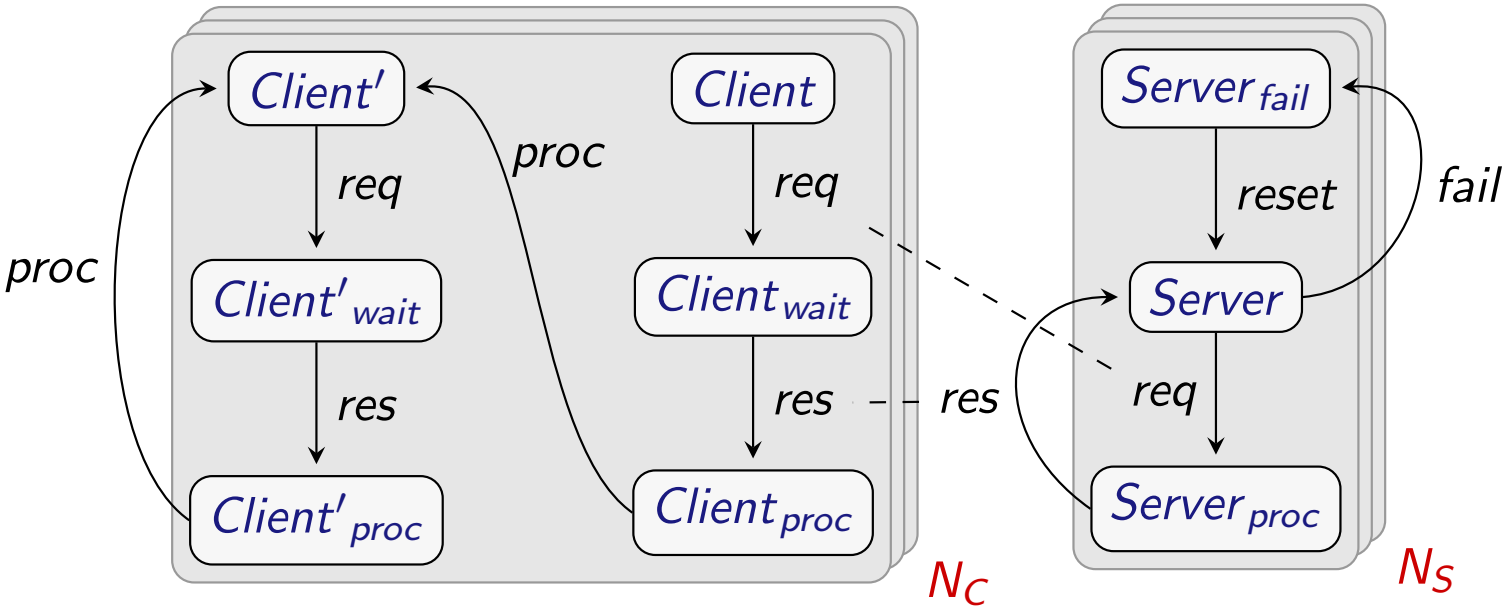
Global passage times — moment bounds



- Moment approximations to component counts contain information about the distribution of T[7]

[7] Richard A. Hayden, Anton Stefanek, and Jeremy T. Bradley. "Fluid computation of passage time distributions in large Markov models". In: *Theoretical Computer Science* 413.1 (2012), pp. 106–141. DOI: 10.1016/j.tcs.2011.07.017.

Global passage times — moment bounds

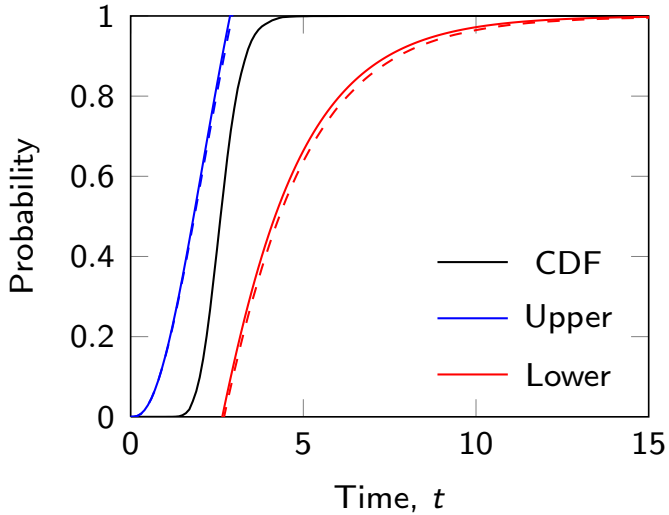


- Moment approximations to component counts contain information about the distribution of T
- Reduced moment problem — find maximum and minimum bounding distributions subject to limited moment information[10]

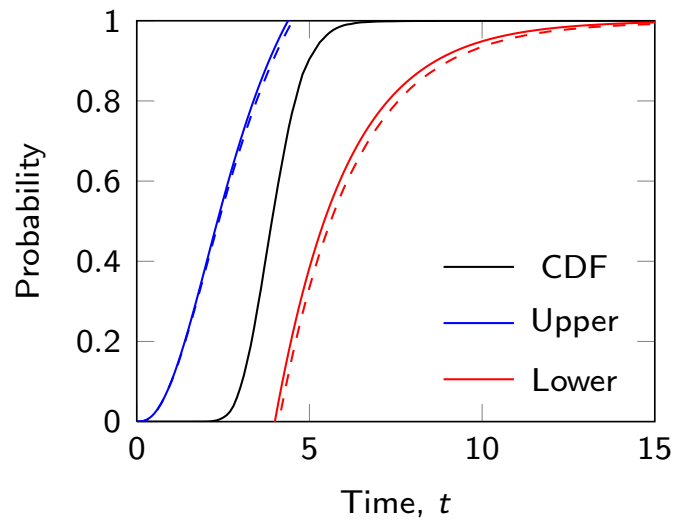
[10] Arpád Tari, Miklós Telek, and Peter Buchholz. "A Unified Approach to the Moments-based Distribution Estimation–Unbounded Support". In: *EPEW'05, European Performance Engineering Workshop*. Vol. 3670. Lecture Notes in Computer Science. Versailles: Springer, 2005, pp. 79–93.

Global passage bounds — first moments

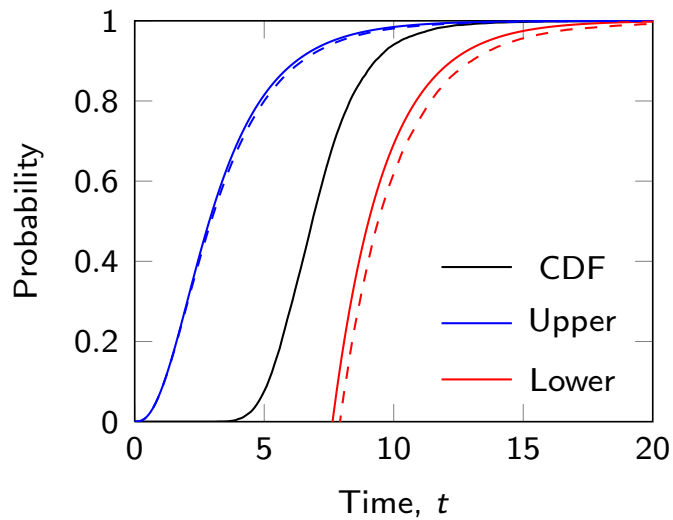
Half of the clients:



Three quarters of the clients:

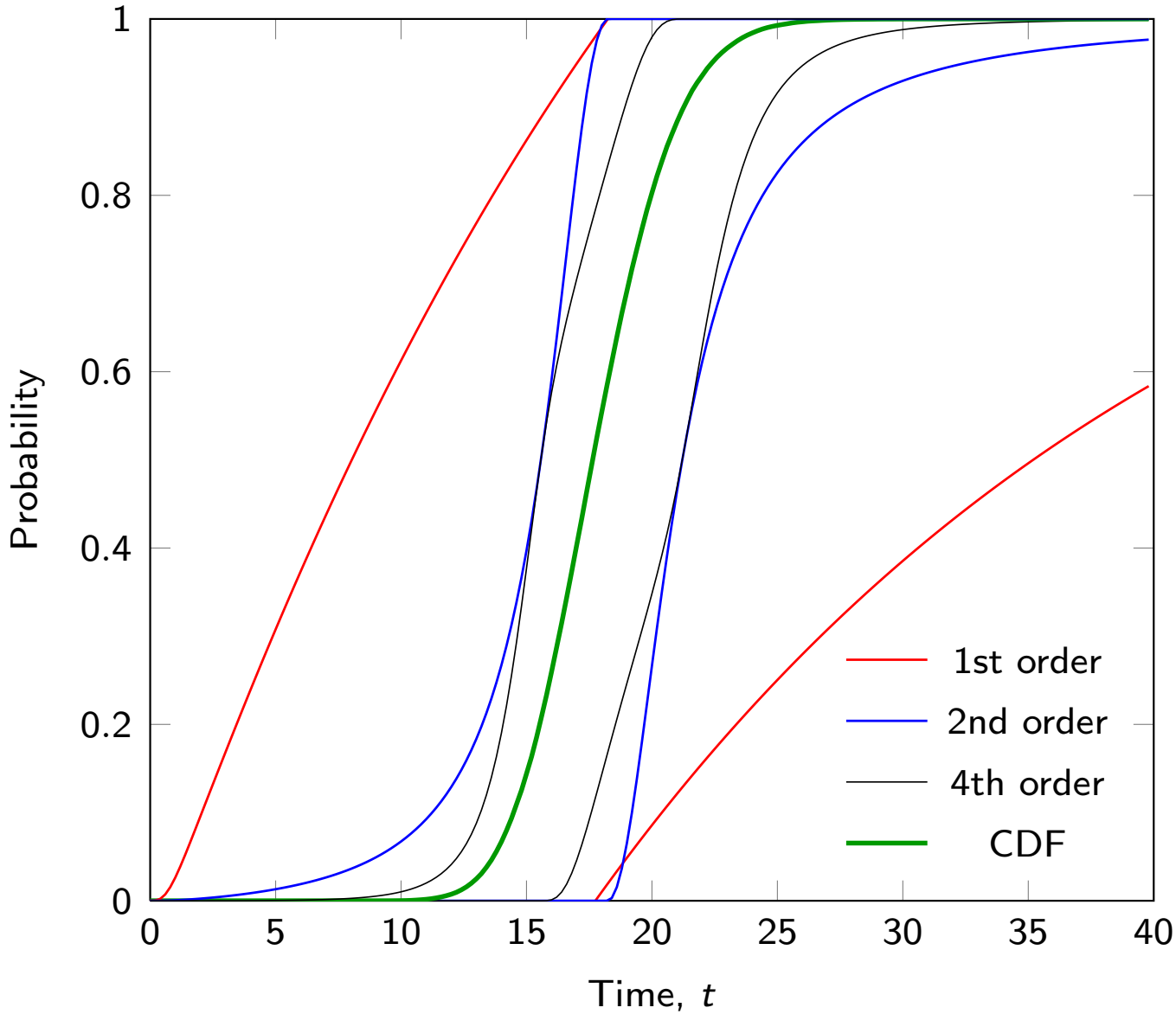


All of the clients:



[7] Richard A. Hayden, Anton Stefanek, and Jeremy T. Bradley. "Fluid computation of passage time distributions in large Markov models". In: *Theoretical Computer Science* 413.1 (2012), pp. 106–141. DOI: 10.1016/j.tcs.2011.07.017.

Global passage bounds — higher moments



[7] Richard A. Hayden, Anton Stefanek, and Jeremy T. Bradley. "Fluid computation of passage time distributions in large Markov models". In: *Theoretical Computer Science* 413.1 (2012), pp. 106–141. DOI: 10.1016/j.tcs.2011.07.017.

Scalable analysis of accumulated reward measures

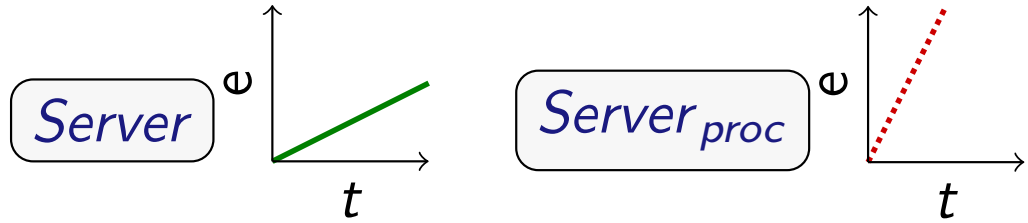
Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

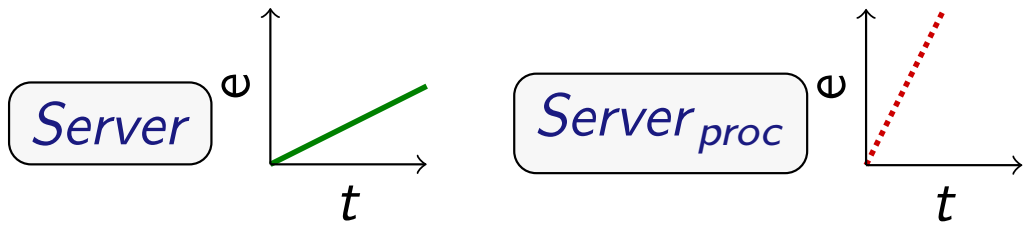
- Cost, energy, heat, ...
- Constant rate

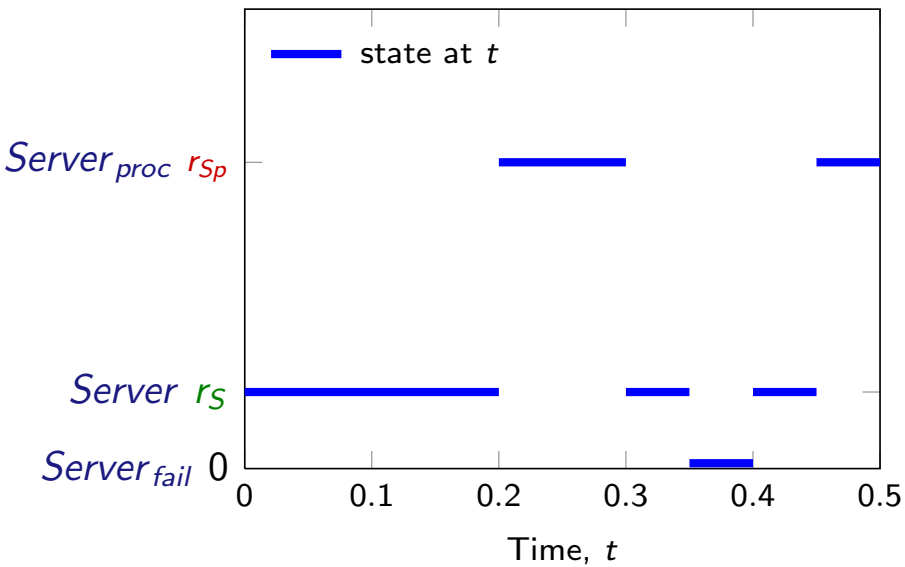


[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

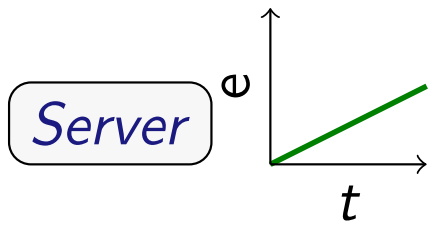


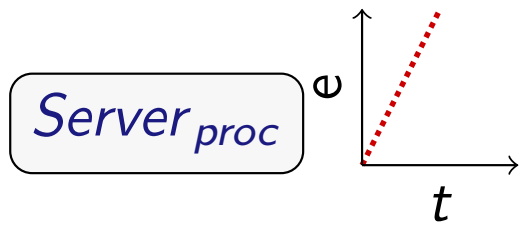


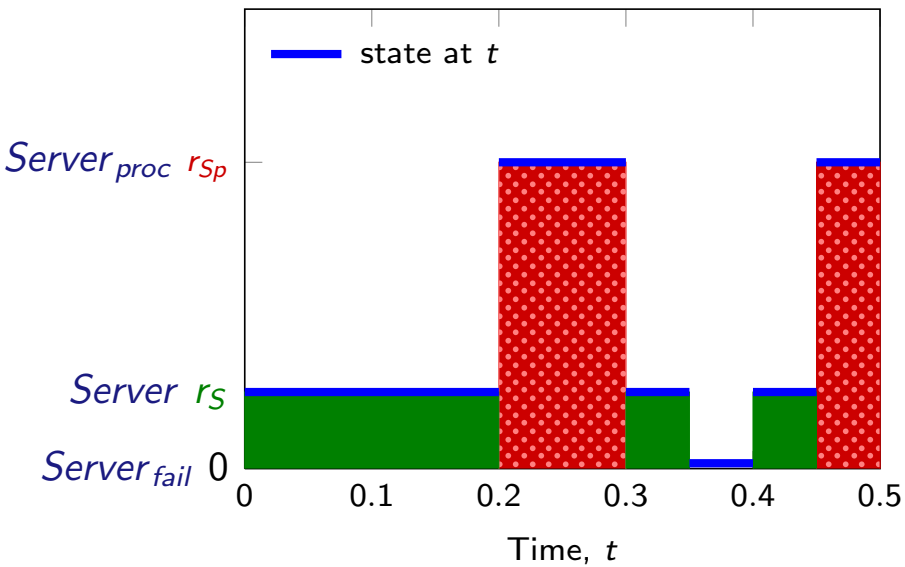
[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate



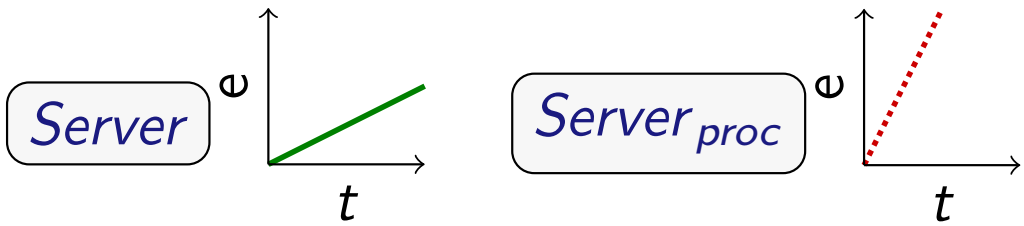


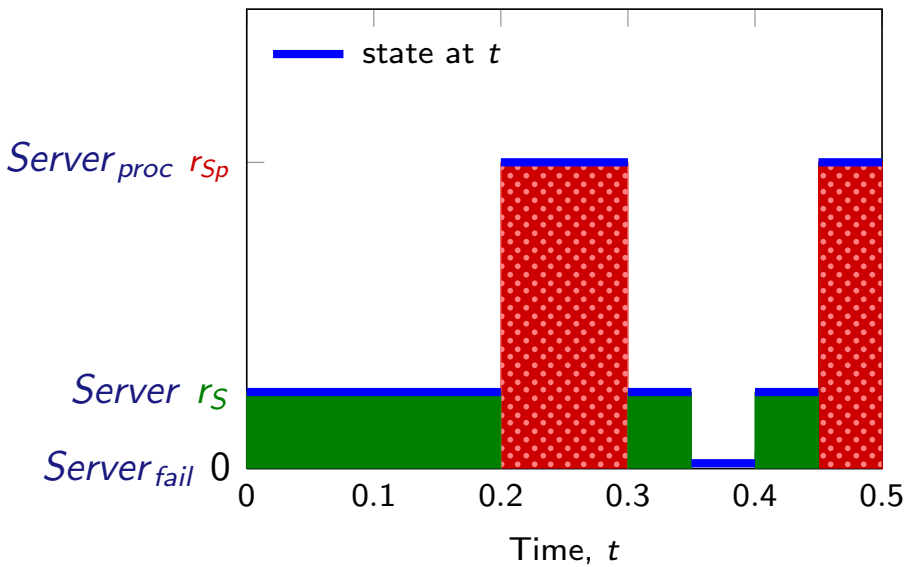


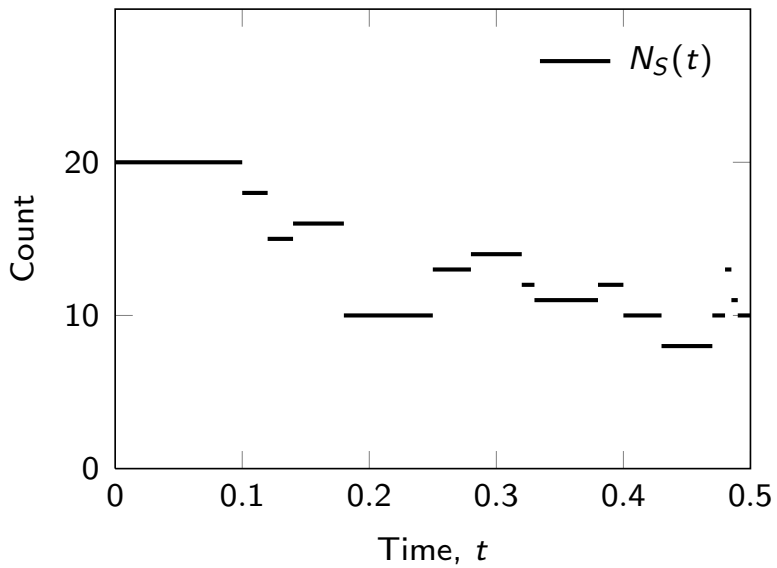
[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate



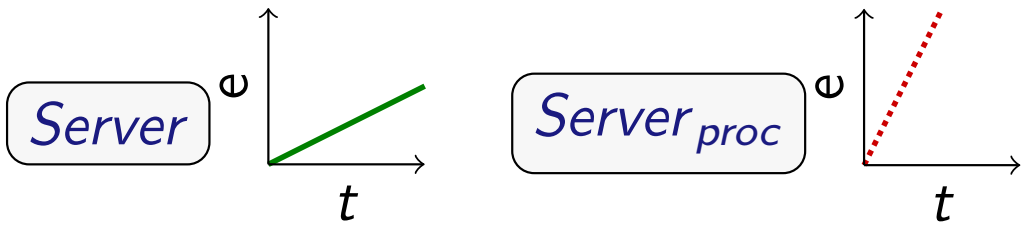


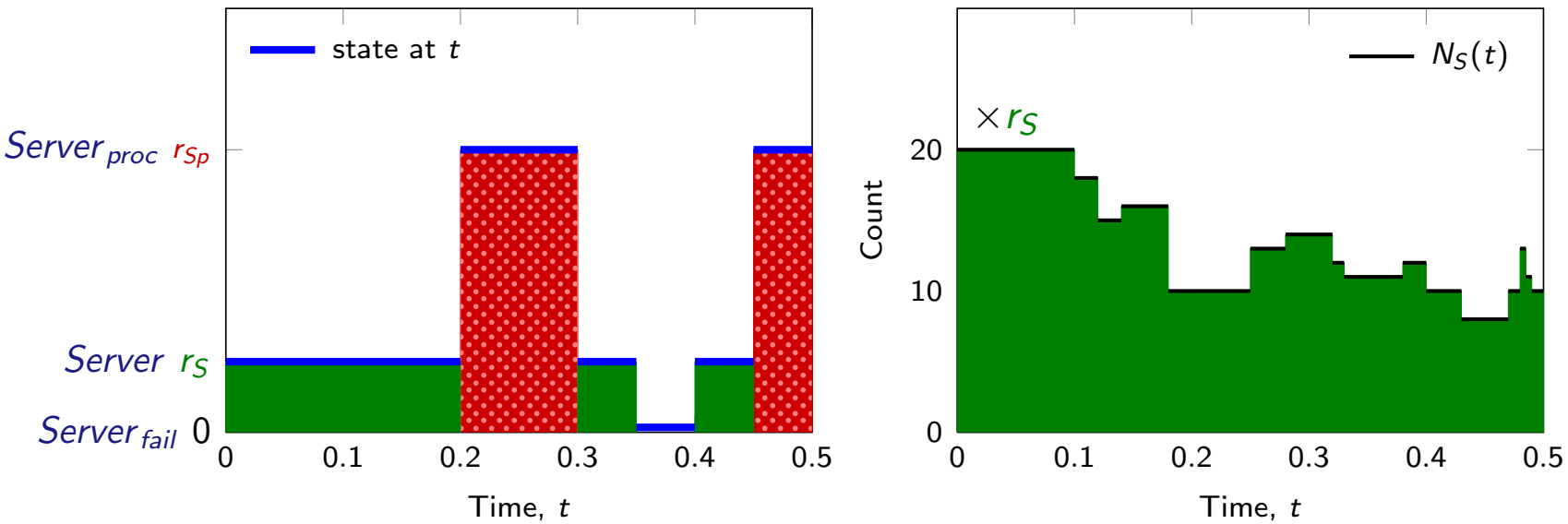


[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

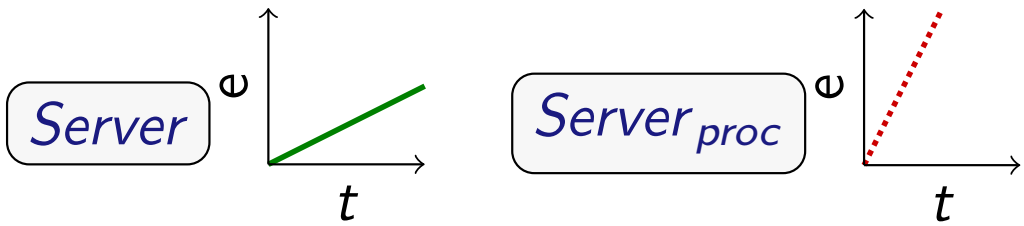


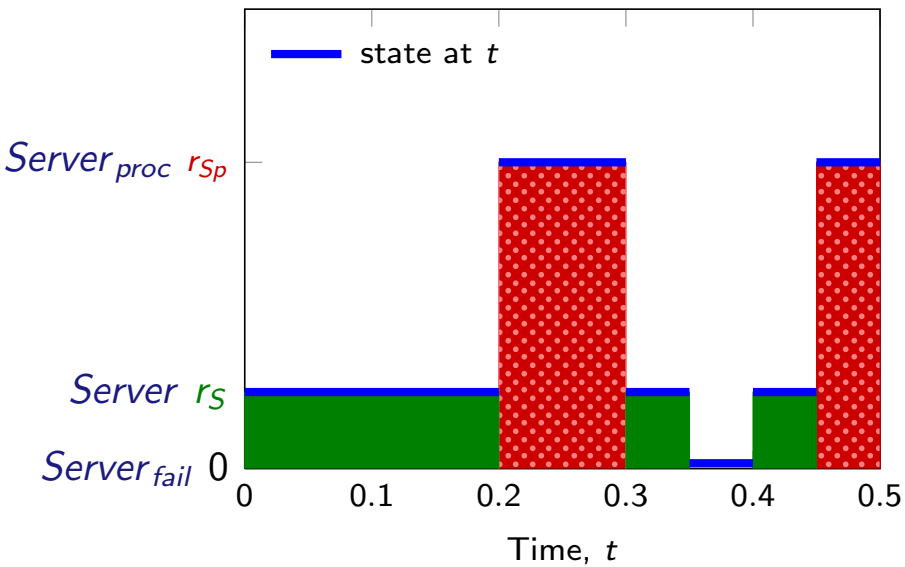


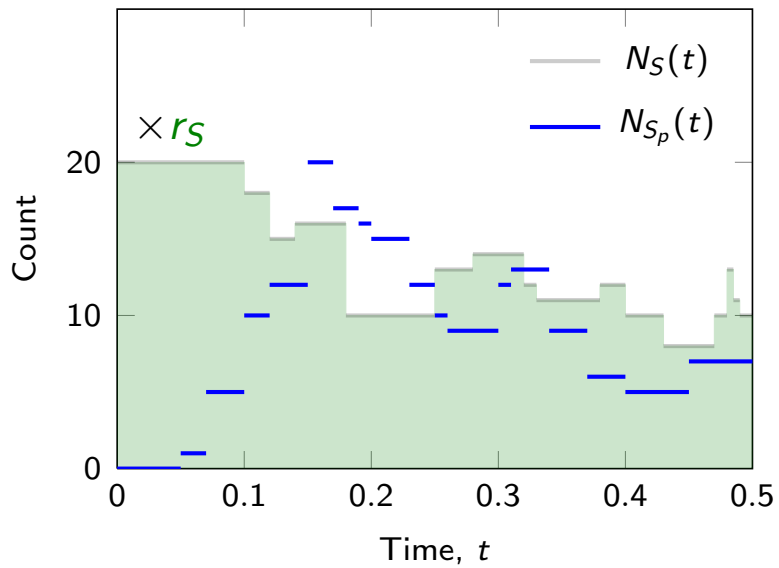
[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate



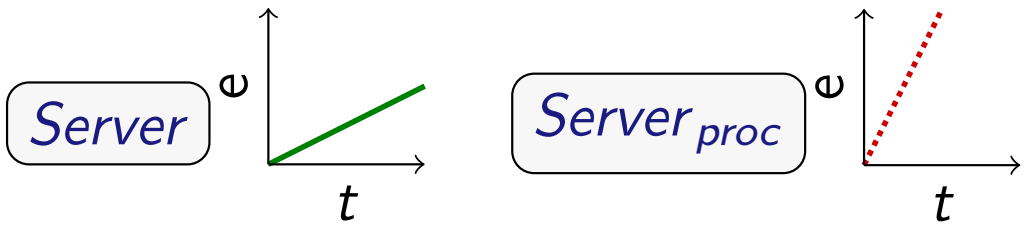


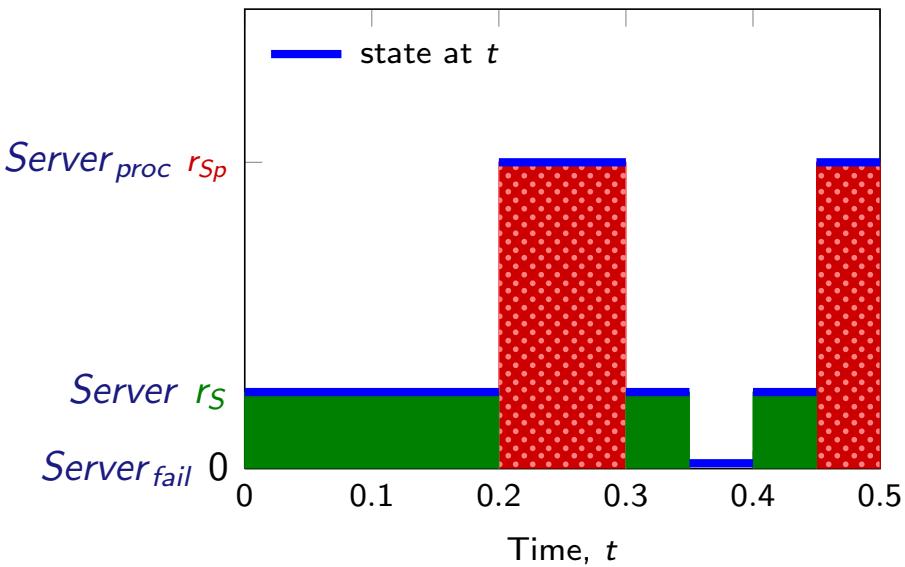


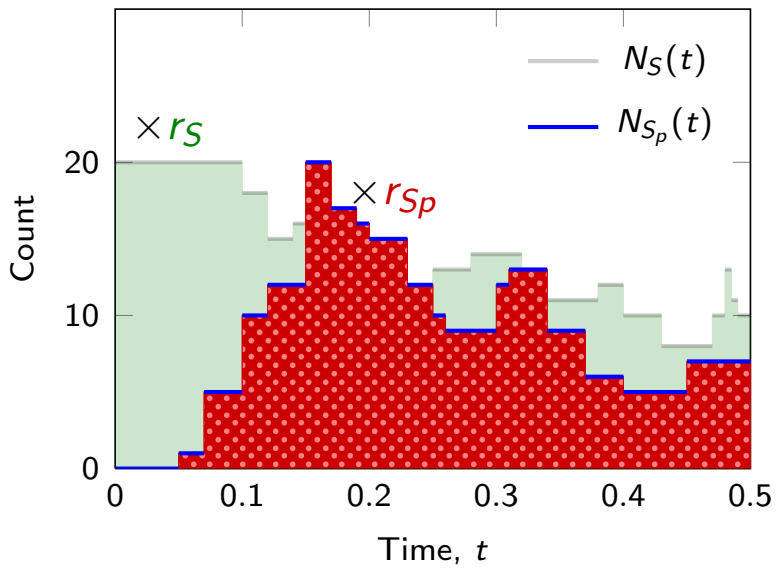
[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate



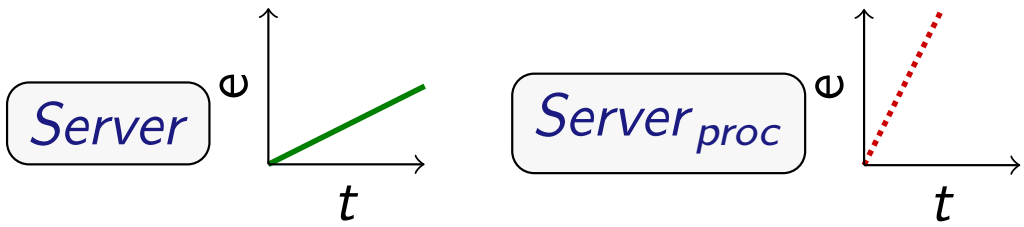


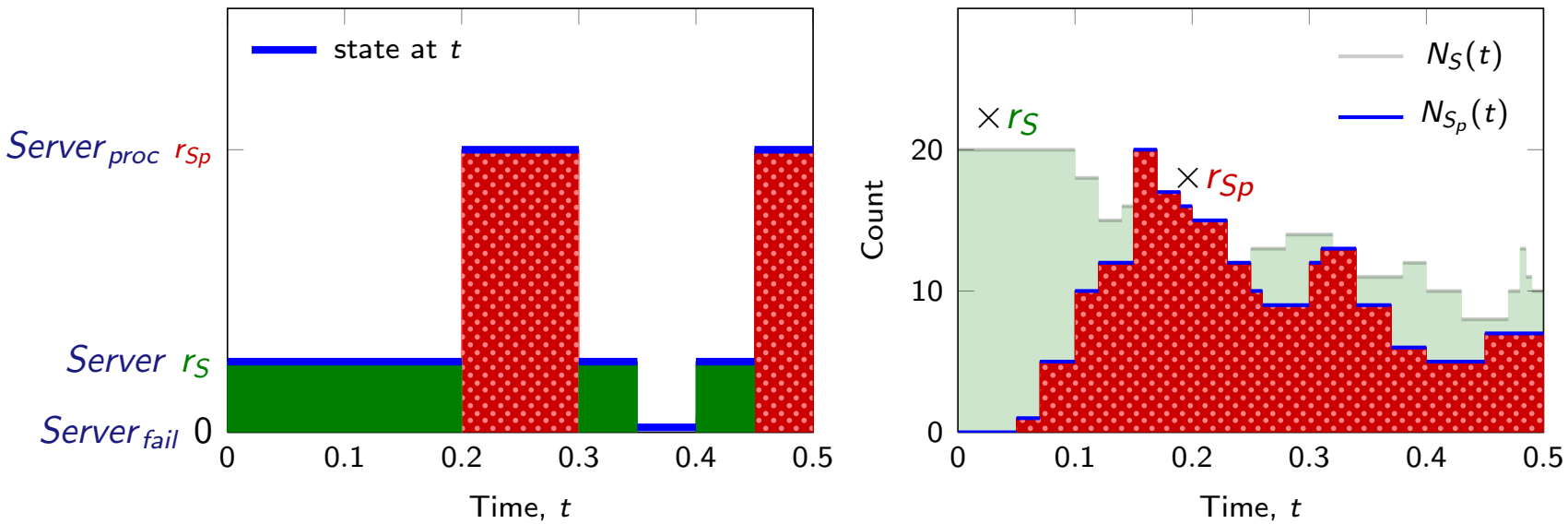


[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Accumulated reward measures

- Cost, energy, heat, ...
- Constant rate

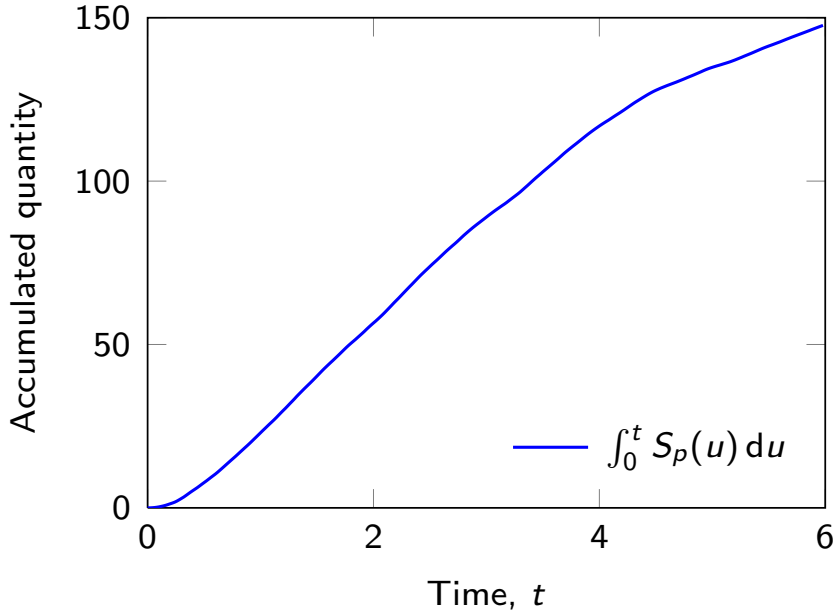




$$\text{total energy}(t) = r_S \int_0^t N_S(u)\,\mathrm{d}u + r_{Sp} \int_0^t N_{S_p}(u)\,\mathrm{d}u$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

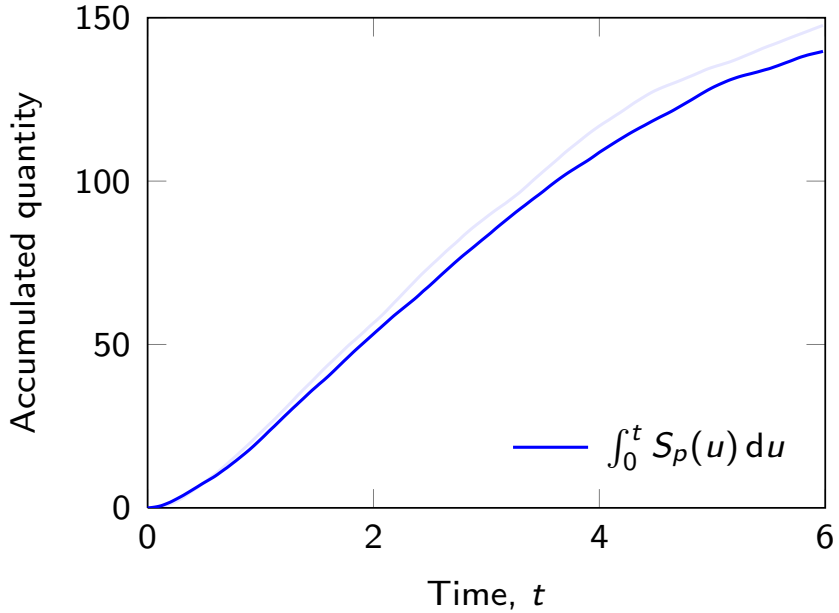
Moment approximations of accumulated rewards



- Simulation also very costly for rewards

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

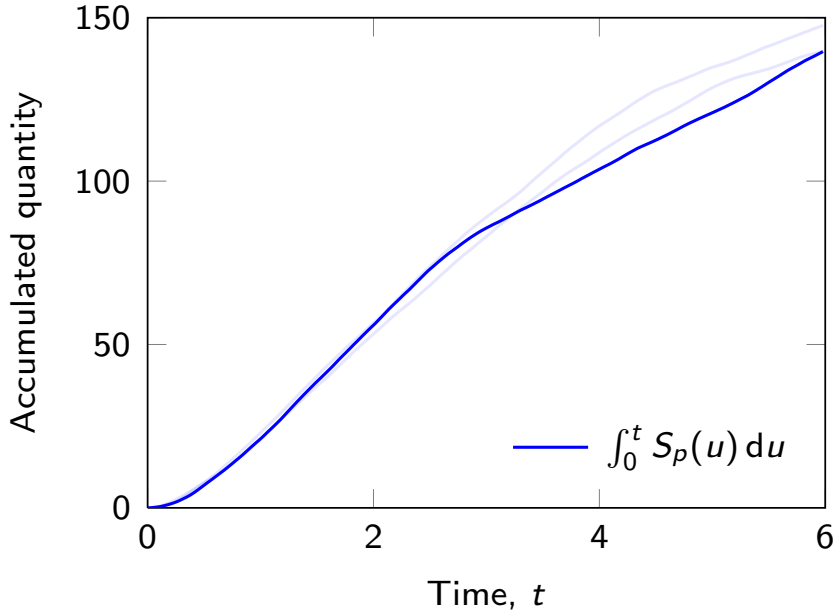
Moment approximations of accumulated rewards



- Simulation also very costly for rewards

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

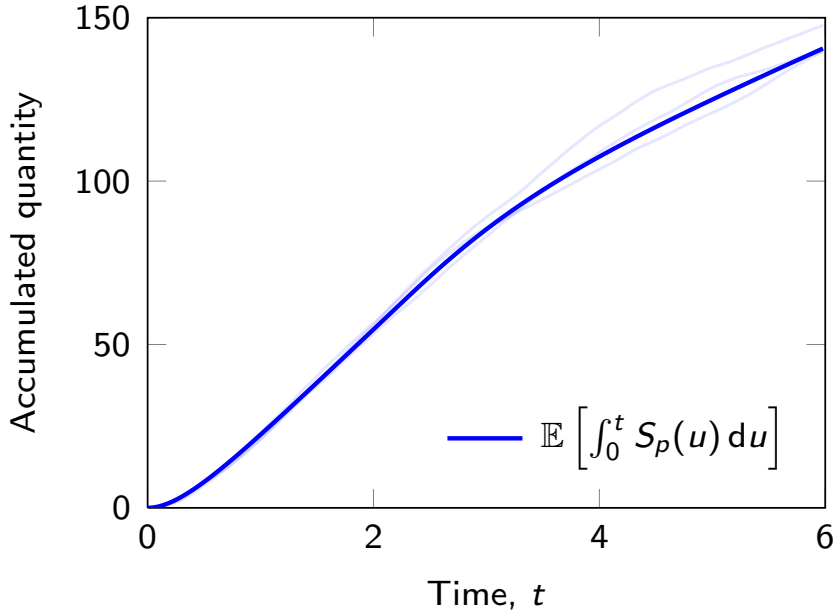
Moment approximations of accumulated rewards



- Simulation also very costly for rewards

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

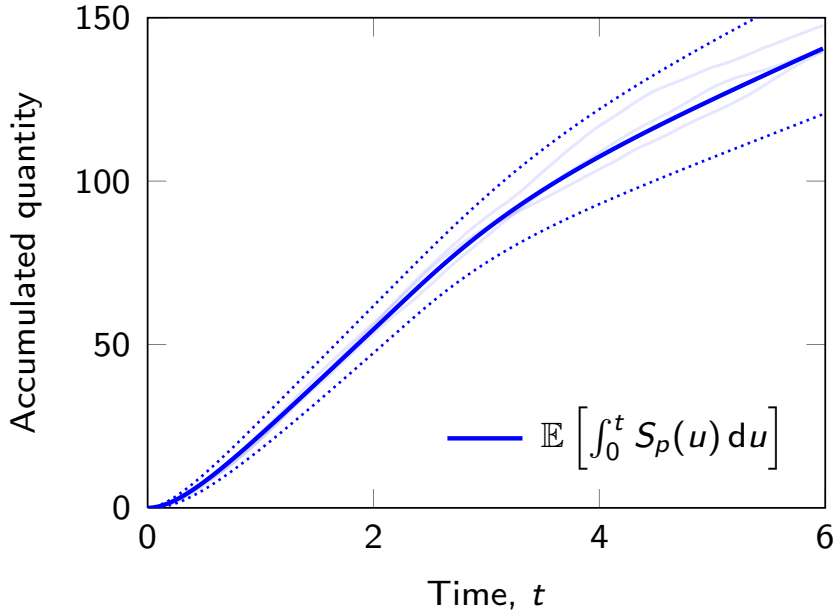


- Simulation also very costly for rewards
- Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_{S_p}(u)\,\mathrm{d}u\right] = \cdots$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

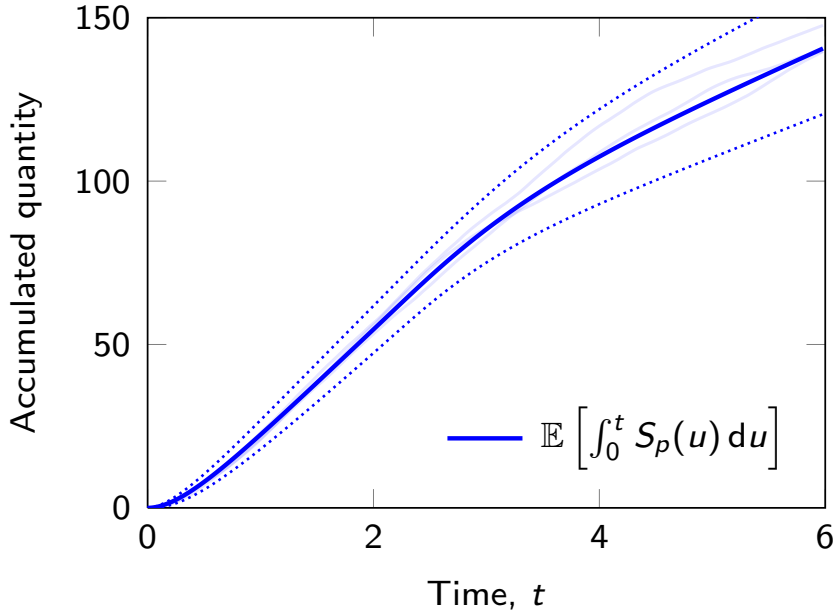


- Simulation also very costly for rewards
- Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_{S_p}(u)\,\mathrm{d}u \int_0^t N_S(u)\,\mathrm{d}u\right] = \cdots$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

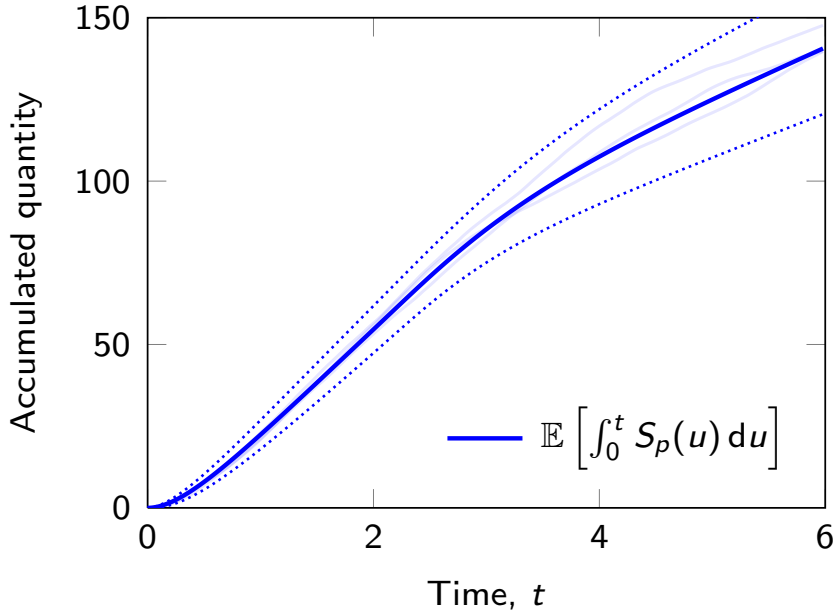


- Simulation also very costly for rewards
- Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_{S_p}(u)\,\mathrm{d}u \int_0^t N_S(u)\,\mathrm{d}u\right] = \cdots$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

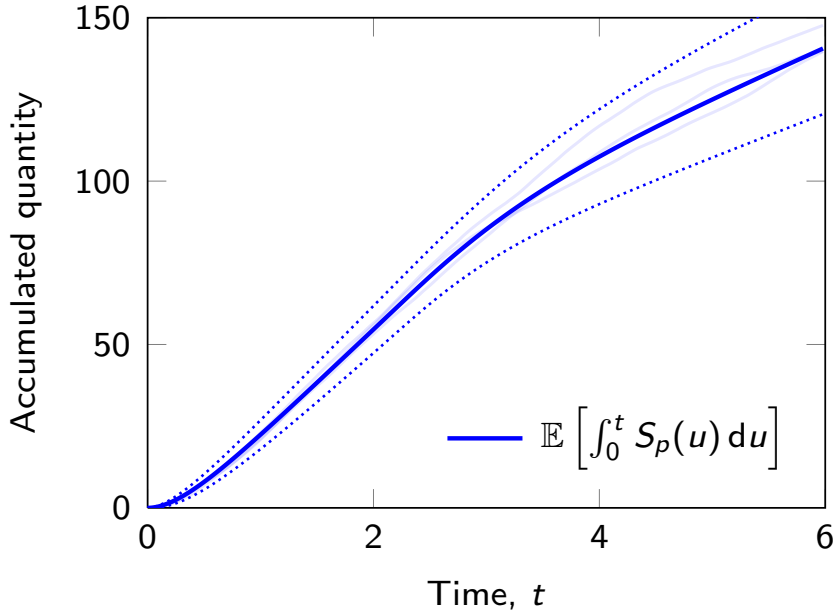


- Simulation also very costly for rewards
- Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_{S_p}(u)\,\mathrm{d}u \int_0^t N_S(u)\,\mathrm{d}u\right] = \cdots$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards



- Simulation also very costly for rewards
- Can extend the ODE system for count moments with ODEs for moments of accumulated counts:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_{S_p}(u)\,\mathrm{d}u \int_0^t N_S(u)\,\mathrm{d}u\right] = \cdots$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] =$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)] = \cdots$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)] = \cdots$$

Second-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\left(\int_0^t N_S(u)\,\mathrm{d}u\right)^2\right] =$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)] = \cdots$$

Second-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\left(\int_0^t N_S(u)\,\mathrm{d}u\right)^2\right] = 2\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right]$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)] = \cdots$$

Second-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\left(\int_0^t N_S(u)\,\mathrm{d}u\right)^2\right] = 2\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right]$$

Combined moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right] =$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)] = \cdots$$

Second-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\left(\int_0^t N_S(u)\,\mathrm{d}u\right)^2\right] = 2\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right]$$

Combined moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right] + \cdots + \mathbb{E}[N_S^2(t)]$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)] = \cdots$$

Second-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)N_{S_p}(t)] = \cdots$$

Second-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\left(\int_0^t N_S(u)\,\mathrm{d}u\right)^2\right] = 2\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right]$$

Combined moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right] + \cdots + \mathbb{E}[N_S^2(t)]$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

Moment approximations of accumulated rewards

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}[N_S(t)]$$

First-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)] = \cdots$$

Second-order moments

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[N_S(t)N_{S_p}(t)] = \cdots$$

Second-order moments

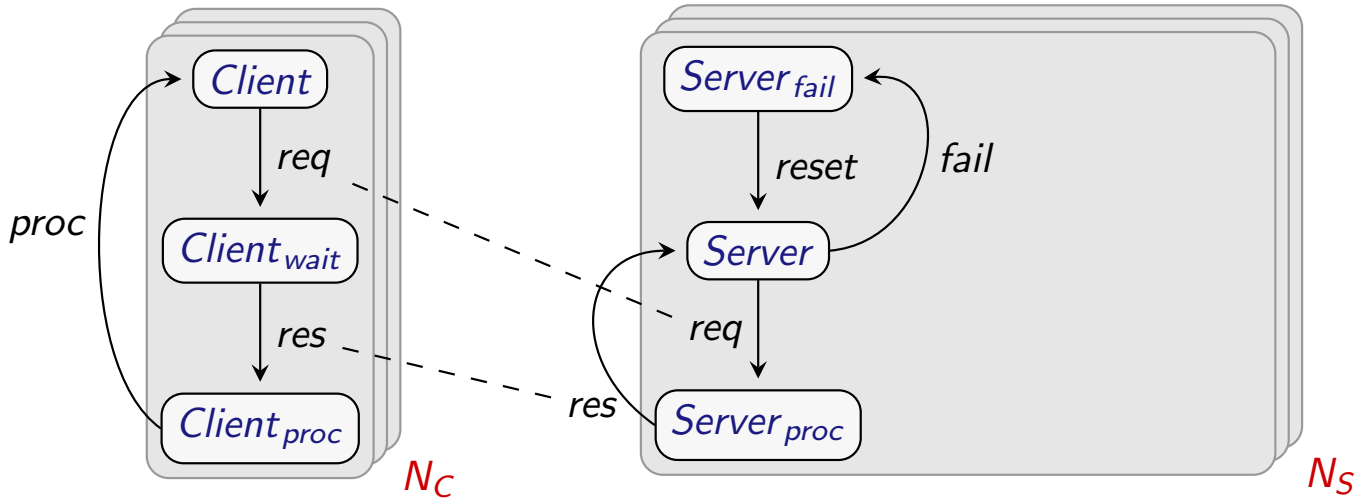
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\left(\int_0^t N_S(u)\,\mathrm{d}u\right)^2\right] = 2\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right]$$

Combined moments

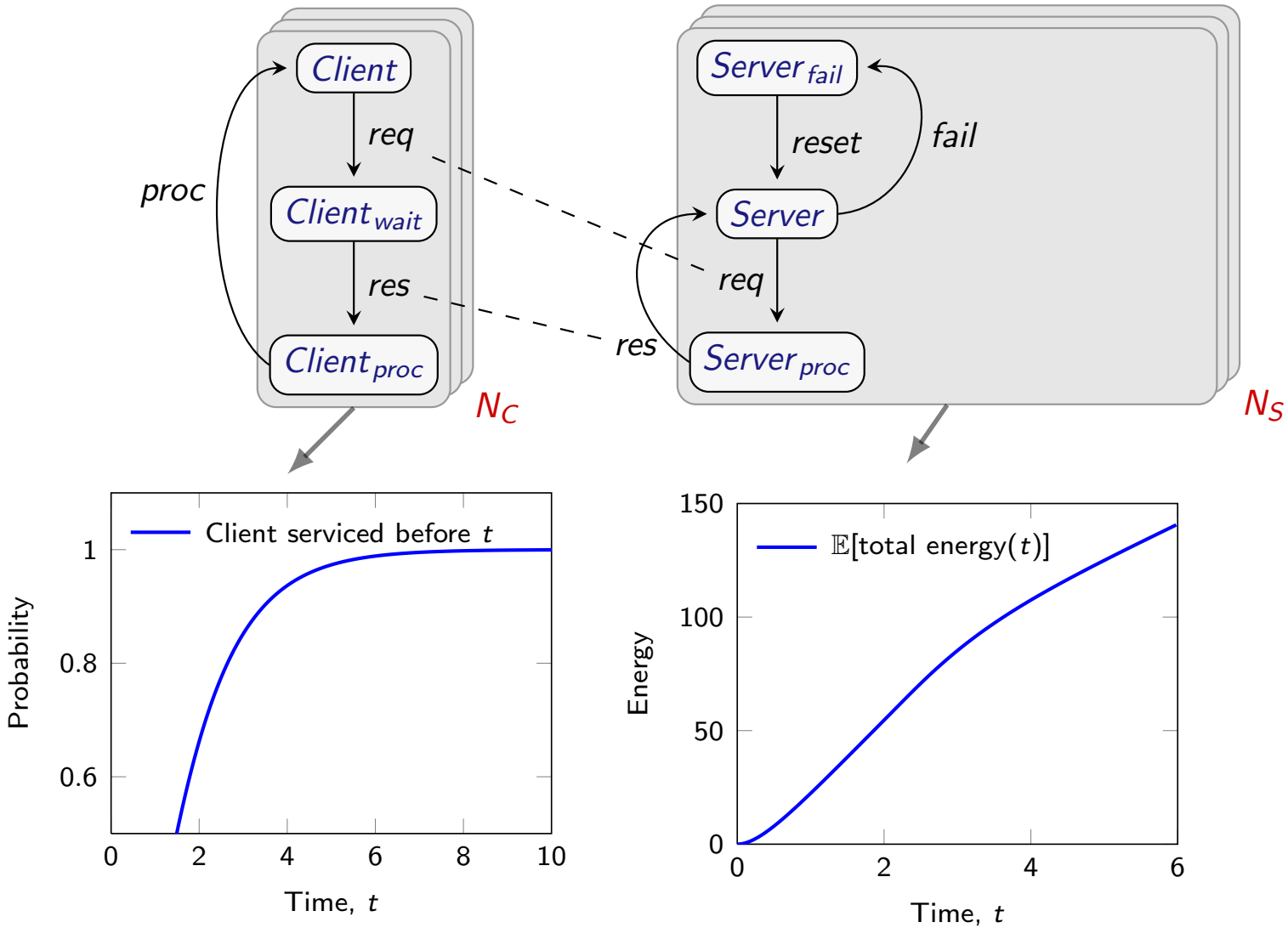
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right] = \mathbb{E}\left[N_S(t)\int_0^t N_S(u)\,\mathrm{d}u\right] + \cdots + \mathbb{E}[N_S^2(t)]$$

[11] Anton Stefanek, Richard A. Hayden, and Jeremy T. Bradley. "Fluid analysis of energy consumption using rewards in massively parallel Markov models". In: *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2011, pp. 121–132. DOI: 10.1145/1958746.1958767.

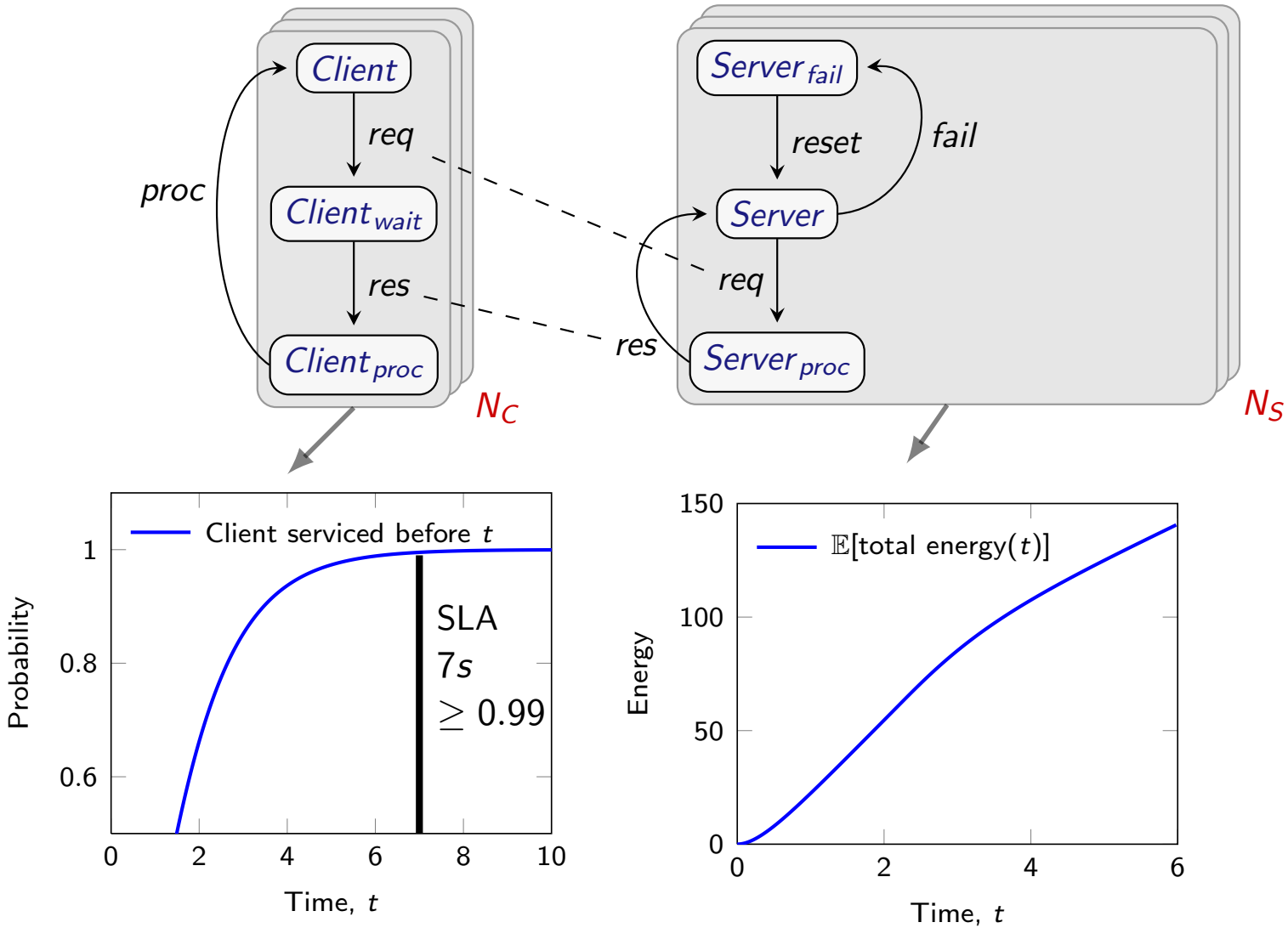
Trade-off between energy and performance



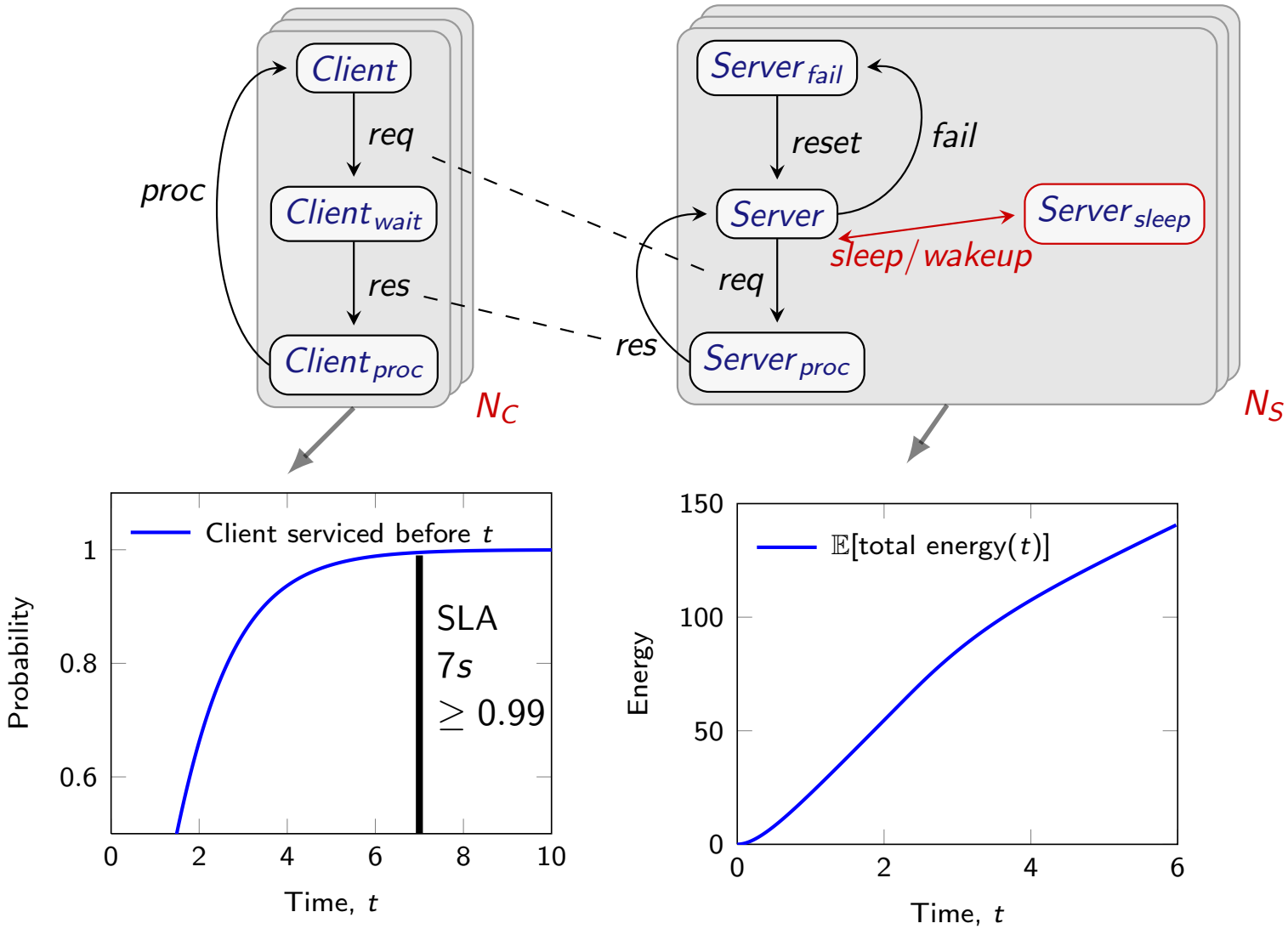
Trade-off between energy and performance



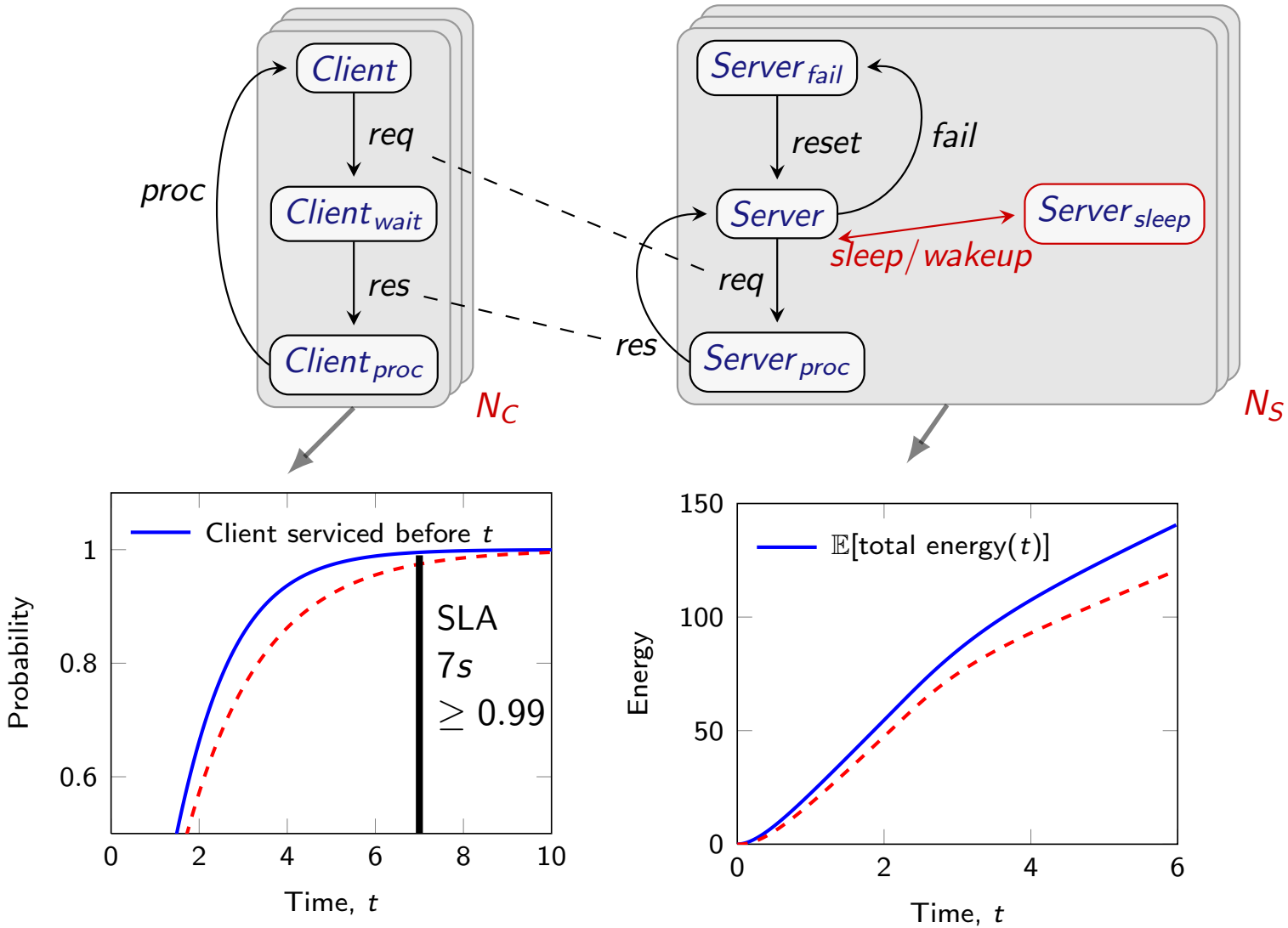
Trade-off between energy and performance



Trade-off between energy and performance



Trade-off between energy and performance



Trade-off between energy and performance

Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Trade-off between energy and performance

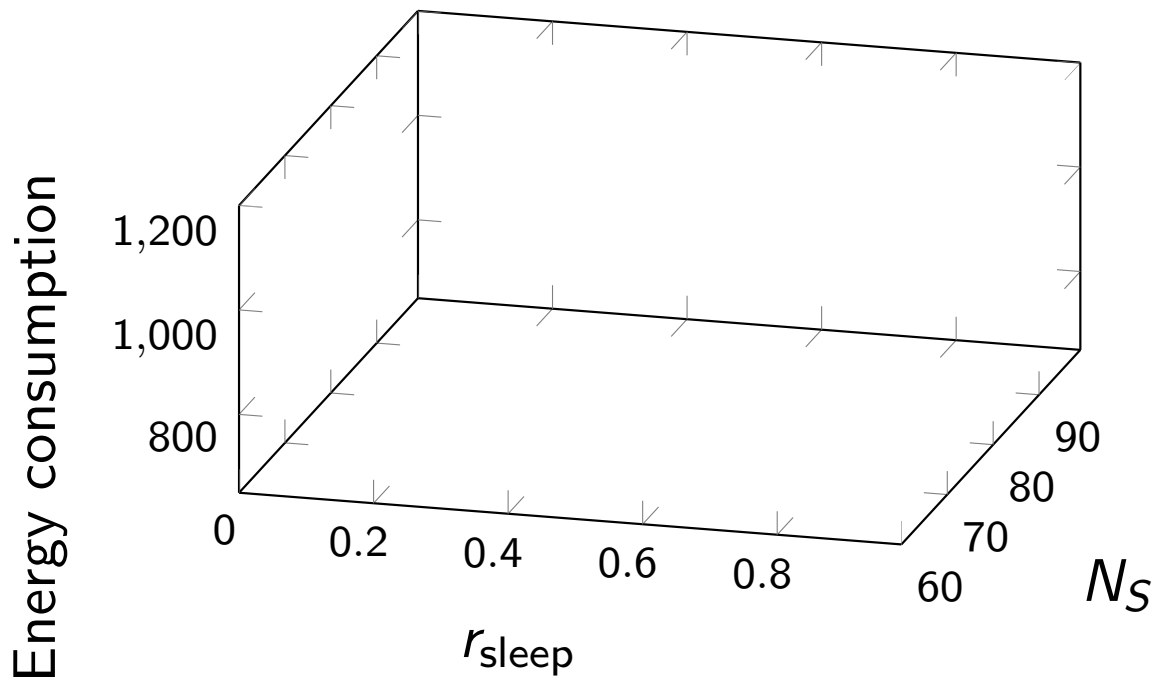
Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Minimise energy consumption while satisfying SLAs

Trade-off between energy and performance

Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Minimise energy consumption while satisfying SLAs

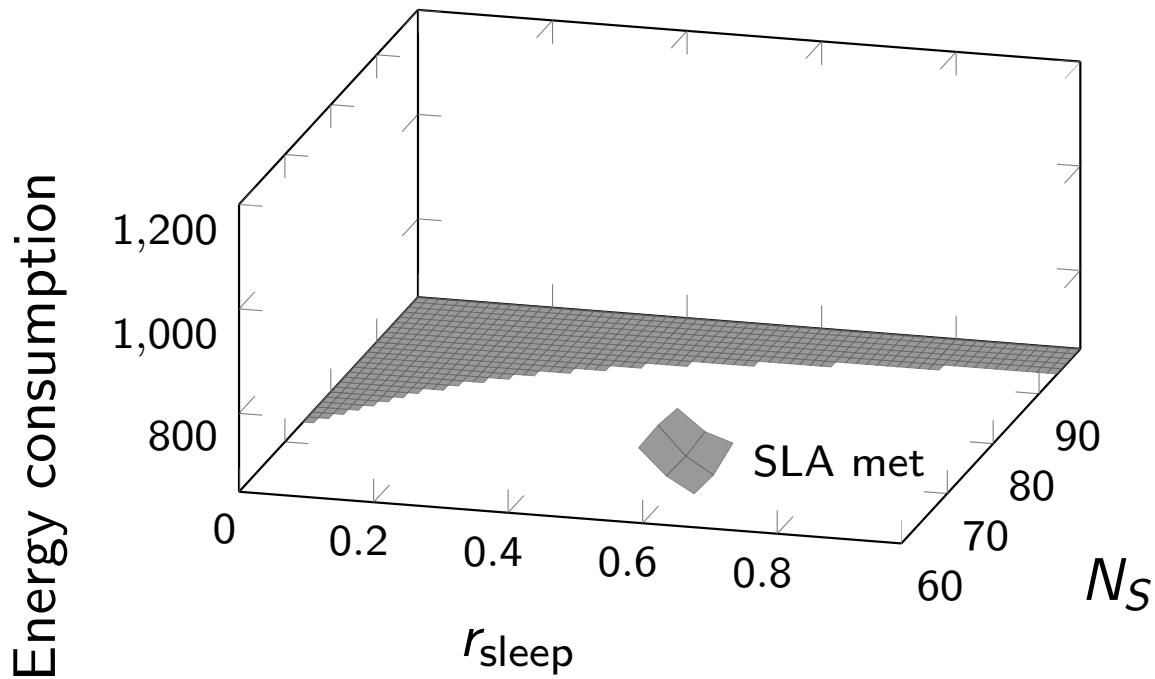


Individual passage-time SLA:

Trade-off between energy and performance

Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Minimise energy consumption while satisfying SLAs

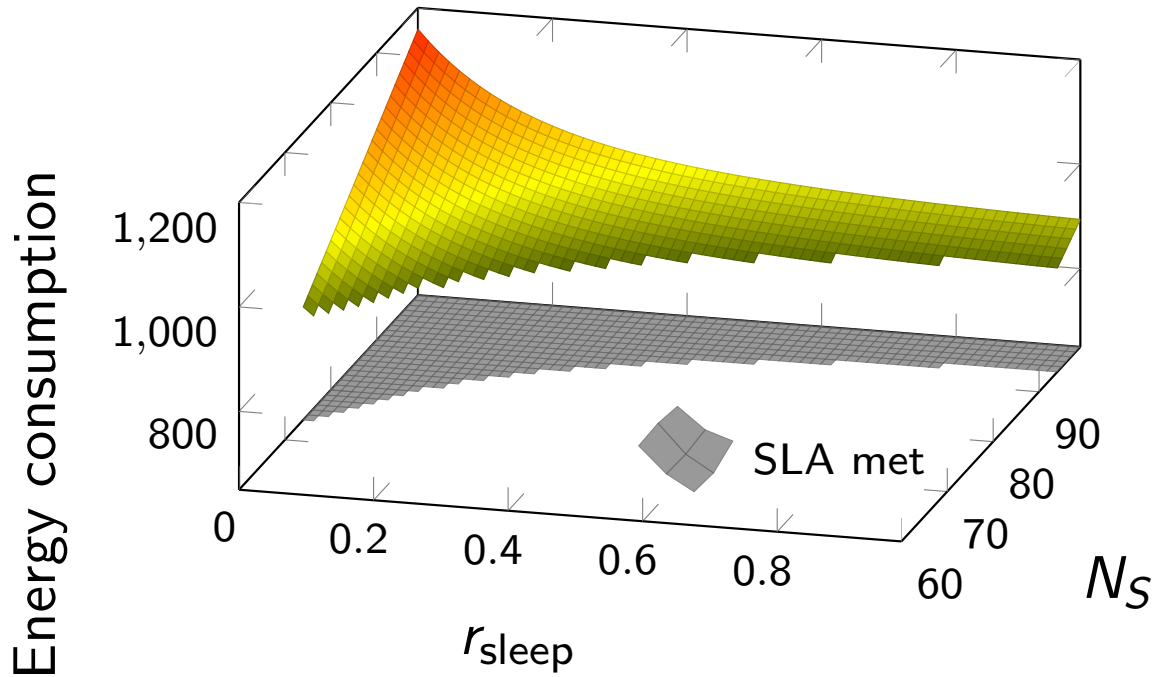


Individual passage-time SLA: clients must finish in at most $7s$ $\geq 99\%$ of the time

Trade-off between energy and performance

Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Minimise energy consumption while satisfying SLAs

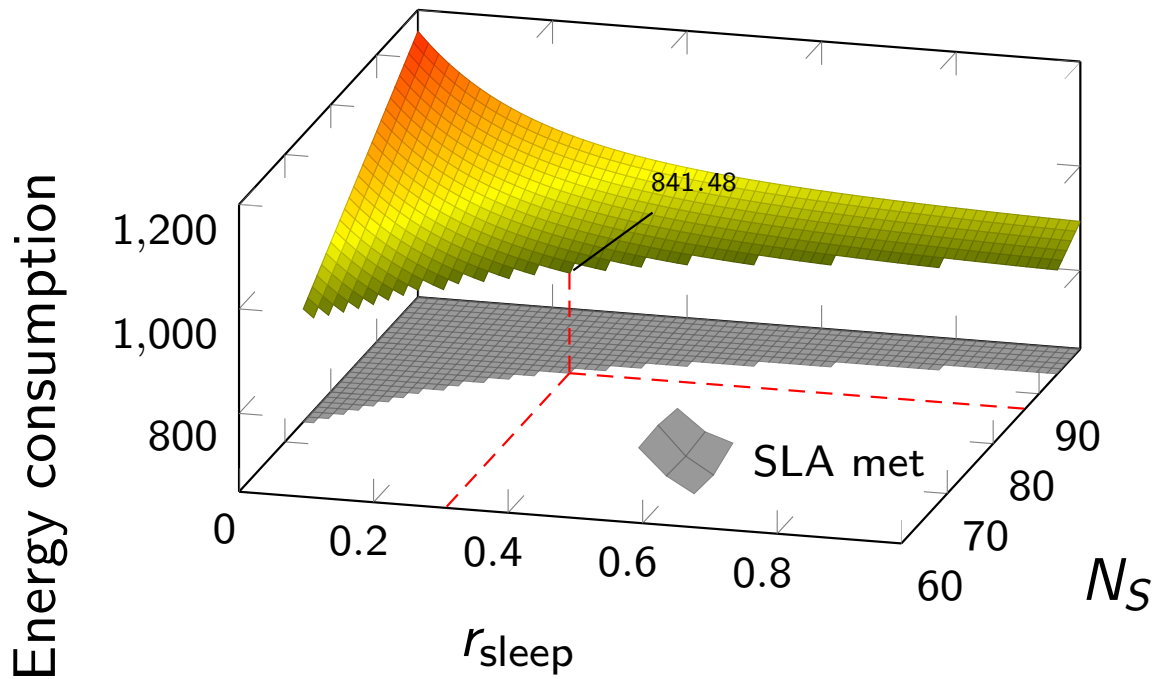


Individual passage-time SLA: clients must finish in at most $7s$ $\geq 99\%$ of the time

Trade-off between energy and performance

Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Minimise energy consumption while satisfying SLAs

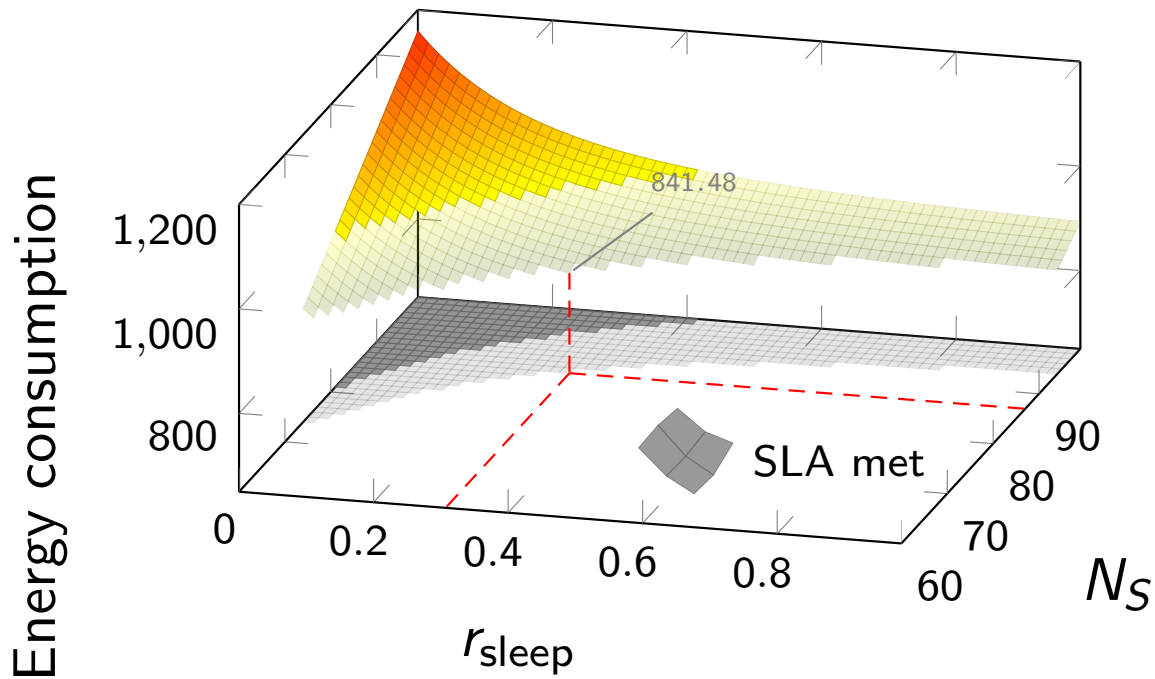


Individual passage-time SLA: clients must finish in at most $7s$ $\geq 99\%$ of the time

Trade-off between energy and performance

Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Minimise energy consumption while satisfying SLAs

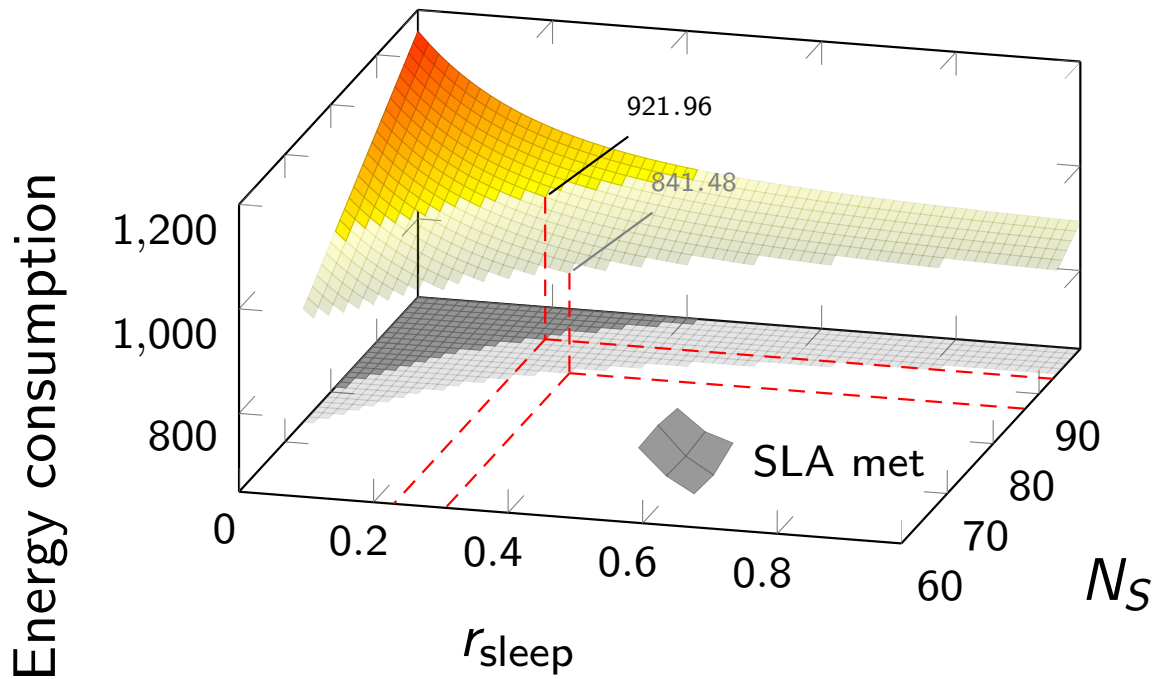


Individual passage-time SLA: clients must finish in at most $7s$ $\geq 99.5\%$ of the time

Trade-off between energy and performance

Scalable analysis allows exploration of many configurations (N_S, sleep rate)

Minimise energy consumption while satisfying SLAs



Individual passage-time SLA: clients must finish in at most $7s$ $\geq 99.5\%$ of the time

Non-Markovian models

Non-Markovian models

- Distributions more general than exponential are required to construct realistic models, for example:
  - Deterministic timeouts in protocols or hardware
  - Heavy-tailed service-time distributions

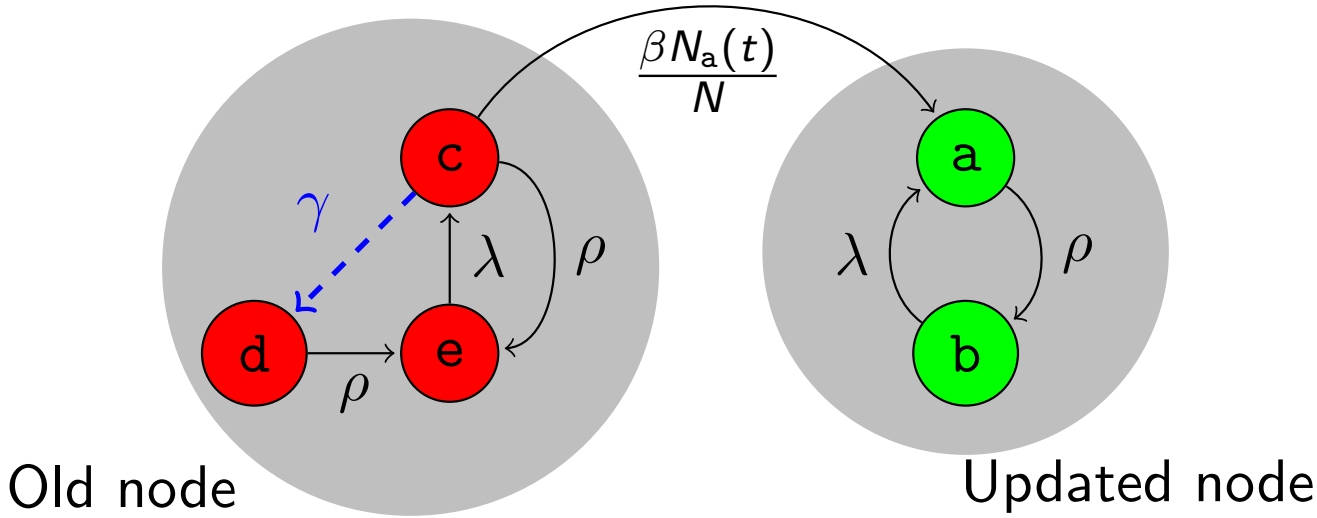
Non-Markovian models

- Distributions more general than exponential are required to construct realistic models, for example:
  - Deterministic timeouts in protocols or hardware
  - Heavy-tailed service-time distributions
- Phase-type approximation is one approach, but can lead to significant increase in a component's local state-space size
  - A 100-phase Erlang approximation to a deterministic distribution of duration 1 has a probability of about 32% of lying outside of $[0.9, 1.1]$

Non-Markovian models

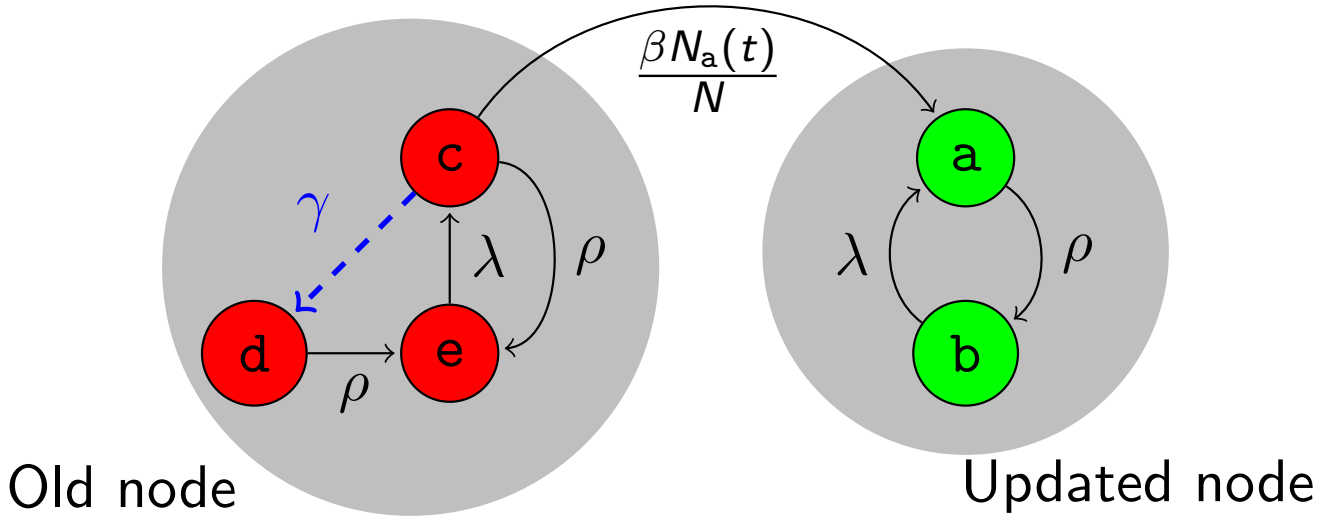
- Distributions more general than exponential are required to construct realistic models, for example:
  - Deterministic timeouts in protocols or hardware
  - Heavy-tailed service-time distributions
- Phase-type approximation is one approach, but can lead to significant increase in a component's local state-space size
  - A 100-phase Erlang approximation to a deterministic distribution of duration 1 has a probability of about 32% of lying outside of $[0.9, 1.1]$
- In the case of deterministic distributions, mean-field approach can be generalised using *delay differential equations*

Software update model with deterministic timeouts



[12] Richard A. Hayden. "Mean-field approximations for performance models with generally-timed transitions". In: *ACM SIGMETRICS Performance Evaluation Review* 39.3 (2011), pp. 119–121. DOI: 10.1145/2160803.2160877.

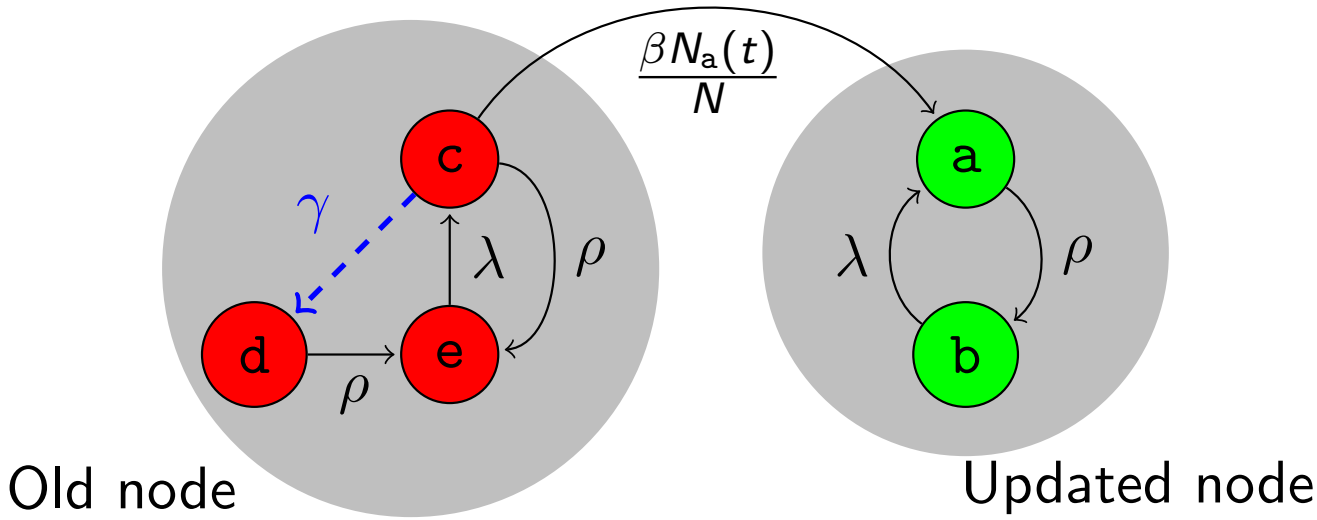
Software update model with deterministic timeouts



$$\dot{\mathbb{E}}[N_c(t)] = -\rho\mathbb{E}[N_c(t)] - \frac{\beta}{N}\mathbb{E}[N_c(t)N_a(t)] + \lambda\mathbb{E}[N_e(t)]$$
$$- \mathbb{E}\left[\underbrace{\mathbf{1}_{\{t\geq\gamma\}}\lambda N_e(t-\gamma)}_{\substack{\text{Rate of determ.}\\ \text{clocks starting at } t-\gamma}} \exp\left(-\int_{t-\gamma}^{t}\frac{\beta N_a(s)}{N}\,ds\right)\exp(-\rho\gamma)\right]$$

[12] Richard A. Hayden. "Mean-field approximations for performance models with generally-timed transitions". In: *ACM SIGMETRICS Performance Evaluation Review* 39.3 (2011), pp. 119–121. DOI: 10.1145/2160803.2160877.

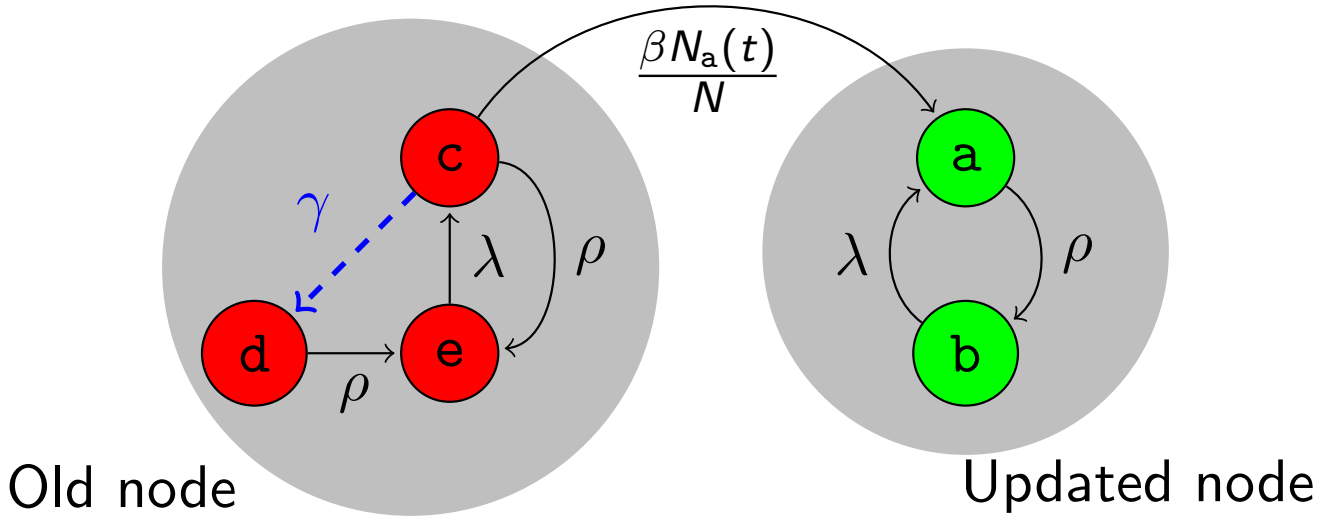
Software update model with deterministic timeouts



$$\dot{\mathbb{E}}[N_c(t)] = -\rho\mathbb{E}[N_c(t)] - \frac{\beta}{N}\mathbb{E}[N_c(t)N_a(t)] + \lambda\mathbb{E}[N_e(t)]$$
$$- \mathbb{E}\left[\mathbf{1}_{\{t\geq\gamma\}}\lambda N_e(t-\gamma)\underbrace{\exp\left(-\int_{t-\gamma}^{t}\frac{\beta N_a(s)}{N}\,ds\right)\exp(-\rho\gamma)}_{\text{Prob. that timeout occurs before node updated or went off}}\right]$$

[12] Richard A. Hayden. "Mean-field approximations for performance models with generally-timed transitions". In: *ACM SIGMETRICS Performance Evaluation Review* 39.3 (2011), pp. 119–121. DOI: 10.1145/2160803.2160877.

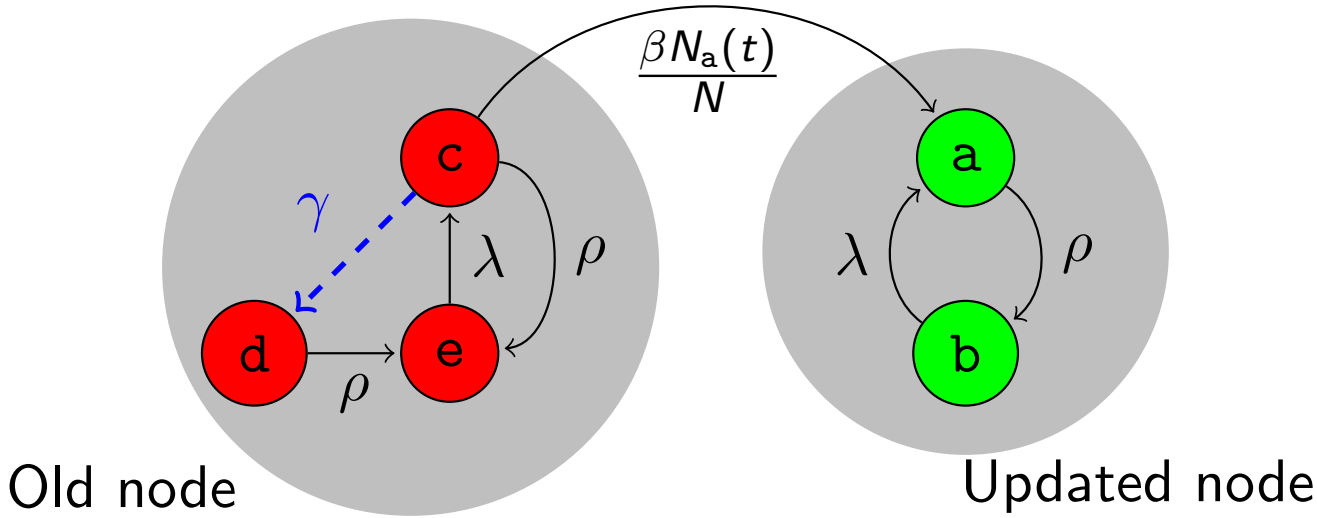
Software update model with deterministic timeouts



$$\dot{\mathbb{E}}[N_c(t)] \approx -\rho\mathbb{E}[N_c(t)] - \frac{\beta}{N}\mathbb{E}[N_c(t)]\mathbb{E}[N_a(t)] + \lambda\mathbb{E}[N_e(t)]$$
$$- \mathbf{1}_{\{t\geq\gamma\}}\lambda\mathbb{E}[N_e(t-\gamma)]\exp\left(-\int_{t-\gamma}^{t}\frac{\beta\mathbb{E}[N_a(s)]}{N}\,\mathrm{d}s\right)\exp(-\rho\gamma)$$

[12] Richard A. Hayden. "Mean-field approximations for performance models with generally-timed transitions". In: *ACM SIGMETRICS Performance Evaluation Review* 39.3 (2011), pp. 119–121. DOI: 10.1145/2160803.2160877.

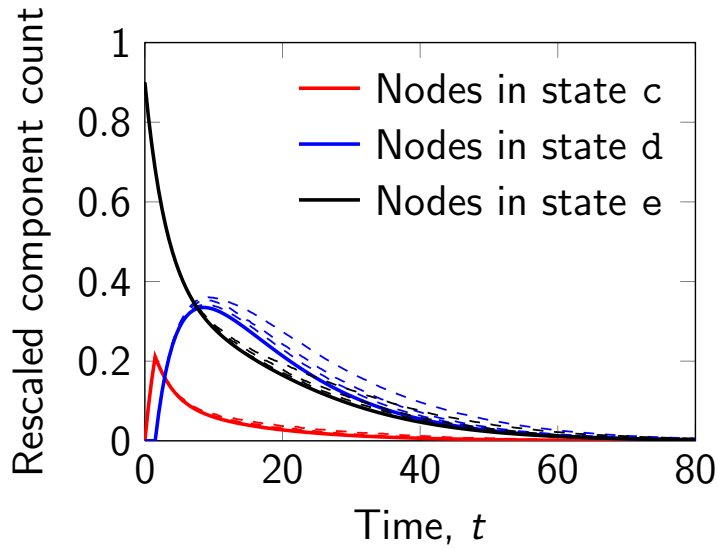
Software update model with deterministic timeouts

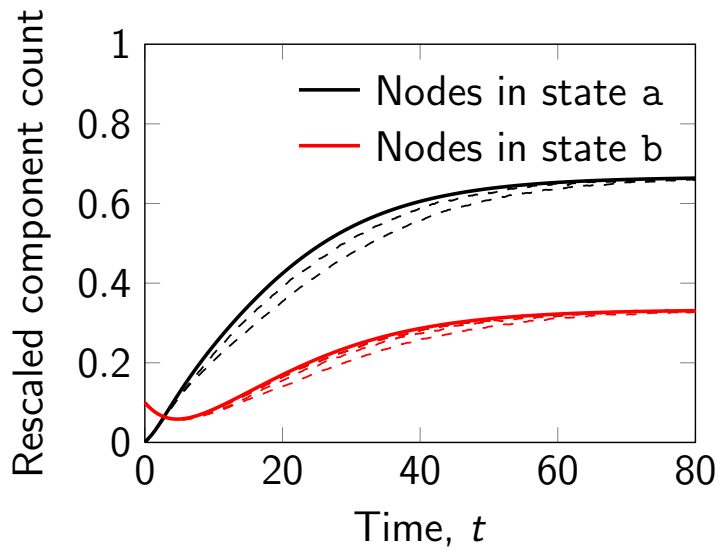


$$\dot{v}_c(t) = -\rho v_c(t) - \frac{\beta}{N} v_c(t) v_a(t) + \lambda v_e(t)$$
$$- \mathbf{1}_{t \geq \gamma} \lambda v_e(t-\gamma) \exp\left(- \int_{t-\gamma}^{t} \frac{\beta v_a(s)}{N} \, \mathrm{d}s \right) \exp(-\rho\gamma)$$

[12] Richard A. Hayden. "Mean-field approximations for performance models with generally-timed transitions". In: *ACM SIGMETRICS Performance Evaluation Review* 39.3 (2011), pp. 119–121. DOI: 10.1145/2160803.2160877.

Software update model with deterministic timeouts





Summary

Fluid analysis provides a scalable analysis framework for massively-parallel performance models, that is able to capture:

- Arbitrary moments of component counts
- Passage-time measures
- Accumulated reward measures
- Certain forms of non-Markovian timing

with implementation in the freely-available GPA tool[1]

1 http://code.google.com/p/gpanalyser/

Thank you![2]

2 Many thanks to Richard Hayden and Anton Stefanek for their expertise with pgf and pgfplots and their help with this presentation. They also did a substantial portion of the research!