Stochastic Analysis in PEPA

Stephen Gilmore
University of Edinburgh

(Joint work with Jane Hillston)

MLQA, York, March 28, 2009
Outline

1. Performance modelling with process algebras
   - Performance Evaluation Process Algebra

2. Comparing performance measures
   - Computed with continuous time
   - Computed with continuous space
   - Comparison of computed measures

3. Case study in Web Services
   - Description
   - Analysis

4. Commentary and comparison
Outline

1. Performance modelling with process algebras
   - Performance Evaluation Process Algebra

2. Comparing performance measures
   - Computed with continuous time
   - Computed with continuous space
   - Comparison of computed measures

3. Case study in Web Services
   - Description
   - Analysis

4. Commentary and comparison
Performance Evaluation Process Algebra

PEPA components perform activities either independently or in co-operation with other components.
Performance Evaluation Process Algebra

PEPA components perform activities either independently or in co-operation with other components.

The rate at which an activity is performed is quantified by some component in each co-operation. The symbol $\top$ indicates that the rate value is quantified elsewhere (not in this component).
PEPA components perform activities either independently or in co-operation with other components.

The rate at which an activity is performed is quantified by some component in each co-operation. The symbol $\top$ indicates that the rate value is quantified elsewhere (not in this component).

\[
(\alpha, r).P \quad \text{Prefix} \\
P_1 + P_2 \quad \text{Choice} \\
P_1 \bowtie_l P_2 \quad \text{Co-operation} \\
P/L \quad \text{Hiding} \\
X \quad \text{Variable}
\]
PEPA: informal semantics (sequential sublanguage)

\[(\alpha, r).S\]

The activity \((\alpha, r)\) takes time \(\Delta t\) (drawn from the exponential distribution with parameter \(r\)).

\[S_1 + S_2\]

In this choice either \(S_1\) or \(S_2\) will complete an activity first. The other is discarded.
PEPA: informal semantics (combinators)

\[ C_1 \uplus_L C_2 \]

All activities of \( C_1 \) and \( C_2 \) with types in \( L \) are shared: others remain individual.

**NOTATION:** write \( C_1 \parallel C_2 \) if \( L \) is empty.

\[ C / L \]

Activities of \( C \) with types in \( L \) are hidden (\( \tau \) type activities) to be thought of as internal delays.
In a PEPA model if we define the stochastic process $X(t)$, such that $X(t) = C_i$ indicates that the system behaves as component $C_i$ at time $t$, then $X(t)$ is a Markov process which can be characterised by a matrix, $Q$. 
Equilibrium probability distribution

A stationary or equilibrium probability distribution, \( \pi(\cdot) \), exists for every time-homogeneous irreducible Markov process whose states are all positive-recurrent.

This distribution is found by solving the global balance equation

\[
\pi Q = 0
\]

subject to the normalisation condition

\[
\sum \pi(C_i) = 1.
\]
CTMCs are memoryless stochastic processes

A continuous-time Markov chain is a memoryless stochastic process.

\[ \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_1) = x_1) = \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n) \]
Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time $t$ has elapsed since the last event, and no events have occurred?
Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after \( t + s \), given that time \( t \) has elapsed since the last event, and no events have occurred?

\[
\Pr(T > t + s | T > t) = \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)}
\]

This value is independent of \( t \) (and so the time already spent has not been remembered).
Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after \( t + s \), given that time \( t \) has elapsed since the last event, and no events have occurred?

\[
\Pr(T > t + s \mid T > t) = \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)} = \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}}
\]
Suppose that the last event was at time 0. What is the probability that the next event will be after \( t + s \), given that time \( t \) has elapsed since the last event, and no events have occurred?

\[
\Pr(T > t + s \mid T > t) = \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)}
\]

\[
= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}}
\]

\[
= e^{-\lambda s}
\]
Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after \( t + s \), given that time \( t \) has elapsed since the last event, and no events have occurred?

\[
\Pr(T > t + s \mid T > t) = \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)}
\]

\[
= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}}
\]

\[
= e^{-\lambda s}
\]

This value is independent of \( t \) (and so the time already spent has not been remembered).
The importance of being exponential

\[
\begin{align*}
(\alpha, r).\text{Stop} & \parallel (\beta, s).\text{Stop} \\
(\alpha, r) & \rightarrow \text{Stop} \parallel (\beta, s).\text{Stop} \\
\text{Stop} & \parallel (\beta, s).\text{Stop} \\
(\beta, s) & \rightarrow \text{Stop} \parallel \text{Stop} \\
(\alpha, r) & \rightarrow \text{Stop} \parallel \text{Stop} \\
(\beta, s) & \rightarrow \text{Stop} \parallel \text{Stop} \\
(\alpha, r) & \rightarrow \text{Stop} \parallel \text{Stop}
\end{align*}
\]
The importance of being exponential

\[
(\alpha, r).\text{Stop} \parallel (\beta, s).\text{Stop}
\]

\[
\text{Stop} \parallel (\beta, s).\text{Stop}
\]

\[
(\beta, s)
\]

\[
\text{Stop} \parallel \text{Stop}
\]

\[
(\alpha, r)
\]

\[
(\alpha, r).\text{Stop} \parallel \text{Stop}
\]

\[
(\beta, s)
\]

\[
\text{Stop} \parallel \text{Stop}
\]

\[
(\alpha, r)
\]
The importance of being exponential

\[(\alpha, r).\text{Stop} \parallel (\beta, s).\text{Stop}\]

\[\Downarrow (\alpha, r)\]

\[\Downarrow \text{Stop} \parallel (\beta, s).\text{Stop}\]

\[\Downarrow (\beta, s)\]

\[\Downarrow \text{Stop} \parallel \text{Stop}\]

\[\Downarrow (\alpha, r)\]

\[\Downarrow (\beta, s).\text{Stop} \parallel \text{Stop}\]
The importance of being exponential

\[(\alpha, r).Stop \parallel (\beta, s).Stop\]

\[(\alpha, r) \rightarrow (\alpha, r) \rightarrow (\beta, s) \rightarrow (\beta, s) \rightarrow (\alpha, r) \rightarrow (\alpha, r) \rightarrow (\beta, s) \rightarrow (\beta, s)\]

Stop \parallel (\beta, s).Stop

Stop \parallel Stop
The importance of being exponential

\[(\alpha, r).Stop \parallel (\beta, s).Stop\]

\[\text{Stop} \parallel (\beta, s).Stop\]
\[\text{Stop} \parallel (\alpha, r).Stop \parallel \text{Stop}\]
\[\text{Stop} \parallel (\alpha, r)\]
\[\text{Stop} \parallel \text{Stop}\]

\[\text{Stop} \parallel (\beta, s)\]
The importance of being exponential

\[
\begin{align*}
(\alpha, r).\text{Stop} \parallel (\beta, s).\text{Stop} \\
(\alpha, r) & \quad \rightarrow \quad (\beta, s) \\
\text{Stop} \parallel (\beta, s).\text{Stop} & \quad \rightarrow \quad (\alpha, r).\text{Stop} \parallel \text{Stop} \\
(\beta, s) & \quad \rightarrow \quad (\alpha, r) \\
\text{Stop} \parallel \text{Stop} & \quad \rightarrow \quad (\beta, s) \quad \parallel \quad (\alpha, r)
\end{align*}
\]
The importance of being exponential

\[(\alpha, r).Stop \parallel (\beta, s).Stop\]

\[(\alpha, r) \quad (\beta, s) \quad (\alpha, r).Stop \parallel Stop \quad (\alpha, r)\]

\[Stop \parallel (\beta, s).Stop\]
The importance of being exponential

\[(\alpha, r).\text{Stop} \parallel (\beta, s).\text{Stop}\]

\[(\alpha, r) \rightarrow (\beta, s) \rightarrow \text{Stop} \parallel (\beta, s).\text{Stop} \rightarrow (\alpha, r).\text{Stop} \parallel \text{Stop} \rightarrow \text{Stop} \parallel \text{Stop} \rightarrow (\alpha, r)\]

The memoryless property of the negative exponential distribution means that residual times do not need to be recorded.
The importance of being exponential

We retain the expansion law of classical process algebra:

\[(\alpha, r).\text{Stop} \parallel (\beta, s).\text{Stop} = (\alpha, r).((\beta, s).\text{Stop} \parallel \text{Stop}) + (\beta, s).((\alpha, r).\text{Stop} \parallel \text{Stop})\]

only if the negative exponential distribution is assumed.
Outline

1. Performance modelling with process algebras
   - Performance Evaluation Process Algebra

2. Comparing performance measures
   - Computed with continuous time
   - Computed with continuous space
   - Comparison of computed measures

3. Case study in Web Services
   - Description
   - Analysis

4. Commentary and comparison
### Queue example

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_0 \triangleq (\text{arrive}, \lambda).Q_1 )</td>
<td>A queue with arrivals at rate ( \lambda ) and service at rate ( \mu ) and capacity 8.</td>
</tr>
<tr>
<td>( Q_i \triangleq (\text{arrive}, \lambda).Q_{i+1} + (\text{serve}, \mu).Q_{i-1} )</td>
<td></td>
</tr>
<tr>
<td>( Q_8 \triangleq (\text{serve}, \mu).Q_7 )</td>
<td></td>
</tr>
</tbody>
</table>

\( 0 < i < 8 \)

A queue with arrivals at rate \( \lambda \), service at rate \( \mu \) and capacity 8 (thus \( 0 \leq \text{len} < 9 \)).
Computing performance measures: CTMCs

Queue example

\[ Q_0 \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 \]
\[ Q_i \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_{i+1} + (\text{serve}, \mu).Q_{i-1} \quad (0 < i < 8) \]
\[ Q_8 \overset{\text{def}}{=} (\text{serve}, \mu).Q_7 \]

A queue with arrivals at rate \( \lambda \), service at rate \( \mu \) and capacity 8 (thus \( 0 \leq \text{len} < 9 \)). For \( \lambda = 1, \mu = 4 \) steady-state is:

<table>
<thead>
<tr>
<th>\text{Index}</th>
<th>\text{Probability}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.7500</td>
</tr>
<tr>
<td>1</td>
<td>0.1875</td>
</tr>
<tr>
<td>2</td>
<td>0.0468</td>
</tr>
<tr>
<td>3</td>
<td>0.0117</td>
</tr>
<tr>
<td>4</td>
<td>0.0029</td>
</tr>
<tr>
<td>5</td>
<td>0.0007</td>
</tr>
<tr>
<td>6</td>
<td>0.0000</td>
</tr>
<tr>
<td>7</td>
<td>0.0000</td>
</tr>
<tr>
<td>8</td>
<td>0.0000</td>
</tr>
</tbody>
</table>
Computing performance measures: CTMCs

Queue example

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_0$</td>
<td>$(\text{arrive}, \lambda)$</td>
<td>$Q_1$</td>
</tr>
<tr>
<td>$Q_i$</td>
<td>$(\text{arrive}, \lambda)$</td>
<td>$Q_{i+1}$ + $(\text{serve}, \mu)$. $Q_{i-1}$</td>
</tr>
<tr>
<td>$Q_8$</td>
<td>$(\text{serve}, \mu)$</td>
<td>$Q_7$</td>
</tr>
</tbody>
</table>

A queue with arrivals at rate $\lambda$, service at rate $\mu$ and capacity 8 (thus $0 \leq \text{len} < 9$). For $\lambda = 1, \mu = 2$ steady-state is:

<table>
<thead>
<tr>
<th>State</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5009</td>
</tr>
<tr>
<td>1</td>
<td>0.2504</td>
</tr>
<tr>
<td>2</td>
<td>0.1252</td>
</tr>
<tr>
<td>3</td>
<td>0.0626</td>
</tr>
<tr>
<td>4</td>
<td>0.0313</td>
</tr>
<tr>
<td>5</td>
<td>0.0156</td>
</tr>
<tr>
<td>6</td>
<td>0.0078</td>
</tr>
<tr>
<td>7</td>
<td>0.0039</td>
</tr>
<tr>
<td>8</td>
<td>0.0019</td>
</tr>
</tbody>
</table>
A queue with arrivals at rate \( \lambda \), service at rate \( \mu \) and capacity 8 (thus \( 0 \leq \text{len} < 9 \)). For \( \lambda = 1, \mu = 1 \) steady-state is:

<table>
<thead>
<tr>
<th></th>
<th>0.1111</th>
<th></th>
<th>0.1111</th>
<th></th>
<th>0.1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.1111</td>
<td>3</td>
<td>0.1111</td>
<td>6</td>
<td>0.1111</td>
</tr>
<tr>
<td>1</td>
<td>0.1111</td>
<td>4</td>
<td>0.1111</td>
<td>7</td>
<td>0.1111</td>
</tr>
<tr>
<td>2</td>
<td>0.1111</td>
<td>5</td>
<td>0.1111</td>
<td>8</td>
<td>0.1111</td>
</tr>
</tbody>
</table>
Computing performance measures: CTMCs

Queue example

\[
\begin{align*}
Q_0 & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 \\
Q_i & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_{i+1} + (\text{serve}, \mu).Q_{i-1} \\
Q_8 & \overset{\text{def}}{=} (\text{serve}, \mu).Q_7 \\
\end{align*}
\]

(0 < i < 8)

A queue with arrivals at rate \(\lambda\), service at rate \(\mu\) and capacity 8 (thus \(0 \leq \text{len} < 9\)). For \(\lambda = 2, \mu = 1\) steady-state is:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0019</td>
<td>3</td>
<td>0.0156</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>0.0039</td>
<td>4</td>
<td>0.0313</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>0.0078</td>
<td>5</td>
<td>0.0626</td>
<td>8</td>
</tr>
</tbody>
</table>
Computing performance measures: CTMCs

Queue example

\[
Q_0 \overset{\text{def}}{=} (arrive, \lambda).Q_1 \quad Q_i \overset{\text{def}}{=} (arrive, \lambda).Q_{i+1} + (serve, \mu).Q_{i-1} \\
Q_8 \overset{\text{def}}{=} (serve, \mu).Q_7
\]

A queue with arrivals at rate \( \lambda \), service at rate \( \mu \) and capacity 8 (thus \( 0 \leq \text{len} < 9 \)). For \( \lambda = 4, \mu = 1 \) steady-state is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( P(i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0000</td>
</tr>
<tr>
<td>1</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>0.0007</td>
</tr>
<tr>
<td>4</td>
<td>0.0029</td>
</tr>
<tr>
<td>5</td>
<td>0.0117</td>
</tr>
<tr>
<td>6</td>
<td>0.0468</td>
</tr>
<tr>
<td>7</td>
<td>0.1875</td>
</tr>
<tr>
<td>8</td>
<td>0.7500</td>
</tr>
</tbody>
</table>
Calculating average queue length: CTMCs

To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

\[ a = \sum_{i=0}^{8} i \pi(i) \]
To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

\[ a = \sum_{i=0}^{8} i \pi(i) \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at equilibrium)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
</tbody>
</table>
Calculating average queue length: CTMCs

To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

\[ a = \sum_{i=0}^{8} i \pi(i) \]

<table>
<thead>
<tr>
<th>Arrival rate ($\lambda$)</th>
<th>Service rate ($\mu$)</th>
<th>Av. queue length (at equilibrium)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
</tbody>
</table>
Calculating average queue length: CTMCs

To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

\[ a = \sum_{i=0}^{8} i \pi(i) \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at equilibrium)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.0000</td>
</tr>
</tbody>
</table>
Calculating average queue length: CTMCs

To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

\[ a = \sum_{i=0}^{8} i \pi(i) \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at equilibrium)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.0000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.0176</td>
</tr>
</tbody>
</table>
Calculating average queue length: CTMCs

To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

\[ a = \sum_{i=0}^{8} i \pi(i) \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at equilibrium)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.0000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.0176</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.6667</td>
</tr>
</tbody>
</table>
Calculating average queue length: CTMCs

To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

\[ a = \sum_{i=0}^{8} i \pi(i) \]

<table>
<thead>
<tr>
<th>Arrival rate (( \lambda ))</th>
<th>Service rate (( \mu ))</th>
<th>Av. queue length (at equilibrium)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.0000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.0176</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.6667</td>
</tr>
</tbody>
</table>
Stochastic Analysis in PEPA

Comparing performance measures
Computed with continuous space

Queues and differential equations

CTMC: 

ODEs:
Queue and differential equations

CTMC:  
ODEs:  

Comparing performance measures
Computed with continuous space
Queues and differential equations
Queues and differential equations

CTMC: 

ODEs:
Computing performance measures: ODEs

\[ \lambda = 1 \]
\[ \mu = 4 \]
Computing performance measures: ODEs

\[ \lambda = 1 \]
\[ \mu = 2 \]
Computing performance measures: ODEs

\[ \lambda = 1 \]
\[ \mu = 1 \]
Computing performance measures: ODEs

\[ \lambda = 2 \]
\[ \mu = 1 \]
Computing performance measures: ODEs

\[ \lambda = 4 \]
\[ \mu = 1 \]
Calculating average queue length: ODEs

To calculate the average queue length, weight the fraction of queues of a given length by the number of customers in the queue.

\[ a = \sum_{i=0}^{8} i \cdot \frac{[Q_i]}{90} \]
To calculate the average queue length, weight the fraction of queues of a given length by the number of customers in the queue.

\[ a = \sum_{i=0}^{8} i \frac{[Q_i]}{90} \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at (t = 50))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
</tbody>
</table>
Calculating average queue length: ODEs

To calculate the average queue length, weight the fraction of queues of a given length by the number of customers in the queue.

\[ a = \sum_{i=0}^{8} i \frac{[Q_i]}{90} \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length ((at \ t = 50))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
</tbody>
</table>
Calculating average queue length: ODEs

To calculate the average queue length, weight the fraction of queues of a given length by the number of customers in the queue.

\[
a = \sum_{i=0}^{8} i \frac{[Q_i]}{90}
\]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at (t = 50))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3.9914</td>
</tr>
</tbody>
</table>
Calculating average queue length: ODEs

To calculate the average queue length, weight the fraction of queues of a given length by the number of customers in the queue.

\[ a = \sum_{i=0}^{8} i \frac{[Q_i]}{90} \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at (t = 50))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3.9914</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.0176</td>
</tr>
</tbody>
</table>
Calculating average queue length: ODEs

To calculate the average queue length, weight the fraction of queues of a given length by the number of customers in the queue.

\[ a = \sum_{i=0}^{8} i \frac{[Q_i]}{90} \]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at (t = 50))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3.9914</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.0176</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.6667</td>
</tr>
</tbody>
</table>
Calculating average queue length: ODEs

To calculate the average queue length, weight the fraction of queues of a given length by the number of customers in the queue.

\[
a = \sum_{i=0}^{8} i \frac{[Q_i]}{90}
\]

<table>
<thead>
<tr>
<th>Arrival rate ((\lambda))</th>
<th>Service rate ((\mu))</th>
<th>Av. queue length (at (t = 50))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.3333</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.9824</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3.9914</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.0176</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.6667</td>
</tr>
</tbody>
</table>
### Comparison of computed measures

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 50$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298624889</td>
<td>$3.8 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
### Comparison of computed measures

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 50$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298624889</td>
<td>$3.8 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.982387959648</td>
<td>0.982387242222</td>
<td>$7.1 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
## Comparison of computed measures

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 50$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298624889</td>
<td>$3.8 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.982387959648</td>
<td>0.982387242222</td>
<td>$7.1 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.000000000000000000000</td>
<td>3.991409877780</td>
<td>$8.6 \times 10^{-3}$</td>
</tr>
</tbody>
</table>
### Comparison of computed measures

<table>
<thead>
<tr>
<th>λ</th>
<th>μ</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 50$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298624889</td>
<td>$3.8 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.982387959648</td>
<td>0.982387242222</td>
<td>$7.1 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.000000000000000</td>
<td>3.991409877780</td>
<td>$8.6 \times 10^{-3}$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.017612040350</td>
<td>7.017612412220</td>
<td>$-3.7 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
## Comparison of computed measures

<table>
<thead>
<tr>
<th>λ</th>
<th>μ</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 50$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298624889</td>
<td>3.8 × 10^{-7}</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.982387959648</td>
<td>0.982387242222</td>
<td>7.1 × 10^{-7}</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.000000000000000000</td>
<td>3.991409877780</td>
<td>8.6 × 10^{-3}</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.017612040350</td>
<td>7.017612412220</td>
<td>−3.7 × 10^{-7}</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.666700990970</td>
<td>7.666701341490</td>
<td>−3.5 × 10^{-7}</td>
</tr>
</tbody>
</table>
## Comparison of computed measures

<table>
<thead>
<tr>
<th>λ</th>
<th>μ</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 50$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298624889</td>
<td>$3.8 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.982387959648</td>
<td>0.982387242222</td>
<td>$7.1 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.000000000000000</td>
<td>3.991409877780</td>
<td>$8.6 \times 10^{-3}$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.017612040350</td>
<td>7.017612412220</td>
<td>$–3.7 \times 10^{-7}$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.666700990970</td>
<td>7.666701341490</td>
<td>$–3.5 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
### Comparison of computed measures

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\mu$</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 100$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298736822</td>
<td>$2.7 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.982387959648</td>
<td>0.982387201111</td>
<td>$7.6 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.000000000000000</td>
<td>3.999979511110</td>
<td>$2.0 \times 10^{-5}$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.017612040350</td>
<td>7.017613132220</td>
<td>$-1.1 \times 10^{-6}$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.666700990970</td>
<td>7.666701089580</td>
<td>$-9.8 \times 10^{-8}$</td>
</tr>
</tbody>
</table>
Comparison of computed measures

<table>
<thead>
<tr>
<th>λ</th>
<th>μ</th>
<th>Av. queue length (CTMCs at equilibrium)</th>
<th>Av. queue length (ODEs at $t = 200$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.333299009029</td>
<td>0.333298753978</td>
<td>$2.5 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.982387959648</td>
<td>0.982386995556</td>
<td>$9.6 \times 10^{-7}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4.000000000000000</td>
<td>4.000000266670</td>
<td>$-2.6 \times 10^{-7}$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.017612040350</td>
<td>7.017613704440</td>
<td>$-1.6 \times 10^{-6}$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.666700990970</td>
<td>7.666701306580</td>
<td>$-3.2 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
Small queue example: CTMCs

Small queue example

\[
\begin{align*}
Q_0 &\overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 & Q_1 &\overset{\text{def}}{=} (\text{arrive}, \lambda).Q_2 + (\text{serve}, \mu).Q_0 \\
Q_2 &\overset{\text{def}}{=} (\text{serve}, \mu).Q_1 & &
\end{align*}
\]
Small queue example: CTMCs

\[
Q_0 \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 \quad Q_1 \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_2 + (\text{serve}, \mu).Q_0 \\
Q_2 \overset{\text{def}}{=} (\text{serve}, \mu).Q_1
\]

\[
Q = \begin{bmatrix}
-\lambda & \lambda & 0 \\
\mu & -\lambda - \mu & \lambda \\
0 & \mu & -\mu
\end{bmatrix}
\]
Small queue example: CTMCs

\[
Q_0 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_1 \quad Q_1 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_2 + (\text{serve, } \mu).Q_0
\]

\[
Q_2 \overset{\text{def}}{=} (\text{serve, } \mu).Q_1
\]

\[
Q = \begin{bmatrix}
-\lambda & \lambda & 0 \\
\mu & -\lambda - \mu & \lambda \\
0 & \mu & -\mu
\end{bmatrix}
\]

\[
\pi Q = 0
\]
Small queue example: CTMCs

\[
Q_0 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_1 \quad Q_1 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_2 + (\text{serve, } \mu).Q_0 \\
Q_2 \overset{\text{def}}{=} (\text{serve, } \mu).Q_1
\]

\[
Q = \begin{bmatrix}
-\lambda & \lambda & 0 \\
\mu & -\lambda - \mu & \lambda \\
0 & \mu & -\mu
\end{bmatrix}
\]

\[
\pi Q = 0 \quad \sum \pi = 1
\]
Small queue example: CTMCs

$\mathbf{Q}_0 \overset{\text{def}}{=} (\text{arrive, } \lambda). \mathbf{Q}_1 \quad \mathbf{Q}_1 \overset{\text{def}}{=} (\text{arrive, } \lambda). \mathbf{Q}_2 + (\text{serve, } \mu). \mathbf{Q}_0 \\
\mathbf{Q}_2 \overset{\text{def}}{=} (\text{serve, } \mu). \mathbf{Q}_1$

\[
\mathbf{Q} = \begin{bmatrix}
-\lambda & \lambda & 0 \\
\mu & -\lambda - \mu & \lambda \\
0 & \mu & -\mu
\end{bmatrix}
\]

\[
\pi \mathbf{Q} = 0 \quad \sum \pi = 1
\]

\[
\pi = \begin{bmatrix}
\frac{\mu^2}{\lambda^2 + \mu \lambda + \mu^2}, & \frac{\mu \lambda}{\lambda^2 + \mu \lambda + \mu^2}, & \frac{\lambda^2}{\lambda^2 + \mu \lambda + \mu^2}
\end{bmatrix}
\]
Small queue example: ODEs

Small queue example

\[
\begin{align*}
Q_0 & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 \\
Q_1 & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_2 + (\text{serve}, \mu).Q_0 \\
Q_2 & \overset{\text{def}}{=} (\text{serve}, \mu).Q_1
\end{align*}
\]
Small queue example: ODEs

\[
\begin{align*}
Q_0 & \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_1 \\
Q_1 & \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_2 + (\text{serve, } \mu).Q_0 \\
Q_2 & \overset{\text{def}}{=} (\text{serve, } \mu).Q_1
\end{align*}
\]

\[
\frac{dQ_0}{dt} = -\lambda Q_0 + \mu Q_1
\]
Small queue example: ODEs

\[
\begin{align*}
Q_0 & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 \\
Q_1 & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_2 + (\text{serve}, \mu).Q_0 \\
Q_2 & \overset{\text{def}}{=} (\text{serve}, \mu).Q_1
\end{align*}
\]

\[
\begin{align*}
\frac{dQ_0}{dt} &= -\lambda Q_0 + \mu Q_1 \\
\frac{dQ_1}{dt} &= \lambda Q_0 - \lambda Q_1 - \mu Q_1 + \mu Q_2
\end{align*}
\]
Small queue example: ODEs

\[
\begin{align*}
Q_0 & \overset{\text{def}}{=} \text{(arrive, } \lambda) \cdot Q_1 \\
Q_1 & \overset{\text{def}}{=} \text{(arrive, } \lambda) \cdot Q_2 + \text{(serve, } \mu) \cdot Q_0 \\
Q_2 & \overset{\text{def}}{=} \text{(serve, } \mu) \cdot Q_1
\end{align*}
\]

\[
\begin{align*}
\frac{dQ_0}{dt} &= -\lambda Q_0 + \mu Q_1 \\
\frac{dQ_1}{dt} &= \lambda Q_0 - \lambda Q_1 - \mu Q_1 + \mu Q_2 \\
\frac{dQ_2}{dt} &= \lambda Q_1 - \mu Q_2
\end{align*}
\]
Small queue example: ODEs (stationary points)

<table>
<thead>
<tr>
<th>Small queue example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_0 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_1$</td>
</tr>
<tr>
<td>$Q_1 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_2 + (\text{serve, } \mu).Q_0$</td>
</tr>
<tr>
<td>$Q_2 \overset{\text{def}}{=} (\text{serve, } \mu).Q_1$</td>
</tr>
</tbody>
</table>

\[
0 = -\lambda Q_0 + \mu Q_1 \\
0 = \lambda Q_0 - \lambda Q_1 - \mu Q_1 + \mu Q_2 \\
0 = \lambda Q_1 - \mu Q_2
\]
Small queue example: ODEs (stationary points)

\[
\begin{align*}
Q_0 & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 \\
Q_1 & \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_2 + (\text{serve}, \mu).Q_0 \\
Q_2 & \overset{\text{def}}{=} (\text{serve}, \mu).Q_1
\end{align*}
\]

\[
0 = \begin{bmatrix} Q_0 & Q_1 & Q_2 \end{bmatrix}
\begin{bmatrix}
-\lambda & \lambda & 0 \\
\mu & -\lambda - \mu & \lambda \\
0 & \mu & -\mu
\end{bmatrix}
\]
Small queue example: ODEs (and CTMC solution)

Small queue example

\[ Q_0 \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_1 \]
\[ Q_1 \overset{\text{def}}{=} (\text{arrive}, \lambda).Q_2 + (\text{serve}, \mu).Q_0 \]
\[ Q_2 \overset{\text{def}}{=} (\text{serve}, \mu).Q_1 \]

\[
\mathbf{p} = \begin{bmatrix}
Q_0 & \frac{\lambda}{\mu} Q_0 & \frac{\lambda^2}{\mu^2} Q_0
\end{bmatrix}
\]
Small queue example: ODEs (and CTMC solution)

\[ Q_0 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_1 \quad Q_1 \overset{\text{def}}{=} (\text{arrive, } \lambda).Q_2 + (\text{serve, } \mu).Q_0 \]
\[ Q_2 \overset{\text{def}}{=} (\text{serve, } \mu).Q_1 \]

\[ p = \begin{bmatrix} Q_0 & \frac{\lambda}{\mu}Q_0 & \frac{\lambda^2}{\mu^2}Q_0 \end{bmatrix} \]

\[ \pi = \begin{bmatrix} \frac{\mu^2}{\lambda^2 + \mu \lambda + \mu^2}, & \frac{\mu \lambda}{\lambda^2 + \mu \lambda + \mu^2}, & \frac{\lambda^2}{\lambda^2 + \mu \lambda + \mu^2} \end{bmatrix} \]
What just happened?

We found that, for a sequential PEPA component, the differential equations are recording the same information as found in the infinitesimal generator matrix of the Markov chain.
What just happened?

We found that, for a sequential PEPA component, the differential equations are recording the same information as found in the infinitesimal generator matrix of the Markov chain.

The stationary points of the system of ODEs for an initial value of 1 make up the stationary probability distribution of the CTMC.
Isn’t this just the Chapman-Kolmogorov equations?

Now that we have discovered that we have a copy of a generator matrix in the ODEs aren’t we just back to

$$\frac{d\pi(t)}{dt} = \pi(t)Q$$
Isn’t this just the Chapman-Kolmogorov equations?

Now that we have discovered that we have a copy of a generator matrix in the ODEs aren’t we just back to

\[ \frac{d\pi(t)}{dt} = \pi(t)Q \]

Only if the system is a single sequential component. For even only two parallel queues, the generator matrix is much larger than the system of ODEs.
Generator matrix for two parallel queues

\[ Q = \begin{bmatrix}
-2\lambda & \lambda & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mu & -2\lambda - \mu & 0 & \lambda & \lambda & 0 & 0 & 0 & 0 & 0 \\
\mu & 0 & -2\lambda - \mu & 0 & \lambda & 0 & 0 & 0 & 0 & \lambda \\
0 & \mu & 0 & -\lambda - \mu & 0 & \lambda & 0 & 0 & 0 & 0 \\
0 & \mu & \mu & 0 & -2\lambda - 2\mu & \lambda & 0 & \lambda & 0 & 0 \\
0 & 0 & 0 & \mu & \mu & -\lambda - 2\mu & \lambda & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mu & -2\mu & \mu & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mu & 0 & \lambda & -\lambda - 2\mu & \mu \\
0 & 0 & \mu & 0 & 0 & 0 & 0 & 0 & \lambda & -\lambda - \mu \\
0 & 0 & 0 & 0 & 0 & 0 & \mu & 0 & \lambda & -\lambda - 2\mu & \mu
\end{bmatrix} \]
Steady-state for two parallel queues

$$\pi = \begin{bmatrix}
\frac{\mu^4}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu^3 \lambda}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu^3 \lambda}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu^2 \lambda^2}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu \lambda^3}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\lambda^4}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu \lambda^3}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu^2 \lambda^2}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu^2 \lambda^2}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}, \\
\frac{\mu^4}{2 \mu \lambda^3 + 3 \mu^2 \lambda^2 + 2 \mu^3 \lambda + \lambda^4 + \mu^4}
\end{bmatrix}$$
Outline

1. Performance modelling with process algebras
   - Performance Evaluation Process Algebra

2. Comparing performance measures
   - Computed with continuous time
   - Computed with continuous space
   - Comparison of computed measures

3. Case study in Web Services
   - Description
   - Analysis

4. Commentary and comparison
The example which we consider is a Web service which has two types of clients:

- first party application clients which access the web service across a secure intranet, and
- second party browser clients which access the Web service across the Internet.

Second party clients route their service requests via trusted brokers.
PEPA model: Second party clients

\[
\begin{align*}
SPC_{idle} & \overset{\text{def}}{=} (\text{compose}_{sp}, r_{sp\_cmp}).SPC_{enc} \\
SPC_{enc} & \overset{\text{def}}{=} (\text{encrypt}_b, r_{sp\_enck}).SPC_{sending} \\
SPC_{sending} & \overset{\text{def}}{=} (\text{request}_b, r_{sp\_req}).SPC_{waiting} \\
SPC_{waiting} & \overset{\text{def}}{=} (\text{response}_b, \top).SPC_{dec} \\
SPC_{dec} & \overset{\text{def}}{=} (\text{decrypt}_b, r_{sp\_decb}).SPC_{idle}
\end{align*}
\]
PEPA model: Brokers

\[
\begin{align*}
\text{Broker}_{idle} & \overset{\text{def}}{=} (\text{request}_b, \top).\text{Broker}_{dec\_input} \\
\text{Broker}_{dec\_input} & \overset{\text{def}}{=} (\text{decrypt}_{sp}, r_{b\_dec\_sp}).\text{Broker}_{enc\_input} \\
\text{Broker}_{enc\_input} & \overset{\text{def}}{=} (\text{encrypt}_{ws}, r_{b\_enc\_ws}).\text{Broker}_{sending} \\
\text{Broker}_{sending} & \overset{\text{def}}{=} (\text{request}_{ws}, r_{b\_req}).\text{Broker}_{waiting} \\
\text{Broker}_{waiting} & \overset{\text{def}}{=} (\text{response}_{ws}, \top).\text{Broker}_{dec\_resp} \\
\text{Broker}_{dec\_resp} & \overset{\text{def}}{=} (\text{decrypt}_{ws}, r_{b\_dec\_ws}).\text{Broker}_{enc\_resp} \\
\text{Broker}_{enc\_resp} & \overset{\text{def}}{=} (\text{encrypt}_{sp}, r_{b\_enc\_sp}).\text{Broker}_{replying} \\
\text{Broker}_{replying} & \overset{\text{def}}{=} (\text{response}_b, r_{b\_resp}).\text{Broker}_{idle}
\end{align*}
\]
PEPA model: First party clients

\[
\begin{align*}
FPC_{idle} & \overset{\text{def}}{=} (\text{compose}_{fp}, r_{fp\_cmp}).FPC_{calling} \\
FPC_{calling} & \overset{\text{def}}{=} (\text{invoke}_{ws}, r_{fp\_inv}).FPC_{blocked} \\
FPC_{blocked} & \overset{\text{def}}{=} (\text{result}_{ws}, \top).FPC_{idle}
\end{align*}
\]
PEPA model: Web service

Second party \rightarrow \text{Broker} \rightarrow \text{Web service} \rightarrow \text{First party}

\begin{align*}
WS_{idle} & \overset{\text{def}}{=} (request_{ws}, \top).WS_{decoding} + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} & \overset{\text{def}}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} & \overset{\text{def}}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} & \overset{\text{def}}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} & \overset{\text{def}}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} & \overset{\text{def}}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} & \overset{\text{def}}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{align*}
PEPA model: Web service

\[
\begin{align*}
WS_{idle} & \overset{\text{def}}{=} (\text{request}_{ws}, \top).WS_{decoding} \\
& + (\text{invoke}_{ws}, \top).WS_{method} \\
WS_{decoding} & \overset{\text{def}}{=} (\text{decryptReq}_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} & \overset{\text{def}}{=} (\text{execute}_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} & \overset{\text{def}}{=} (\text{encryptResp}_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} & \overset{\text{def}}{=} (\text{response}_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} & \overset{\text{def}}{=} (\text{execute}_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} & \overset{\text{def}}{=} (\text{result}_{ws}, r_{ws\_res}).WS_{idle}
\end{align*}
\]
PEPA model: Web service

\[WS_{idle} \overset{\text{def}}{=} (\text{request}_{ws}, \top) \cdot WS_{decoding}\]
\[+ (\text{invoke}_{ws}, \top) \cdot WS_{method}\]
\[WS_{decoding} \overset{\text{def}}{=} (\text{decryptReq}_{ws}, r_{ws\_dec\_b}) \cdot WS_{execution}\]
\[WS_{execution} \overset{\text{def}}{=} (\text{execute}_{ws}, r_{ws\_exec}) \cdot WS_{securing}\]
\[WS_{securing} \overset{\text{def}}{=} (\text{encryptResp}_{ws}, r_{ws\_enc\_b}) \cdot WS_{responding}\]
\[WS_{responding} \overset{\text{def}}{=} (\text{response}_{ws}, r_{ws\_resp\_b}) \cdot WS_{idle}\]
\[WS_{method} \overset{\text{def}}{=} (\text{execute}_{ws}, r_{ws\_exec}) \cdot WS_{returning}\]
\[WS_{returning} \overset{\text{def}}{=} (\text{result}_{ws}, r_{ws\_res}) \cdot WS_{idle}\]
PEPA model: Web service

\[
\begin{align*}
WS_{idle} & \stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
& + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} & \stackrel{def}{=} (decryptReq_{ws}, \text{r}_{ws}_{dec\_b}).WS_{execution} \\
WS_{execution} & \stackrel{def}{=} (execute_{ws}, \text{r}_{ws}_{exec}).WS_{securing} \\
WS_{securing} & \stackrel{def}{=} (encryptResp_{ws}, \text{r}_{ws}_{enc\_b}).WS_{responding} \\
WS_{responding} & \stackrel{def}{=} (response_{ws}, \text{r}_{ws}_{resp\_b}).WS_{idle} \\
WS_{method} & \stackrel{def}{=} (execute_{ws}, \text{r}_{ws}_{exec}).WS_{returning} \\
WS_{returning} & \stackrel{def}{=} (result_{ws}, \text{r}_{ws}_{res}).WS_{idle}
\end{align*}
\]
In the initial state of the system model we represent each of the four component types being initially in their idle state.

\[
\text{System} \overset{\text{def}}{=} (SPC_{idle} \otimes_{\mathcal{K}} \text{Broker}_{idle}) \otimes_{\mathcal{L}} (WS_{idle} \otimes_{\mathcal{M}} FPC_{idle})
\]

where \(\mathcal{K} = \{ request_b, response_b \}\), \(\mathcal{L} = \{ request_{ws}, response_{ws} \}\), \(\mathcal{M} = \{ invoke_{ws}, result_{ws} \}\).
PEPA model: System composition

In the initial state of the system model we represent each of the four component types being initially in their idle state.

\[
\text{System} \overset{\text{def}}{=} (SPC_{idle} \underset{\mathcal{K}}{\bowtie} Broker_{idle}) \underset{\mathcal{L}}{\bowtie} (WS_{idle} \underset{\mathcal{M}}{\bowtie} FPC_{idle})
\]

where

\[
\mathcal{K} = \{ \text{request}_b, \text{response}_b \}
\]
\[
\mathcal{L} = \{ \text{request}_{ws}, \text{response}_{ws} \}
\]
\[
\mathcal{M} = \{ \text{invoke}_{ws}, \text{result}_{ws} \}
\]

This model represents the smallest possible instance of the system, where there is one instance of each component type. We evaluate the system as the number of clients, brokers, and copies of the service increase.
Cost of analysis

- Performance models admit many different types of analysis. Some have lower evaluation cost, but are less informative, such as steady-state analysis. Others have higher evaluation cost, but are more informative, such as transient analysis.

- We compare ODE-based evaluation against other techniques which could be used to analyse the model.

- We compare against steady-state and transient analysis as implemented by the PRISM probabilistic model-checker, which provides PEPA as one of its input languages. We also compare against Monte Carlo Markov Chain simulation.
## Running times from analyses (in seconds)

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
</tbody>
</table>
**Running times from analyses (in seconds)**

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6,304</td>
<td>860</td>
<td>2.15</td>
<td>2.26</td>
<td>2.31</td>
<td>2.45</td>
<td>2.81</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>1000</td>
<td>1000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Running times from analyses (in seconds)

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6,304</td>
<td>860</td>
<td>2.15</td>
<td>2.26</td>
<td>2.31</td>
<td>2.45</td>
<td>2.81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1,130,496</td>
<td>161,296</td>
<td>172.48</td>
<td>255.48</td>
<td>588.80</td>
<td>2.48</td>
<td>2.83</td>
</tr>
</tbody>
</table>
## Running times from analyses (in seconds)

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6,304</td>
<td>860</td>
<td>2.15</td>
<td>2.26</td>
<td>2.31</td>
<td>2.45</td>
<td>2.81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1,130,496</td>
<td>161,296</td>
<td>172.48</td>
<td>255.48</td>
<td>588.80</td>
<td>2.48</td>
<td>2.83</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>$&gt;234M$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.44</td>
<td>2.85</td>
</tr>
</tbody>
</table>
### Running times from analyses (in seconds)

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6,304</td>
<td>860</td>
<td>2.15</td>
<td>2.26</td>
<td>2.31</td>
<td>2.45</td>
<td>2.81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1,130,496</td>
<td>161,296</td>
<td>172.48</td>
<td>255.48</td>
<td>588.80</td>
<td>2.48</td>
<td>2.83</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>$&gt;234M$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.78</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.78</td>
</tr>
</tbody>
</table>
## Running times from analyses (in seconds)

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6,304</td>
<td>860</td>
<td>2.15</td>
<td>2.26</td>
<td>2.31</td>
<td>2.45</td>
<td>2.81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1,130,496</td>
<td>161,296</td>
<td>172.48</td>
<td>255.48</td>
<td>588.80</td>
<td>2.48</td>
<td>2.83</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>$&gt;234M$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.44</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.78</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>500</td>
<td>1000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3.72</td>
</tr>
<tr>
<td>2000</td>
<td>2000</td>
<td>2000</td>
<td>2000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.77</td>
</tr>
</tbody>
</table>
Running times from analyses (in seconds)

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6,304</td>
<td>860</td>
<td>2.15</td>
<td>2.26</td>
<td>2.31</td>
<td>2.45</td>
<td>2.81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1,130,496</td>
<td>161,296</td>
<td>172.48</td>
<td>255.48</td>
<td>588.80</td>
<td>2.48</td>
<td>2.83</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>$&gt;234M$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.44</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.78</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>500</td>
<td>1000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3.72</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5.44</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>2.77</td>
</tr>
</tbody>
</table>
## Running times from analyses (in seconds)

<table>
<thead>
<tr>
<th>Second party clients</th>
<th>Brokers</th>
<th>Web service instances</th>
<th>First party clients</th>
<th>Number of states in the full state-space</th>
<th>Number of states in the aggregated state-space</th>
<th>Sparse matrix steady-state</th>
<th>Matrix/MTBDD steady-state</th>
<th>Transient solution for time $t = 100$</th>
<th>MCMC simulation one run to $t = 100$</th>
<th>ODE solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>1.04</td>
<td>1.10</td>
<td>1.01</td>
<td>2.47</td>
<td>2.81</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6,304</td>
<td>860</td>
<td>2.15</td>
<td>2.26</td>
<td>2.31</td>
<td>2.45</td>
<td>2.81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1,130,496</td>
<td>161,296</td>
<td>172.48</td>
<td>255.48</td>
<td>588.80</td>
<td>2.48</td>
<td>2.83</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>$&gt;234M$</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>500</td>
<td>1000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

- **Second party clients**: Number of second party clients in the scenario.
- **Brokers**: Number of brokers in the scenario.
- **Web service instances**: Number of web service instances in the scenario.
- **First party clients**: Number of first party clients in the scenario.
- **Number of states in the full state-space**: Number of states in the full state-space.
- **Number of states in the aggregated state-space**: Number of states in the aggregated state-space.
- **Sparse matrix steady-state**: Running time for Sparse matrix steady-state.
- **Matrix/MTBDD steady-state**: Running time for Matrix/MTBDD steady-state.
- **Transient solution for time $t = 100$**: Running time for transient solution at $t = 100$.
- **MCMC simulation one run to $t = 100$**: Running time for MCMC simulation one run to $t = 100$.
- **ODE solution**: Running time for ODE solution.
Second party clients

Second party Client

- Decoding
- Encoding
- Idle
- Sending
- Waiting
Stochastic Analysis in PEPA

Case study in Web Services

Analysis

Brokers

![Diagram showing the analysis of brokers with different states over time.]

- Decoding input
- Decoding response
- Encoding input
- Encoding response
- Idle
- Replying
- Sending
- Waiting
First party clients
Web service
Outline

1. Performance modelling with process algebras
   - Performance Evaluation Process Algebra

2. Comparing performance measures
   - Computed with continuous time
   - Computed with continuous space
   - Comparison of computed measures

3. Case study in Web Services
   - Description
   - Analysis

4. Commentary and comparison
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit \textit{steady-state} and \textit{transient} analysis (by solving the CTMC).
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit *steady-state* and *transient* analysis (by solving the CTMC).

Steady-state is cheaper but less informative. Transient is more informative but more expensive.
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit *steady-state* and *transient* analysis (by solving the CTMC).

Steady-state is cheaper but less informative. Transient is more informative but more expensive.

Major drawback: state-space explosion. Generating the state-space is slow. Solving the CTMC is slow.
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit steady-state and transient analysis (by solving the CTMC).

Steady-state is cheaper but less informative. Transient is more informative but more expensive.

Major drawback: state-space explosion. Generating the state-space is slow. Solving the CTMC is slow.

In practice effective only to systems of size $10^6$ states, even when using clever storage representations.
Mapping PEPA to ODEs admits \textit{course-of-values} analysis by solving the ODE (akin to transient analysis).
Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).

Major benefit: avoids state-space generation entirely.
Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).

Major benefit: avoids state-space generation entirely.

Major benefit: ODE solving is effective in practice, leaning towards suitability for interactive experimentation. Good for modellers, gaining more insights into the system behaviour.
Commentary and comparison

- Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).
- Major benefit: avoids state-space generation entirely.
- Major benefit: ODE solving is effective in practice, leaning towards suitability for interactive experimentation. Good for modellers, gaining more insights into the system behaviour.
- Effective for systems of size $10^{10^6}$ states and beyond.
Markov chain modelling with PEPA

J. Hillston.
A Compositional Approach to Performance Modelling.

J. Hillston.
Tuning systems: From composition to performance.
The Needham Lecture paper.
ODE-based modelling with PEPA

J. Hillston.
Fluid flow approximation of PEPA models.

Mario Bravetti, Stephen Gilmore, Claudio Guidi, and Mirco Tribastone.
Replicating web services for scalability.