SPAs for performance modelling: Lecture 9 — Hybrid Approximation

Jane Hillston

LFCS, School of Informatics
The University of Edinburgh
Scotland

18th April 2013

THE UNIVERSITY
of EDINBURGH

Outline

Outline

Introduction

All approximation techniques introduce errors and as modellers we should always be conscious of this.

Introduction

All approximation techniques introduce errors and as modellers we should always be conscious of this.

Generally we are willing to trade some accuracy for efficiency or even tractability.

Introduction

All approximation techniques introduce errors and as modellers we should always be conscious of this.

Generally we are willing to trade some accuracy for efficiency or even tractability.

But we should remain aware that there will be cases for which the approach is inappropriate because too much error is introduced.

Adequacy of fluid approximation

Recall that Kurtz's theorem only tells us that the behaviour of the deterministic system (ODEs) and stochastic system (CTMC) will be the same when the total population approaches infinity in a scaled way.

Adequacy of fluid approximation

Recall that Kurtz's theorem only tells us that the behaviour of the deterministic system (ODEs) and stochastic system (CTMC) will be the same when the total population approaches infinity in a scaled way.

It does not tell us how large an N is big enough to count as infinity.

Adequacy of fluid approximation

Recall that Kurtz's theorem only tells us that the behaviour of the deterministic system (ODEs) and stochastic system (CTMC) will be the same when the total population approaches infinity in a scaled way.

It does not tell us how large an N is big enough to count as infinity.

Moreover the existing error bounds by Darling and Norris are extremely loose, and so do not help us to predict how big N should be.

Fluid approximation of moments

In previous lectures we have seen how a set of ODEs can be derived from a PEPA model.

Fluid approximation of moments

In previous lectures we have seen how a set of ODEs can be derived from a PEPA model.

The ODEs approximate the expectation of the population counts for each derivative, i.e. the first moment.

Fluid approximation of moments

In previous lectures we have seen how a set of ODEs can be derived from a PEPA model.

The ODEs approximate the expectation of the population counts for each derivative, i.e. the first moment.

Bradley and Hayden from Imperial College have generalised this approach to construct sets of ODEs to also approximate higher moments.

Fluid approximation of moments

In previous lectures we have seen how a set of ODEs can be derived from a PEPA model.

The ODEs approximate the expectation of the population counts for each derivative, i.e. the first moment.

Bradley and Hayden from Imperial College have generalised this approach to construct sets of ODEs to also approximate higher moments.

This offers more information about the distribution of the population count, rather than simply its expectation.

Example model

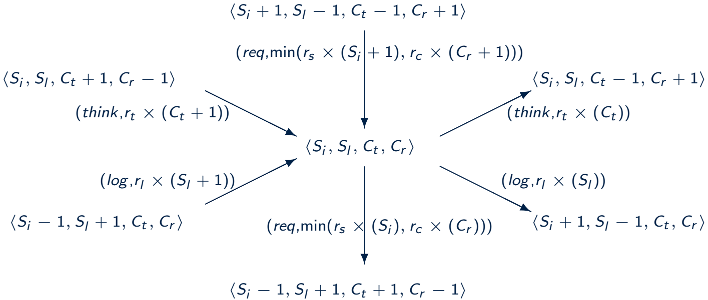
$$
\begin{aligned}
C_{thinking} &\stackrel{def}{=} (think, r_t).C_{requesting} \\
C_{requesting} &\stackrel{def}{=} (req, r_c).C_{thinking} \\
S_{idle} &\stackrel{def}{=} (req, r_s).S_{logging} \\
S_{logging} &\stackrel{def}{=} (log, r_l).S_{idle} \\
CS &\stackrel{def}{=} S_{idle}[n_s] \underset{\{req\}}{\bowtie} C_{thinking}[n_c]
\end{aligned}
$$

Example model

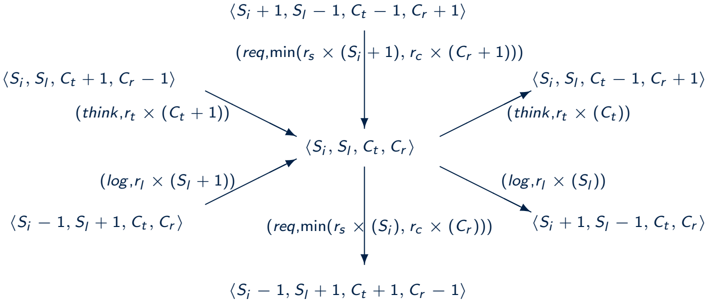
$$
\begin{aligned}
C_{thinking} &\stackrel{def}{=} (think, r_t).C_{requesting} \\
C_{requesting} &\stackrel{def}{=} (req, r_c).C_{thinking} \\
S_{idle} &\stackrel{def}{=} (req, r_s).S_{logging} \\
S_{logging} &\stackrel{def}{=} (log, r_l).S_{idle} \\
CS &\stackrel{def}{=} S_{idle}[n_s] \underset{\{req\}}{\bowtie} C_{thinking}[n_c]
\end{aligned}
$$

For some parametrizations of this model, the model's behaviour can accurately be characterized by the fluid flow approximations of its moments. However, for others, the moments are not sufficient to capture the model's behaviour, highlighting the danger of relying only on the results of fluid flow analysis.

Transitions into and out of a typical state



Transitions into and out of a typical state



This view of the system is the basis of generating the ODEs for the moments of the system.

Chapman-Kolmogorov equations

$$
\begin{aligned}
\frac{d\ p_{\langle S_i,S_l,C_t,C_r\rangle}(t)}{dt} = \\
& + (C_t + 1) \times r_t \times p_{\langle S_i,S_l,C_t+1,C_r-1\rangle}(t) \\
& + (S_l + 1) \times r_l \times p_{\langle S_i-1,S_l+1,C_t,C_r\rangle}(t) \\
& + \min(\ (S_i + 1) \times r_s\ , (C_r + 1) \times r_c\ ) \times p_{\langle S_i+1,S_l-1,C_t-1,C_r+1\rangle}(t) \\
& - \min(\ S_i \times r_s\ , C_r \times r_c\ ) \times p_{\langle S_i,S_l,C_t,C_r\rangle}(t) \\
& - S_l \times r_l \times p_{\langle S_i,S_l,C_t,C_r\rangle}(t) \\
& - C_t \times r_t \times p_{\langle S_i,S_l,C_t,C_r\rangle}(t).
\end{aligned}
$$

One variable/equation for every state of the system.

First moment approximation

$$
\begin{aligned}
\frac{d\ \mathbb{E}[C_r](t)}{dt} &= \sum_{\langle S_i,\, S_l,\, C_t,\, C_l\rangle \in \mathbb{D}} \frac{C_r \times p_{\langle S_i, S_l, C_t, C_r\rangle}(t)}{d\ t} \\
&= + \sum_{\langle S_i,\, S_l,\, C_t,\, C_l\rangle \in \mathbb{D}} C_t \times r_t \times p_{\langle S_i, S_l, C_t, C_r\rangle}(t) \\
&\quad - \sum_{\langle S_i,\, S_l,\, C_t,\, C_l\rangle \in \mathbb{D}} \min(S_i \times r_s, C_r \times r_c) \times p_{\langle S_i, S_l, C_t, C_r\rangle}(t) \\
&= + r_t \times \mathbb{E}[C_t](t) - \mathbb{E}[\min(S_i \times r_s, C_r \times r_c)](t).
\end{aligned}
$$

One variable/equation for each component of the state vector.

First moment approximation

$$
\begin{aligned}
\frac{d\ \mathbb{E}[C_r](t)}{dt} &= \sum_{\langle S_i,\, S_l,\, C_t,\, C_l \rangle \in \mathbb{D}} \frac{C_r \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t)}{d\ t} \\
&= + \sum_{\langle S_i,\, S_l,\, C_t,\, C_l \rangle \in \mathbb{D}} C_t \times r_t \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t) \\
&\quad - \sum_{\langle S_i,\, S_l,\, C_t,\, C_l \rangle \in \mathbb{D}} \min(S_i \times r_s, C_r \times r_c) \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t) \\
&= + r_t \times \mathbb{E}[C_t](t) - \mathbb{E}[\min(S_i \times r_s, C_r \times r_c)](t).
\end{aligned}
$$

One variable/equation for each component of the state vector.

Note $\mathbb{E}[\min(S_i \times r_s, C_r \times r_c)]$.

First moment approximation

Approximating $\mathbb{E}[\min(S_i \times r_s, C_r \times r_c)]$ with $\min(\mathbb{E}[S_i \times r_s], \mathbb{E}[C_r \times r_c]) = \min(r_s \times \mathbb{E}[S_i], r_c \times \mathbb{E}[C_r])$.

$$
\begin{aligned}
\frac{d\ \mathbb{E}' C_t(t)}{dt} &= -r_t \times \mathbb{E}' C_t(t) + \min(r_c \times \mathbb{E}' C_r(t), r_s \times \mathbb{E}' S_i(t)) \\
\frac{d\ \mathbb{E}' C_r(t)}{dt} &= -\min(r_c \times \mathbb{E}' C_r(t), r_s \times \mathbb{E}' S_i(t)) + r_t \times \mathbb{E}' C_t(t) \\
\frac{d\ \mathbb{E}' S_i(t)}{dt} &= -\min(r_c \times \mathbb{E}' C_r(t), r_s \times \mathbb{E}' S_i(t)) + r_l \times \mathbb{E}' S_l(t) \\
\frac{d\ \mathbb{E}' S_l(t)}{dt} &= +\min(r_c \times \mathbb{E}' C_r(t), r_s \times \mathbb{E}' S_i(t)) - r_l \times \mathbb{E}' S_l(t)
\end{aligned}
$$

One variable/equation for each component of the state vector.

Parameterisation 1

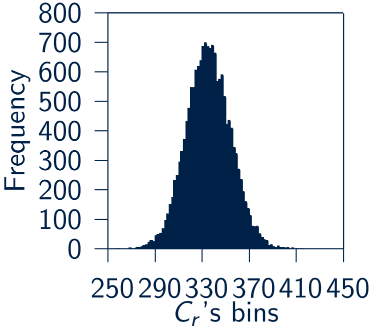
$$
\begin{aligned}
C_{thinking} &\stackrel{def}{=} (think, r_t).C_{requesting} \\
C_{requesting} &\stackrel{def}{=} (req, r_c).C_{thinking} \\
S_{idle} &\stackrel{def}{=} (req, r_s).S_{logging} \\
S_{logging} &\stackrel{def}{=} (log, r_l).S_{idle} \\
CS &\stackrel{def}{=} S_{idle}[n_s] \underset{\{req\}}{\bowtie} C_{thinking}[n_c]
\end{aligned}
$$

Parameterisation 1

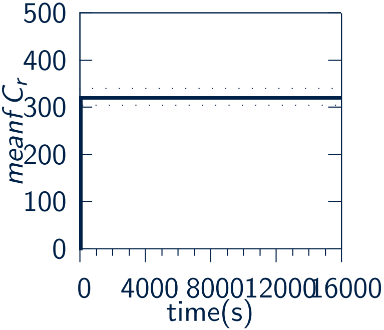
$$C_{thinking} \stackrel{def}{=} (think, r_t).C_{requesting}$$
$$C_{requesting} \stackrel{def}{=} (req, r_c).C_{thinking}$$
$$S_{idle} \stackrel{def}{=} (req, r_s).S_{logging}$$
$$S_{logging} \stackrel{def}{=} (log, r_l).S_{idle}$$
$$CS \stackrel{def}{=} S_{idle}[n_s] \underset{\{req\}}{\bowtie} C_{thinking}[n_c]$$

Parameter	Value	Description
r_s	500	On average, it takes 1/500th of an hour for a server to initiate a communication link with a client.
r_l	120	On average, it takes 1/120th of an hour for a server to process a request.
c_r	2	On average, it takes 1/2 of an hour for a client to initiate a communication link with a server.
c_t	0.06	On average, it takes 1/0.06th of a hours for a client to think.
n_s	10	Total population of servers.
n_c	10000	Total population of clients.

Distribution of C_r found by SSA



Mean and std deviation found via fluid approximation



Some numerical results

	n_s	3	4	5	6	7	8	9	10
$\mathbb{E}[C_r]$	F.F.A.	5645	4193	2741	1290.3	322	322	322	322
	M.C.	5644	4192	2740	1290	490	384	349	335
	Err.(%)	0.01	0.01	0.03	0.04	34	16	7.7	3.8
$\sigma[C_r]$	F.F.A.	60.62	70	78.26	85.73	17.66	17.66	17.66	17.66
	M.C.	60.45	69.45	78.79	86.5	36.90	23.49	19.84	18.72
	Err.(%)	0.26	0.25	0.67	0.9	52.3	25.20	11.44	5.6

Some numerical results

	n_s	3	4	5	6	7	8	9	10
$\mathbb{E}[C_r]$	F.F.A.	5645	4193	2741	1290.3	322	322	322	322
	M.C.	5644	4192	2740	1290	490	384	349	335
	Err.(%)	0.01	0.01	0.03	0.04	34	16	7.7	3.8
$\sigma[C_r]$	F.F.A.	60.62	70	78.26	85.73	17.66	17.66	17.66	17.66
	M.C.	60.45	69.45	78.79	86.5	36.90	23.49	19.84	18.72
	Err.(%)	0.26	0.25	0.67	0.9	52.3	25.20	11.44	5.6

Modified model

$$
\begin{aligned}
C_{thinking} &\stackrel{def}{=} (think, r_t).C_{requesting} \\
C_{requesting} &\stackrel{def}{=} (req, r_c).C_{thinking} \\
S_{idle} &\stackrel{def}{=} (req, r_s).S_{logging} + (brk, r_b).S_{broken} \\
S_{logging} &\stackrel{def}{=} (log, r_l).S_{idle} \\
S_{broken} &\stackrel{def}{=} (fix, r_f).S_{idle} \\
CS &\stackrel{def}{=} S_{idle}[n_s] \underset{\{req\}}{\bowtie} C_{thinking}[n_c]
\end{aligned}
$$

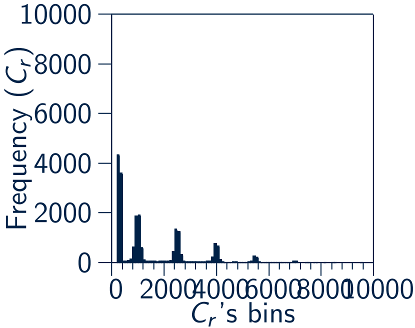
More numerical results

	n_s	3	4	5	6	7	8	9	10	12	14	16	18
$\mathbb{E}[C_r]$	F.F.A.	7119	6159	5199	4239	3279	2319	1359	399	243	243	243	243
	M. C.	7156	6177	5236	4295	3387	2599	1975	1460	843	533	378	309
	Err.(%)	0.6	0.2	0.7	1.3	3.18	10.77	31.1	72.6	71.1	54.4	35.7	21
$\sigma[C_r]$	F.F.A.	1240	1432	1601	1753	1894	2025	2148	959	15.42	15.42	15.42	15.42
	M.C.	1245	1420	1609	1758	1808	1792	1656	1456	1048	713	470	314
	Err.(%)	0.42	0.79	0.52	0.25	4.7	13	29	34.1	98	97	96	95

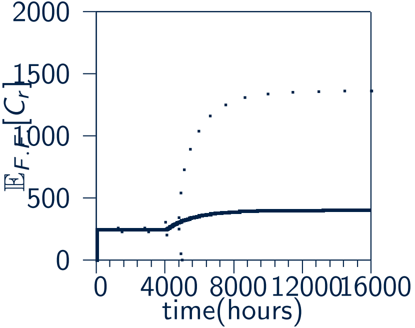
More numerical results

	n_s	3	4	5	6	7	8	9	10	12	14	16	18
$\mathbb{E}[C_r]$	F.F.A.	7119	6159	5199	4239	3279	2319	1359	399	243	243	243	243
	M. C.	7156	6177	5236	4295	3387	2599	1975	1460	843	533	378	309
	Err.(%)	0.6	0.2	0.7	1.3	3.18	10.77	31.1	72.6	71.1	54.4	35.7	21
$\sigma[C_r]$	F.F.A.	1240	1432	1601	1753	1894	2025	2148	959	15.42	15.42	15.42	15.42
	M.C.	1245	1420	1609	1758	1808	1792	1656	1456	1048	713	470	314
	Err.(%)	0.42	0.79	0.52	0.25	4.7	13	29	34.1	98	97	96	95

Distribution of C_r found by SSA



Mean and std deviation found via fluid approximation



Outline

Motivation

We have seen that the fluid approximation can be an accurate way to estimate the population counts and some performance measures for some systems in which we have large populations interacting.

Motivation

We have seen that the fluid approximation can be an accurate way to estimate the population counts and some performance measures for some systems in which we have large populations interacting.

However, we have also seen that there are cases where this technique should not be used because it will lead to inaccurate estimates of the performance of the system, and then simulation becomes the best way to tackle the system.

Motivation: combining the approaches

- The ODE-based solution is much more computationally efficient than stochastic simulation (even when using Gillespie's efficient SSA).

Motivation: combining the approaches

- The ODE-based solution is much more computationally efficient than stochastic simulation (even when using Gillespie's efficient SSA).
- Typically problems arise when there is a mix of some large populations and some small, or some fast actions and some slow.

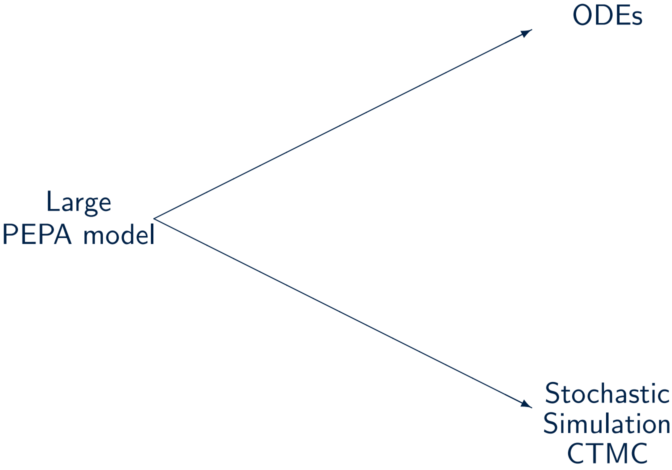
Motivation: combining the approaches

- The ODE-based solution is much more computationally efficient than stochastic simulation (even when using Gillespie's efficient SSA).
- Typically problems arise when there is a mix of some large populations and some small, or some fast actions and some slow.
- So it is natural to consider if there might be a way to combine the approaches.

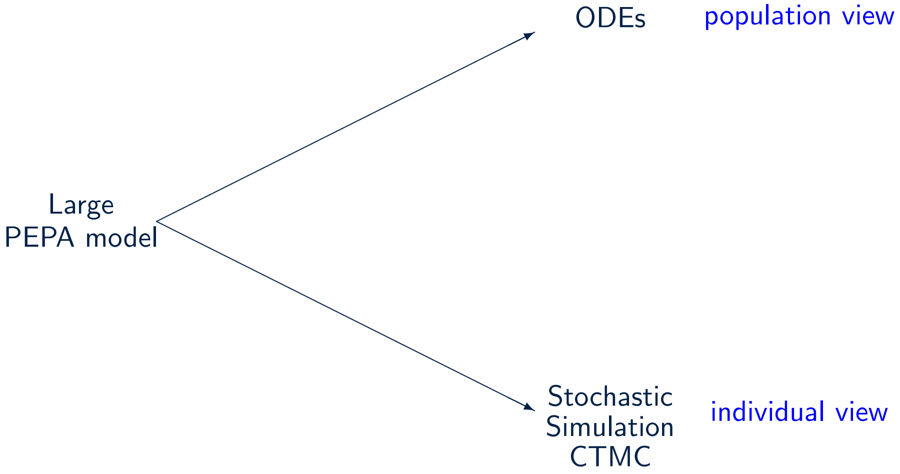
Motivation: combining the approaches

- The ODE-based solution is much more computationally efficient than stochastic simulation (even when using Gillespie's efficient SSA).
- Typically problems arise when there is a mix of some large populations and some small, or some fast actions and some slow.
- So it is natural to consider if there might be a way to combine the approaches.
- In particular we aim to resort to the less efficient discrete approach for those parts of the model where it is strictly necessary.

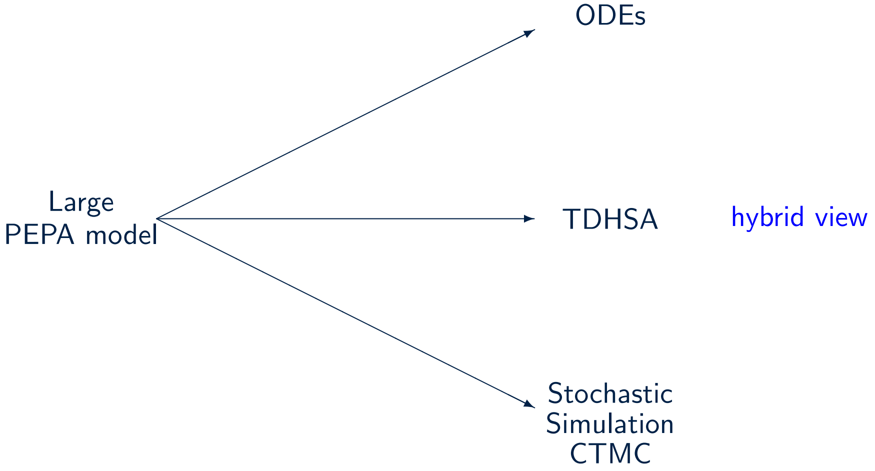
Motivation: Alternative Representations



Motivation: Alternative Representations



Motivation: Alternative Representations



Making a hybrid approximation

Hybrid is a term which is used to refer to models in which some state variables are discrete and some state variables are continuous.

Making a hybrid approximation

Hybrid is a term which is used to refer to models in which some state variables are discrete and some state variables are continuous.

Of course in a PEPA model all state variable are discrete — they are the population counts which tell us in each state how many of each derivative type we have.

Making a hybrid approximation

Hybrid is a term which is used to refer to models in which some state variables are discrete and some state variables are continuous.

Of course in a PEPA model all state variable are discrete — they are the population counts which tell us in each state how many of each derivative type we have.

In the fluid approximation we choose to approximate all of these discrete variables by continuous ones to obtain a set of ODEs.

Making a hybrid approximation

Hybrid is a term which is used to refer to models in which some state variables are discrete and some state variables are continuous.

Of course in a PEPA model all state variable are discrete — they are the population counts which tell us in each state how many of each derivative type we have.

In the fluid approximation we choose to approximate all of these discrete variables by continuous ones to obtain a set of ODEs.

In the hybrid approximation we choose to approximate some populations as continuous whilst keeping the others discrete.

Making a hybrid approximation

Hybrid is a term which is used to refer to models in which some state variables are discrete and some state variables are continuous.

Of course in a PEPA model all state variable are discrete — they are the population counts which tell us in each state how many of each derivative type we have.

In the fluid approximation we choose to approximate all of these discrete variables by continuous ones to obtain a set of ODEs.

In the hybrid approximation we choose to approximate some populations as continuous whilst keeping the others discrete.

The result is a set of discrete states each of which has an associated set of ODEs.

PDMPs and TDHSA

There are many different formalisms that may be used to specify this type of hybrid system (discrete states with sets of ODEs).

PDMPs and TDHSA

There are many different formalisms that may be used to specify this type of hybrid system (discrete states with sets of ODEs).

Formally we will work in terms of stochastic processes termed Piecewise Deterministic Markov Processes (PDMP).

PDMPs and TDHSA

There are many different formalisms that may be used to specify this type of hybrid system (discrete states with sets of ODEs).

Formally we will work in terms of stochastic processes termed Piecewise Deterministic Markov Processes (PDMP).

But these are little difficult to work with directly, so we will use a form of automata, called Transition Driven Stochastic Hybrid Automata (TDSHAs) as an intermediary.

PDMPs and TDHSA

There are many different formalisms that may be used to specify this type of hybrid system (discrete states with sets of ODEs).

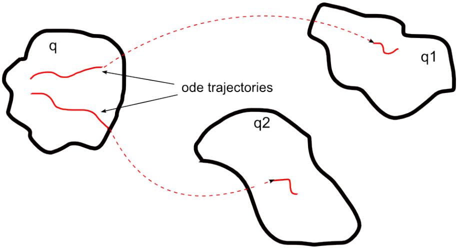
Formally we will work in terms of stochastic processes termed Piecewise Deterministic Markov Processes (PDMP).

But these are little difficult to work with directly, so we will use a form of automata, called Transition Driven Stochastic Hybrid Automata (TDSHAs) as an intermediary.

PEPA $\longrightarrow$ TDSHA $\longrightarrow$ PDMP

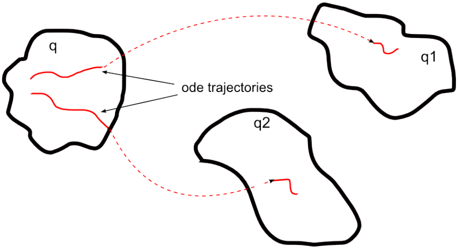
Piecewise deterministic Markov processes

- class of stochastic processes
- continuous trajectories over subsets of $\mathbb{R}^{|\mathbf{X}|}$
- instantaneous jumps at boundaries of regions
- stochastic jumps when guards are true



Piecewise deterministic Markov processes

- class of stochastic processes
- continuous trajectories over subsets of $\mathbb{R}^{|\mathbf{X}|}$
- instantaneous jumps at boundaries of regions
- stochastic jumps when guards are true



- jumps to boundaries are prohibited

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$
- set of events/actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$

Transition-driven stochastic hybrid automata (TDSHA)

- subset of piecewise deterministic Markov processes (PDMPs)
- set of (control) modes: $Q = \{q_1, \ldots, q_m\}$
- set of variables: $\mathbf{X} = \{X_1, \ldots, X_n\}$
- set of events/actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$
- initial state: $(q, (x_1, \ldots, x_n))$

Transition-driven stochastic hybrid automata (TDSHA)

- instantaneous transitions
  - source mode, target mode, event name
  - guard: activation condition over variables
  - reset: function determining new values of variables
  - priority/weight: to resolve non-determinism

Transition-driven stochastic hybrid automata (TDSHA)

- instantaneous transitions
  - source mode, target mode, event name
  - guard: activation condition over variables
  - reset: function determining new values of variables
  - priority/weight: to resolve non-determinism
- stochastic transitions
  - source mode, target mode, event name
  - rate: function defining speed of transition
  - guard: activation condition over variables
  - reset: function determining new values of variables

Transition-driven stochastic hybrid automata (TDSHA)

- instantaneous transitions
  - source mode, target mode, event name
  - guard: activation condition over variables
  - reset: function determining new values of variables
  - priority/weight: to resolve non-determinism
- stochastic transitions
  - source mode, target mode, event name
  - rate: function defining speed of transition
  - guard: activation condition over variables
  - reset: function determining new values of variables
- continuous transitions (flows)
  - source mode
  - vector specifying variables involved
  - Lipschitz continuous function

Transition-driven stochastic hybrid automata (TDSHA)

- continuous behaviour in a mode
  - consider all continuous transitions in that mode
  - trajectory is given by solution of $d\mathbf{X}/dt = \sum s \cdot f(\mathbf{X})$

Transition-driven stochastic hybrid automata (TDSHA)

- continuous behaviour in a mode
  - consider all continuous transitions in that mode
  - trajectory is given by solution of $d\mathbf{X}/dt = \sum s \cdot f(\mathbf{X})$
- instantaneous behaviour: fire when guard becomes true

Transition-driven stochastic hybrid automata (TDSHA)

- continuous behaviour in a mode
  - consider all continuous transitions in that mode
  - trajectory is given by solution of $d\mathbf{X}/dt = \sum s \cdot f(\mathbf{X})$
- instantaneous behaviour: fire when guard becomes true
- stochastic behaviour: fire according to rate

Transition-driven stochastic hybrid automata (TDSHA)

- continuous behaviour in a mode
  - consider all continuous transitions in that mode
  - trajectory is given by solution of $d\mathbf{X}/dt = \sum s \cdot f(\mathbf{X})$
- instantaneous behaviour: fire when guard becomes true
- stochastic behaviour: fire according to rate
- product of TDSHAs
  - pairs of modes and union of variables
  - combining transitions
    (with conditions on resets and initial values)

TDSHA synchronised product

- $\mathcal{T} = \mathcal{T}_1 \oplus_L \mathcal{T}_2$ has $Q = Q_1 \times Q_2$ and $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$

TDSHA synchronised product

- $\mathcal{T} = \mathcal{T}_1 \oplus_L \mathcal{T}_2$ has $Q = Q_1 \times Q_2$ and $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$
- continuous transitions: extend vector to cover $\mathbf{X}$
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition from q_1 and q_2 with a and new function is PEPA cooperation rate (i.e. bounded capacity)

TDSHA synchronised product

- $\mathcal{T} = \mathcal{T}_1 \oplus_L \mathcal{T}_2$ has $Q = Q_1 \times Q_2$ and $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$
- continuous transitions: extend vector to cover $\mathbf{X}$
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition from q_1 and q_2 with a and new function is PEPA cooperation rate (i.e. bounded capacity)
- stochastic transitions:
  - $a \notin L$: (q_1, q_2) has every transition from q_1 and from q_2
  - $a \in L$: (q_1, q_2) has every transition that both q_1 and q_2 have with a, new rate is PEPA cooperation rate and conjunction of resets is taken

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$

- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$P \quad \stackrel{def}{=} \quad S_1 \underset{L_2}{\bowtie} S_2 \underset{L_3}{\bowtie} \cdots \underset{L_n}{\bowtie} S_n$$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$S_1 \qquad S_2 \qquad \cdots \qquad S_n$$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$\begin{array}{cccc} S_1 & S_2 & \cdots & S_n \\ \downarrow & \downarrow & & \downarrow \\ \mathcal{T}_1 & \mathcal{T}_2 & \cdots & \mathcal{T}_n \end{array}$$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q}(a_j, r_j).S'$
- mapping

$$S_1 \quad \underset{L_2}{\bowtie} \quad S_2 \quad \underset{L_3}{\bowtie} \quad \cdots \quad \underset{L_n}{\bowtie} \quad S_n$$

$$\mathcal{T}_1 \qquad \mathcal{T}_2 \qquad \cdots \qquad \mathcal{T}_n$$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q} (a_j, r_j).S'$
- mapping

$$S_1 \quad \underset{L_2}{\bowtie} \quad S_2 \quad \underset{L_3}{\bowtie} \quad \cdots \quad \underset{L_n}{\bowtie} \quad S_n$$

$$\mathcal{T}_1 \quad \oplus_{L_2} \quad \mathcal{T}_2 \quad \oplus_{L_3} \quad \cdots \quad \oplus_{L_n} \quad \mathcal{T}_n$$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q}(a_j, r_j).S'$
- mapping

$$S_1 \quad \underset{L_2}{\bowtie} \quad S_2 \quad \underset{L_3}{\bowtie} \quad \cdots \quad \underset{L_n}{\bowtie} \quad S_n$$

$$\mathcal{T} \quad = \quad \mathcal{T}_1 \quad \oplus_{L_2} \quad \mathcal{T}_2 \quad \oplus_{L_3} \quad \cdots \quad \oplus_{L_n} \quad \mathcal{T}_n$$

Overview of the mapping

- PEPA has two-level syntax
  - sequential components: $S ::= (a, r).S \mid S + S$
  - parallel components: $P ::= P \underset{L}{\bowtie} P \mid S$
- assume sequential components: $S = \sum_{j=1}^{q}(a_j, r_j).S'$
- mapping

$$P \stackrel{def}{=} S_1 \underset{L_2}{\bowtie} S_2 \underset{L_3}{\bowtie} \cdots \underset{L_n}{\bowtie} S_n$$

$$\downarrow$$

$$\mathcal{T} = \mathcal{T}_1 \oplus_{L_2} \mathcal{T}_2 \oplus_{L_3} \cdots \oplus_{L_n} \mathcal{T}_n$$

Mapping sequential components

A decision must be made with respect to each derivative of each component about whether its count is to be treated as a discrete or a continuous variable.

Mapping sequential components

A decision must be made with respect to each derivative of each component about whether its count is to be treated as a discrete or a continuous variable.

The decision will be based on a decision about each action type.

Mapping sequential components

A decision must be made with respect to each derivative of each component about whether its count is to be treated as a discrete or a continuous variable.

The decision will be based on a decision about each action type.

We rely on the modeller to decide for each action type whether it represents a continuous action or a discrete action.

Mapping sequential components

A decision must be made with respect to each derivative of each component about whether its count is to be treated as a discrete or a continuous variable.

The decision will be based on a decision about each action type.

We rely on the modeller to decide for each action type whether it represents a continuous action or a discrete action.

Any derivative that enables a continuous action will be treated as a continuous variable in the state vector.

A client/server system with breakdowns and repairs

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w
\end{aligned}
$$

$$
\begin{aligned}
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r
\end{aligned}
$$

$$
Sys \stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
$$

A client/server system with breakdowns and repairs

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w \\
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r \\
Sys &\stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
\end{aligned}
$$

State Representation

$$\omega = (\omega_{S_w}, \omega_{S_l}, \omega_{S_b}, \omega_{U_r}, \omega_{U_t})$$

Initial state is $(1, 0, 0, N, 0)$

In the discrete case

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w \\
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r \\
Sys &\stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
\end{aligned}
$$

Request action

$$(1,0,0,N,0) \xrightarrow{request, min(1 \times r_{reply}, N \times r_{req})} (0,1,0,N-1,1)$$

In the discrete case

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w \\
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r \\
Sys &\stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
\end{aligned}
$$

Break action

$$(1, 0, 0, N, 0) \xrightarrow{break, 1 \times r_{break}} (0, 0, 1, N, 0)$$

In the continuous case

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w \\
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r \\
Sys &\stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
\end{aligned}
$$

Request action

$$
\mathbf{x}(t) \xrightarrow{request, min(1 \times r_{reply}, N \times r_{req})} \mathbf{x}(t) + (-1, +1, 0, -1, +1)
$$

In the continuous case

$$\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w \\
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r \\
Sys &\stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
\end{aligned}$$

Break action

$$\mathbf{x}(t) \xrightarrow{break, 1 \times r_{break}} \mathbf{x}(t) + (-1, 0, +1, 0, 0)$$

In the continuous case

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w \\
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r \\
Sys &\stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
\end{aligned}
$$

ODE for S_w

$$
\frac{dx_{S_w}(t)}{dt} = -min(x_{S_w}(t) r_{reply}, x_{U_r}(t) r_{req} - s_{S_w}(t) r_{break} + x_{S_l}(t) r_{log} + x_{S_r}(t) r_{re}
$$

Hybrid interpretation

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, r_{reply}).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, r_{log}).S_w \\
S_b &\stackrel{def}{=} (repair, r_{repair}).S_w \\
U_r &\stackrel{def}{=} (request, r_{req}).U_t \\
U_t &\stackrel{def}{=} (think, r_{think}).U_r \\
Sys &\stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]
\end{aligned}
$$

We may assume that the activities *break* and *repair* occur at a much lower frequency and a much lower rate than the other activities in the model.

Hybrid interpretation

So for our hybrid approximation we treat these activities as discrete and the other activities as continuous.

Hybrid interpretation

So for our hybrid approximation we treat these activities as discrete and the other activities as continuous.

- $\mathcal{A}_c = \{request, log, think\}$

Hybrid interpretation

So for our hybrid approximation we treat these activities as discrete and the other activities as continuous.

- $\mathcal{A}_c = \{request, log, think\}$
- $\mathcal{A}_d = \{break, repair\}$

Hybrid interpretation

So for our hybrid approximation we treat these activities as discrete and the other activities as continuous.

- $\mathcal{A}_c = \{request, log, think\}$
- $\mathcal{A}_d = \{break, repair\}$
- $\mathbf{X} = (X_{S_w}, X_{S_l}, X_{U_r}, X_{U_t})$ (continuous variables)

Hybrid interpretation

So for our hybrid approximation we treat these activities as discrete and the other activities as continuous.

- $\mathcal{A}_c = \{request, log, think\}$
- $\mathcal{A}_d = \{break, repair\}$
- $\mathbf{X} = (X_{S_w}, X_{S_l}, X_{U_r}, X_{U_t})$ (continuous variables)
- $q_0 = (0, 2, N)$

Hybrid interpretation

So for our hybrid approximation we treat these activities as discrete and the other activities as continuous.

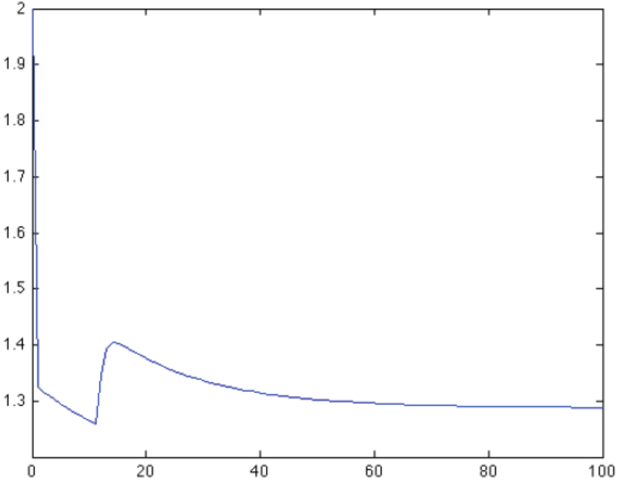
- $\mathcal{A}_c = \{request, log, think\}$
- $\mathcal{A}_d = \{break, repair\}$
- $\mathbf{X} = (X_{S_w}, X_{S_l}, X_{U_r}, X_{U_t})$ (continuous variables)
- $q_0 = (0, 2, N)$
- $q_0 = (1, 1, N)$

Hybrid interpretation

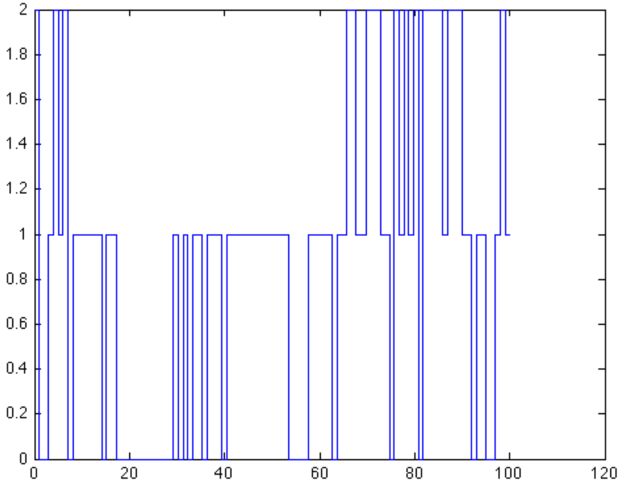
So for our hybrid approximation we treat these activities as discrete and the other activities as continuous.

- $\mathcal{A}_c = \{request, log, think\}$
- $\mathcal{A}_d = \{break, repair\}$
- $\mathbf{X} = (X_{S_w}, X_{S_l}, X_{U_r}, X_{U_t})$ (continuous variables)
- $q_0 = (0, 2, N)$
- $q_0 = (1, 1, N)$
- $q_0 = (2, 0, N)$

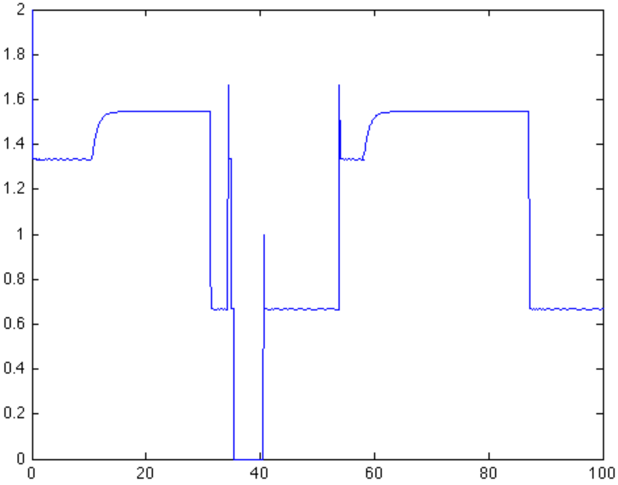
Fluid Dynamics: Working servers vs. time



Stochastic Dynamics: Working servers vs. time



Hybrid Dynamics: Working servers vs. time



Numerical Evaluation: set up

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, scale \times 1000).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, scale \times 2000).S_w \\
S_b &\stackrel{def}{=} (repair, 0.05).S_w \\
U_r &\stackrel{def}{=} (request, scale \times 100).U_t \\
U_t &\stackrel{def}{=} (think, scale \times 10).U_r \\
Sys &\stackrel{def}{=} S_w[N_S] \underset{\{request\}}{\bowtie} U_r[N_c]
\end{aligned}
$$

Numerical Evaluation: set up

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, scale \times 1000).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, scale \times 2000).S_w \\
S_b &\stackrel{def}{=} (repair, 0.05).S_w \\
U_r &\stackrel{def}{=} (request, scale \times 100).U_t \\
U_t &\stackrel{def}{=} (think, scale \times 10).U_r \\
Sys &\stackrel{def}{=} S_w[N_S] \underset{\{request\}}{\bowtie} U_r[N_c]
\end{aligned}
$$

- $scale \in \{0.1, 10.0, 100.0\}$

Numerical Evaluation: set up

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, scale \times 1000).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, scale \times 2000).S_w \\
S_b &\stackrel{def}{=} (repair, 0.05).S_w \\
U_r &\stackrel{def}{=} (request, scale \times 100).U_t \\
U_t &\stackrel{def}{=} (think, scale \times 10).U_r \\
Sys &\stackrel{def}{=} S_w[N_S] \underset{\{request\}}{\bowtie} U_r[N_c]
\end{aligned}
$$

- $scale \in \{0.1, 10.0, 100.0\}$
- $N_S \in \{2, 6\}$

Numerical Evaluation: set up

$$
\begin{aligned}
S_w &\stackrel{def}{=} (request, scale \times 1000).S_l + (break, r_{break}).S_b \\
S_l &\stackrel{def}{=} (log, scale \times 2000).S_w \\
S_b &\stackrel{def}{=} (repair, 0.05).S_w \\
U_r &\stackrel{def}{=} (request, scale \times 100).U_t \\
U_t &\stackrel{def}{=} (think, scale \times 10).U_r \\
Sys &\stackrel{def}{=} S_w[N_S] \underset{\{request\}}{\bowtie} U_r[N_c]
\end{aligned}
$$

- $scale \in \{0.1, 10.0, 100.0\}$
- $N_S \in \{2, 6\}$
- $N_C \in \{10, 100, 300\}$

Numerical Evaluation: set up

For each model configuration we calculated the steady-state probability of having 0 or 1 broken servers.

Errors were computed with respect to the numerical solution of the Markov chain.

Numerical Evaluation: results

N_c	N_s	scale	$\overline{X}^{S_b} = 0$	$\overline{X}^{S_b} = 1$	H	S	S/H
10	2	0.1	1.82%	3.58%	267	3	1.2E-2
100	2	0.1	0.67%	1.35%	1099	38	3.5E-2
300	2	0.1	0.70%	3.42%	529	69	1.3E-1
10	6	0.1	6.44%	0.52%	352	3	7.0E-3
100	6	0.1	2.35%	0.89%	566	18	3.1E-2
300	6	0.1	2.82%	1.53%	317	25	8.0E-2
10	2	10.0	0.54%	0.96%	547	253	4.6E-1
100	2	10.0	0.08%	0.21%	827	2618	3.2E+0
300	2	10.0	0.80%	3.20%	252	5092	2.0E+1
10	6	10.0	2.49%	2.64%	485	154	3.1E-1
100	6	10.0	3.86%	1.39%	623	1298	2.1E+0
300	6	10.0	1.30%	1.14%	876	5112	5.8E+0
10	2	100.0	0.13%	0.35%	204	3186	1.6E+1
100	2	100.0	0.35%	1.24%	589	20344	3.4E+1
300	2	100.0	0.01%	0.06%	438	51682	1.2E+2
10	6	100.0	2.19%	0.96%	217	1100	5.1E+0
100	6	100.0	2.14%	1.81%	301	13207	4.4E+1
300	6	100.0	0.09%	3.98%	592	39956	6.7E+1

Ongoing issues

- We currently assume that the modeller is responsible for partition action types and derivatives.

Ongoing issues

- We currently assume that the modeller is responsible for partition action types and derivatives.
- There is an issue of how to make transitions from continuous state to discrete states in the general case: we have a solution but it may not be the best one.

Illustrative example

Since both activities and components can be classified as discrete or continuous there are several different cases that can arise in the evolution of a model.

Illustrative example

Since both activities and components can be classified as discrete or continuous there are several different cases that can arise in the evolution of a model.

The example presented in the following slides is constructed to illustrate each of the different cases.

Illustrative example

Since both activities and components can be classified as discrete or continuous there are several different cases that can arise in the evolution of a model.

The example presented in the following slides is constructed to illustrate each of the different cases.

It illustrates some of the problems that can occur and our current solution to these problems.

Clients and servers example

- clients

$$
\begin{aligned}
\mathrm{Cr} &\stackrel{\mathit{def}}{=} (\mathrm{request}, r_{rq}).\mathrm{Ct} \\
\mathrm{Ct} &\stackrel{\mathit{def}}{=} (\mathrm{think}, r_{th}).\mathrm{Cr}
\end{aligned}
$$

Clients and servers example

- clients

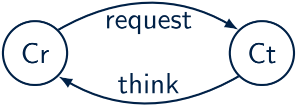
$$\begin{aligned} Cr &\stackrel{def}{=} (request, r_{rq}).Ct \\ Ct &\stackrel{def}{=} (think, r_{th}).Cr \end{aligned}$$

- servers

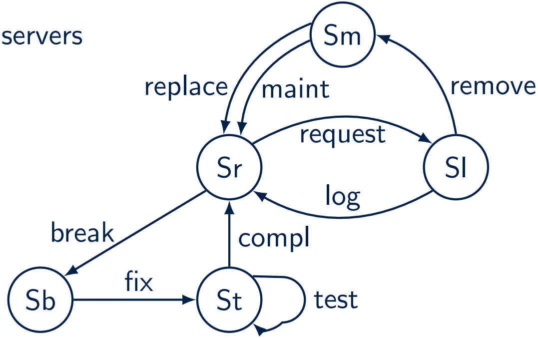
$$\begin{aligned} Sr &\stackrel{def}{=} (request, r_{rp}).Sl + (break, r_{bk}).Sb \\ Sl &\stackrel{def}{=} (log, r_{lg}).Sr + (remove, r_{rm}).Sm \\ Sm &\stackrel{def}{=} (maint, r_{mn}).Sr + (replace, r_{rc}).Sr \\ Sb &\stackrel{def}{=} (fix, r_{fx}).St \\ St &\stackrel{def}{=} (test, r_{ts}).St + (compl, r_{cm}).Sr \end{aligned}$$

Clients and servers example

- clients



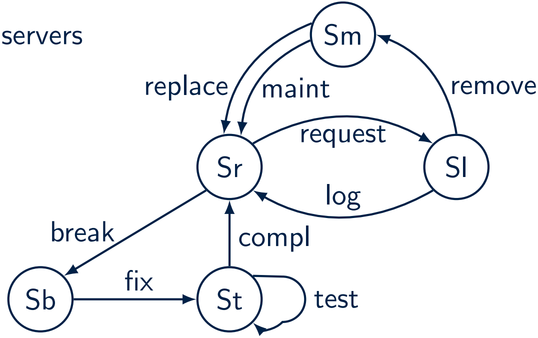
- servers



Clients and servers example

- clients

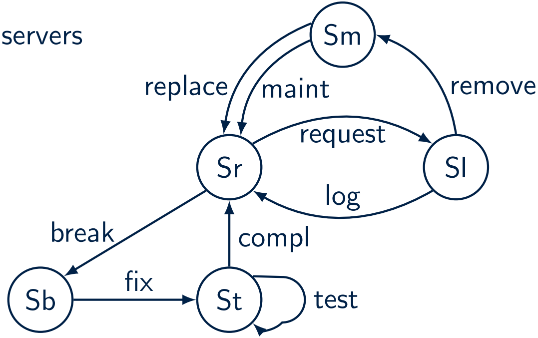
- servers



Clients and servers example

- clients

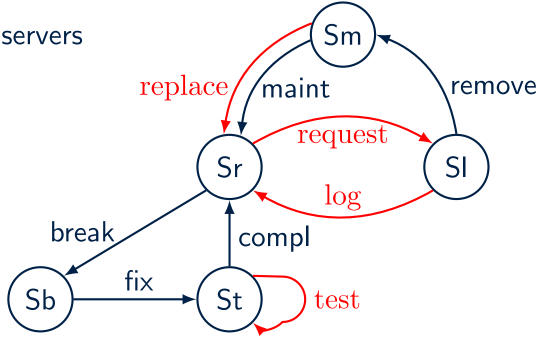
- servers



Clients and servers example

- clients

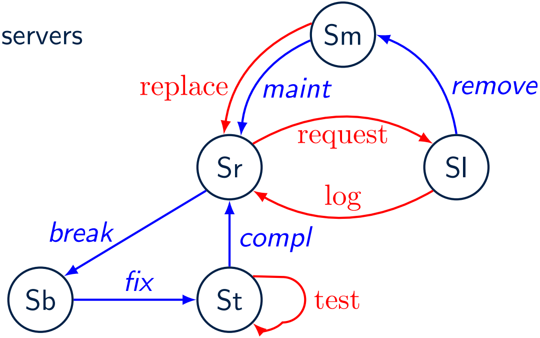
- servers



Clients and servers example

- clients

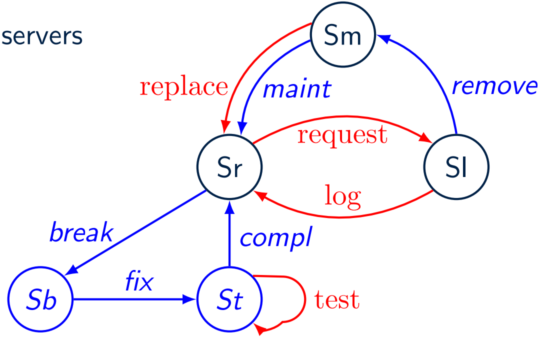
- servers



Clients and servers example

- clients

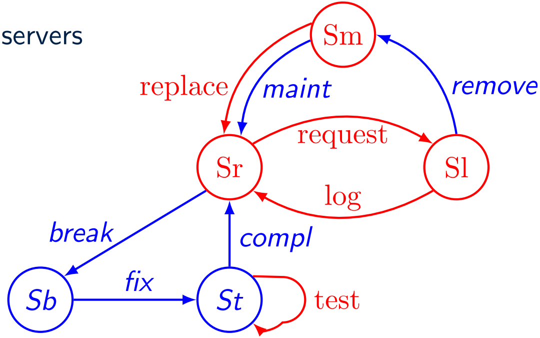
- servers



Clients and servers example

- clients

- servers



Clients and servers example

- clients

$$\mathrm{Cr} \stackrel{def}{=} (\mathrm{request}, r_{rq}).\mathrm{Ct}$$
$$\mathrm{Ct} \stackrel{def}{=} (\mathrm{think}, r_{th}).\mathrm{Cr}$$

- servers

$$\mathrm{Sr} \stackrel{def}{=} (\mathrm{request}, r_{rp}).\mathrm{Sl} + (\mathit{break}, r_{bk}).\mathit{Sb}$$
$$\mathrm{Sl} \stackrel{def}{=} (\mathrm{log}, r_{lg}).\mathrm{Sr} + (\mathit{remove}, r_{rm}).\mathrm{Sm}$$
$$\mathrm{Sm} \stackrel{def}{=} (\mathit{maint}, r_{mn}).\mathrm{Sr} + (\mathrm{replace}, r_{rc}).\mathrm{Sr}$$
$$\mathit{Sb} \stackrel{def}{=} (\mathit{fix}, r_{fx}).\mathit{St}$$
$$\mathit{St} \stackrel{def}{=} (\mathrm{test}, r_{ts}).\mathit{St} + (\mathit{compl}, r_{cm}).\mathrm{Sr}$$

Mapping to TDSHA

- continuous sequential components: $\mathrm{Cr}, \mathrm{Ct}, \mathrm{Sr}, \mathrm{Sl}, \mathrm{Sm}$
- integral sequential components: *Sb*, *St*

Mapping to TDSHA

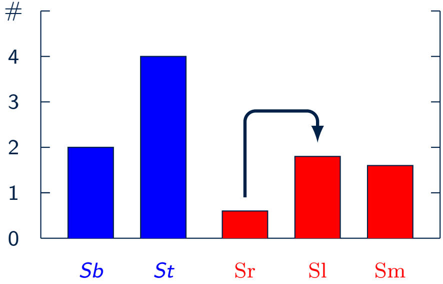
- continuous sequential components: $\mathrm{Cr}, \mathrm{Ct}, \mathrm{Sr}, \mathrm{Sl}, \mathrm{Sm}$
- integral sequential components: Sb, St
- population vector: $(\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{Sm}, \#\mathit{Sb}, \#\mathit{St})$

Mapping to TDSHA

- continuous sequential components: $\mathrm{Cr}, \mathrm{Ct}, \mathrm{Sr}, \mathrm{Sl}, \mathrm{Sm}$
- integral sequential components: Sb, St
- population vector: $(\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{Sm}, \#Sb, \#St)$
- PEPA is conservative: both $N_C = \#\mathrm{Cr} + \#\mathrm{Ct}$ and $N_S = \#\mathrm{Sr} + \#\mathrm{Sl} + \#\mathrm{Sm} + \#Sb + \#St$ are invariant
- TDSHA
  - modes: $(\#Sb, \#St) \in \{0, \ldots, N_S\} \times \{0, \ldots, N_S\}$
  - variables: $(X_{\mathrm{Cr}}, X_{\mathrm{Ct}}, X_{\mathrm{Sr}}, X_{\mathrm{Sl}}, X_{\mathrm{Sm}})$
  - initial state: $((\#Sb, \#St), (\#\mathrm{Cr}, \#\mathrm{Ct}, \#\mathrm{Sr}, \#\mathrm{Sl}, \#\mathrm{St}))$
  - continuous and stochastic transitions

Continuous transitions between continuous components

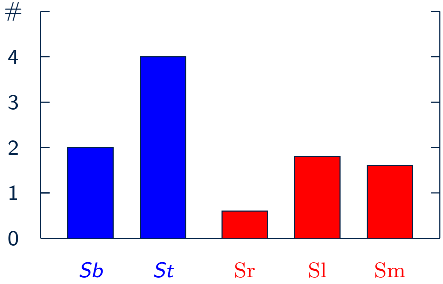
- $\mathrm{Sr} \xrightarrow{(\mathrm{request}, r_{rp} \cdot \#\mathrm{Sr})}_{\star} \mathrm{Sl}$
- continuous transition: flow is determined by ODEs



- $((\#Sb, \#St), (0, 0, -1, 1, 0), r_{rp} \cdot \#\mathrm{Sr}, \mathrm{request})$

Continuous transition at a discrete component

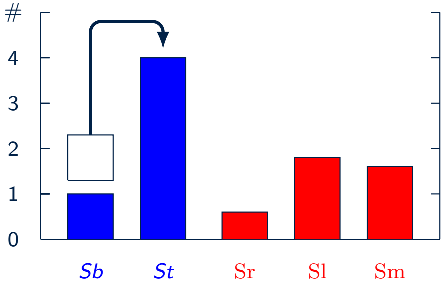
- $St \xrightarrow{(\text{test}, r_{ts} \cdot \# St)}_\star St$
- continuous transition: no flow because single component



- $((\# Sb, \# St), (0, 0, 0, 0, 0), r_{ts} \cdot \# St, \text{request})$

Discrete transitions between discrete components

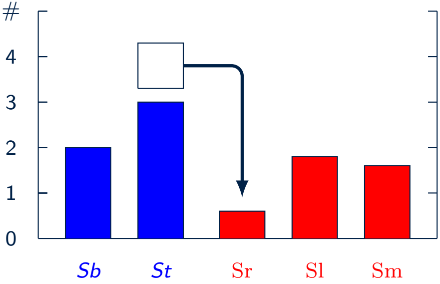
- $Sb \xrightarrow{(fix, r_{fx} \cdot \#Sb)}_{\star} St$
- stochastic transition: unit quantity is shifted



- $((\#Sb, \#St), (\#Sb - 1, \#St + 1), true, true, r_{fx} \cdot \#Sb, fix)$

Discrete transition from discrete to continuous component

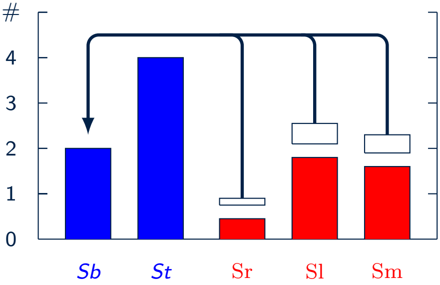
- $St \xrightarrow{(compl, r_{cm} \cdot \#St)}_{\star} \mathrm{Sr}$
- stochastic transition: unit quantity is shifted



- $((\#Sb, \#St), (\#Sb, \#St - 1), true, R, r_{cm} \cdot \#St, compl)$ with $R = (X'_{\mathrm{Sr}} = X_{\mathrm{Sr}} + 1)$

Discrete transition from continuous to discrete component

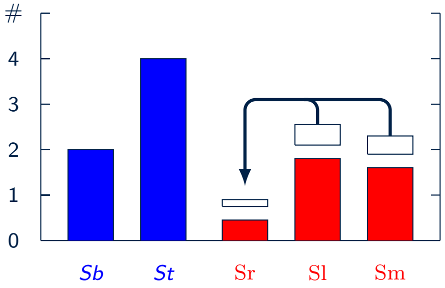
- $\mathrm{Sr} \xrightarrow{(break, r_{bk} \cdot \#\mathrm{Sr})}_{\star} Sb$
- stochastic transition: unit quantity is shifted proportionally



- $((\#Sb, \#St), (\#Sb + 1, \#St), true, R, r_{bk} \cdot \#\mathrm{Sr}, break)$ with $R = (X'_{\mathrm{Sr}} = X_{\mathrm{Sr}} - z_r) \wedge (X'_{\mathrm{Sl}} = X_{\mathrm{Sl}} - z_l) \wedge (X'_{\mathrm{Sm}} = X_{\mathrm{Sm}} - z_m)$ and $z_r + z_l + z_m = 1$

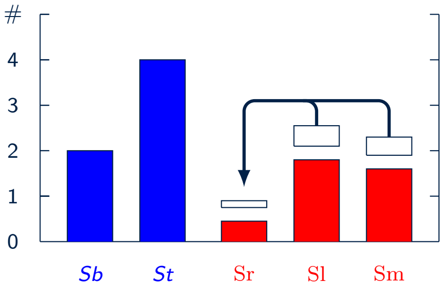
Discrete transition between continuous components

- $\text{Sm} \xrightarrow{(\mathit{maint}, r_{mn} \cdot \#\text{Sm})}_{\star} \text{Sr}$
- stochastic transition: unit quantity is shifted proportionally

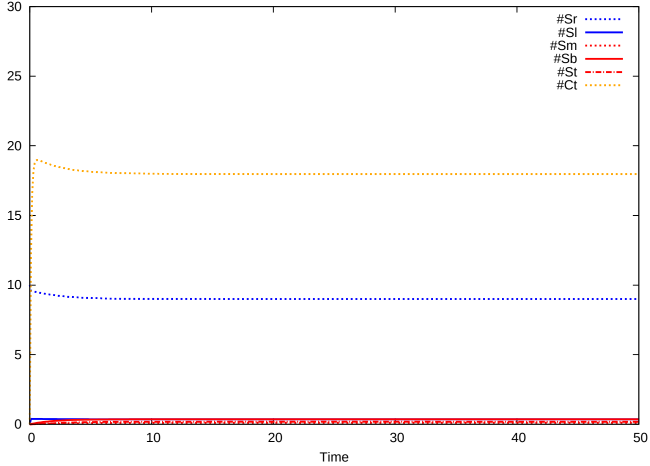


Discrete transition between continuous components

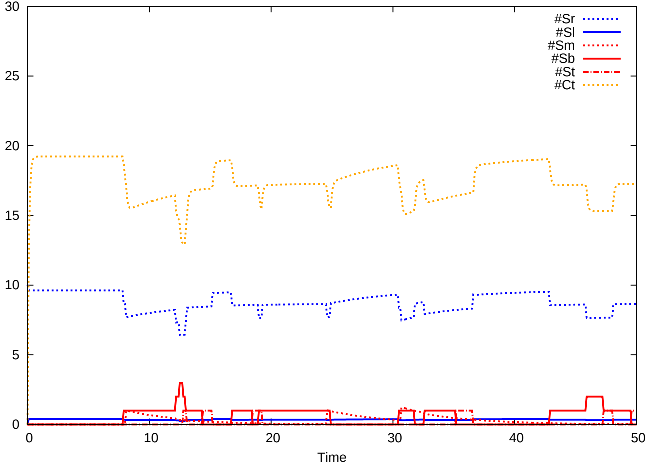
- $((\#Sb, \#St), (\#Sb, \#St), true, R, r_{mn} \cdot \#\mathrm{Sm}, maint)$ where $R = (X'_{\mathrm{Sr}} = X_{\mathrm{Sr}} - z_r + 1) \wedge (X'_{\mathrm{Sl}} = X_{\mathrm{Sl}} - z_l) \wedge (X'_{\mathrm{Sm}} = X_{\mathrm{Sm}} - z_m)$ and $z_r + z_l + z_m = 1$



Continuous determinstic simulation



Hybrid simulation



Conclusions

- The hybrid semantics for PEPA is a bridge between the fully discrete approach and the deterministic approach of fluid approximation.

Conclusions

- The hybrid semantics for PEPA is a bridge between the fully discrete approach and the deterministic approach of fluid approximation.
- The numerical results suggest that hybrid simulation may yield accurate results faster than full stochastic simulation