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

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).

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).

$(\alpha, r).P$	Prefix
$P_1 + P_2$	Choice
$P_1 \underset{L}{\bowtie} P_2$	Co-operation
P/L	Hiding
X	Variable

PEPA: informal semantics (sequential sublanguage)

$(\alpha, r).S$

The activity (α, r) takes time Δt (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)

$C_1 \underset{L}{\bowtie} C_2$

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 (τ type activities) to be thought of as internal delays.

PEPA and Markov processes

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, $\boldsymbol{Q}$.

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

$$\pi \boldsymbol{Q} = \mathbf{0}$$

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.

$$
\begin{aligned}
& \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_1) = x_1) \\
= \; & \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n)
\end{aligned}
$$

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?

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?

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

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?

$$\begin{aligned} \Pr(T > t + s \mid T > t) &= \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \end{aligned}$$

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?

$$\begin{aligned}
\Pr(T > t + s \mid T > t) &= \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)} \\
&= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\
&= e^{-\lambda s}
\end{aligned}$$

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?

$$\begin{aligned} \Pr(T > t + s \mid T > t) &= \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\ &= e^{-\lambda s} \end{aligned}$$

This value is independent of t (and so the time already spent has not been remembered).

The importance of being exponential

$$
\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}
$$

The importance of being exponential

$$
\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}
$$

The importance of being exponential

$$
\begin{tikzcd}
& (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "(\alpha, r)"'] \arrow[dr, "(\beta, s)"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "(\beta, s)"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "(\alpha, r)"] \\
& Stop \parallel Stop &
\end{tikzcd}
$$

The importance of being exponential

$$
\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "(\alpha, r)"'] \arrow[dd, "(\alpha, r)"', "(\beta, s)"] \arrow[dr, "(\beta, s)"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "(\beta, s)"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "(\alpha, r)"] \\
 & Stop \parallel Stop &
\end{tikzcd}
$$

The importance of being exponential

$$
\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}
$$

The importance of being exponential

$$
\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}
$$

The importance of being exponential

$$\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}$$

The importance of being exponential

$$
\begin{tikzcd}
& (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "(\alpha, r)"'] \arrow[dr, "(\beta, s)"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "(\beta, s)"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "(\alpha, r)"] \\
& Stop \parallel Stop &
\end{tikzcd}
$$

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

The importance of being exponential

$$
\begin{tikzcd}
 & (\alpha, r).Stop \parallel (\beta, s).Stop \arrow[dl, "{(\alpha, r)}"'] \arrow[dr, "{(\beta, s)}"] & \\
Stop \parallel (\beta, s).Stop \arrow[dr, "{(\beta, s)}"'] & & (\alpha, r).Stop \parallel Stop \arrow[dl, "{(\alpha, r)}"] \\
 & Stop \parallel Stop &
\end{tikzcd}
$$

We retain the expansion law of classical process algebra:

$$
\begin{aligned}
&(\alpha, r).Stop \parallel (\beta, s).Stop = \\
&\qquad (\alpha, r).(\beta, s).(Stop \parallel Stop) + (\beta, s).(\alpha, r).(Stop \parallel Stop)
\end{aligned}
$$

only if the negative exponential distribution is assumed.

Outline

Computing performance measures: CTMCs

Queue example

$$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_i \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_{i+1} + (\mathit{serve}, \mu).Q_{i-1}$$

$$Q_8 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_7 \qquad (0 < i < 8)$$

A queue with arrivals at rate λ, service at rate μ and capacity 8 (thus $0 \leq \text{len} < 9$).

Computing performance measures: CTMCs

Queue example

$$Q_0 \stackrel{def}{=} (arrive, \lambda).Q_1 \qquad Q_i \stackrel{def}{=} (arrive, \lambda).Q_{i+1} + (serve, \mu).Q_{i-1}$$
$$Q_8 \stackrel{def}{=} (serve, \mu).Q_7 \qquad (0 < i < 8)$$

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

0	0.7500	3	0.0117	6	0.0000
1	0.1875	4	0.0029	7	0.0000
2	0.0468	5	0.0007	8	0.0000

Computing performance measures: CTMCs

Queue example

$$Q_0 \stackrel{def}{=} (arrive, \lambda).Q_1 \qquad Q_i \stackrel{def}{=} (arrive, \lambda).Q_{i+1} + (serve, \mu).Q_{i-1}$$

$$Q_8 \stackrel{def}{=} (serve, \mu).Q_7 \qquad (0 < i < 8)$$

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

0	0.5009	3	0.0626	6	0.0078
1	0.2504	4	0.0313	7	0.0039
2	0.1252	5	0.0156	8	0.0019

Computing performance measures: CTMCs

Queue example

$$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_i \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_{i+1} + (\mathit{serve}, \mu).Q_{i-1}$$

$$Q_8 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_7 \qquad (0 < i < 8)$$

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

0	0.1111	3	0.1111	6	0.1111
1	0.1111	4	0.1111	7	0.1111
2	0.1111	5	0.1111	8	0.1111

Computing performance measures: CTMCs

Queue example

$$Q_0 \stackrel{def}{=} (arrive, \lambda).Q_1 \qquad Q_i \stackrel{def}{=} (arrive, \lambda).Q_{i+1} + (serve, \mu).Q_{i-1}$$
$$Q_8 \stackrel{def}{=} (serve, \mu).Q_7 \qquad (0 < i < 8)$$

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

0	0.0019	3	0.0156	6	0.1252
1	0.0039	4	0.0313	7	0.2504
2	0.0078	5	0.0626	8	0.5009

Computing performance measures: CTMCs

Queue example

$$Q_0 \stackrel{\text{def}}{=} (arrive, \lambda).Q_1 \qquad Q_i \stackrel{\text{def}}{=} (arrive, \lambda).Q_{i+1} + (serve, \mu).Q_{i-1}$$

$$Q_8 \stackrel{\text{def}}{=} (serve, \mu).Q_7 \qquad (0 < i < 8)$$

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

0	0.0000	3	0.0007	6	0.0468
1	0.0000	4	0.0029	7	0.1875
2	0.0000	5	0.0117	8	0.7500

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.

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

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at equilibrium)
1	4	0.3333

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at equilibrium)
1	4	0.3333
1	2	0.9824

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at equilibrium)
1	4	0.3333
1	2	0.9824
1	1	4.0000

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at equilibrium)
1	4	0.3333
1	2	0.9824
1	1	4.0000
2	1	7.0176

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at equilibrium)
1	4	0.3333
1	2	0.9824
1	1	4.0000
2	1	7.0176
4	1	7.6667

Calculating average queue length: CTMCs

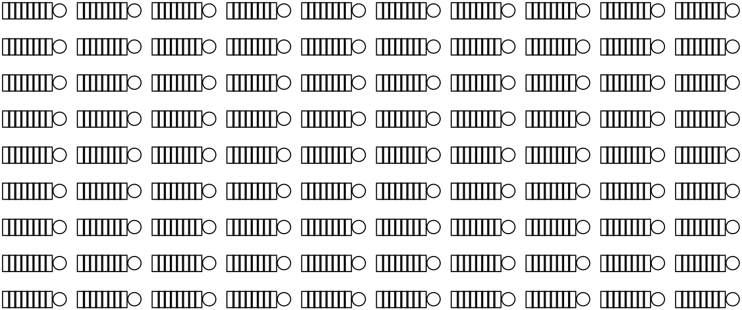
To calculate the average queue length, weight the probability of a state by the number of customers in the queue at that point.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at equilibrium)
1	4	0.3333
1	2	0.9824
1	1	4.0000
2	1	7.0176
4	1	7.6667

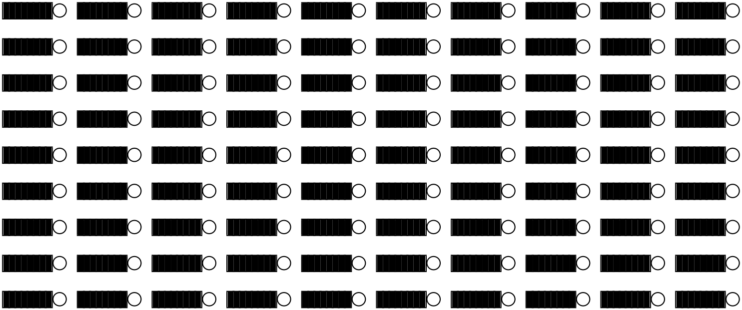
Queues and differential equations

CTMC: 

ODEs: 

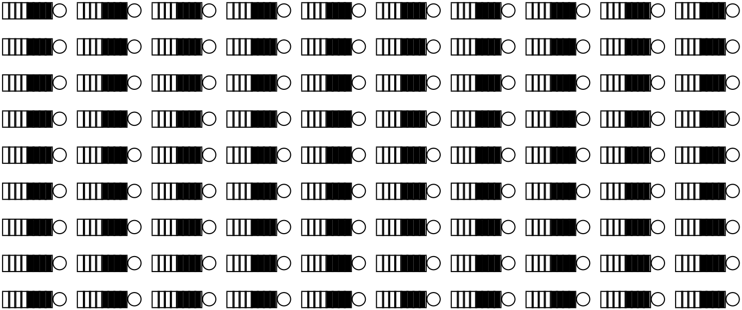
Queues and differential equations

CTMC: 

ODEs: 

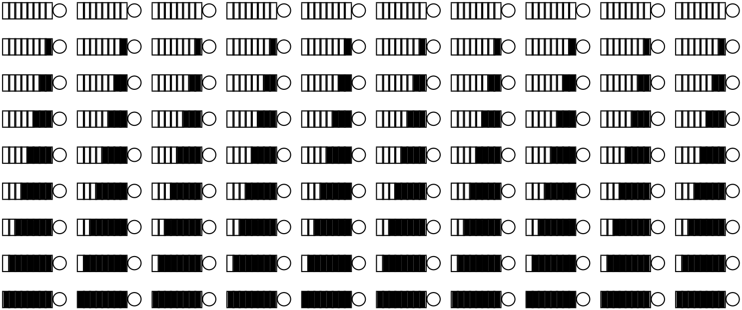
Queues and differential equations

CTMC: 

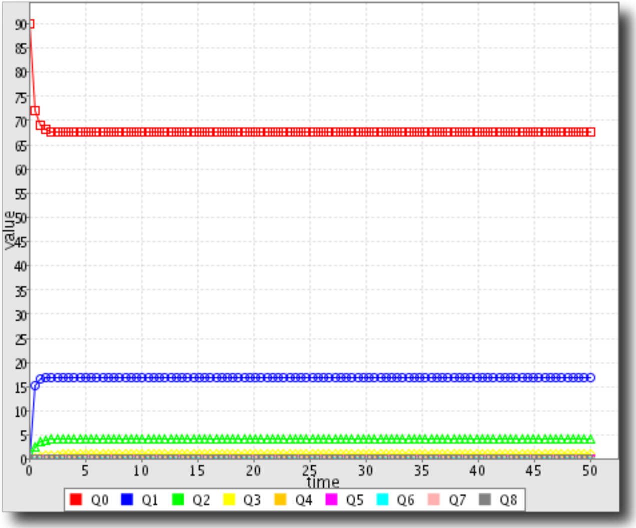
ODEs: 

Queues and differential equations

CTMC: 

ODEs: 

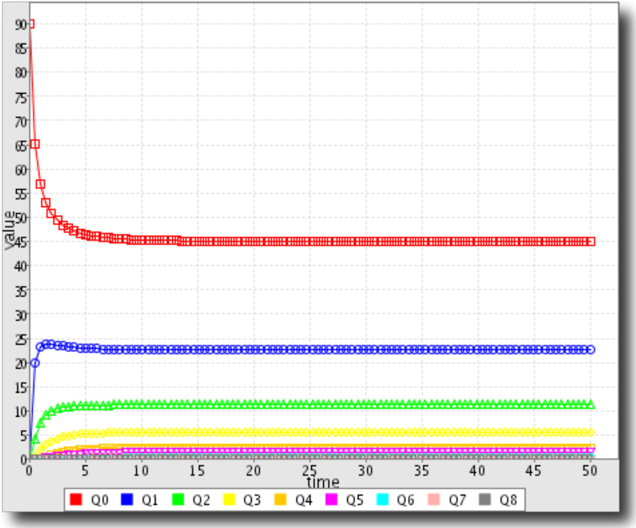
Computing performance measures: ODEs



$\lambda = 1$

$\mu = 4$

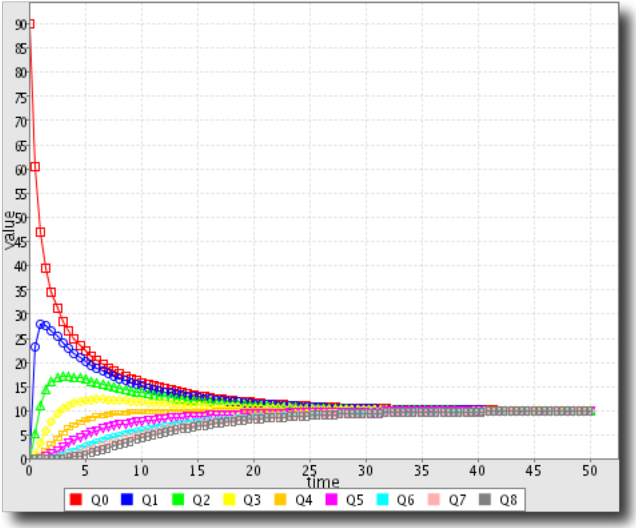
Computing performance measures: ODEs



$\lambda = 1$

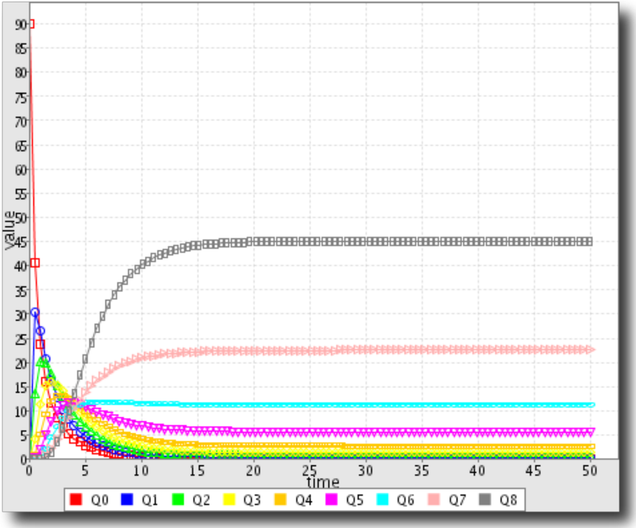
$\mu = 2$

Computing performance measures: ODEs



$$\lambda = 1$$
$$\mu = 1$$

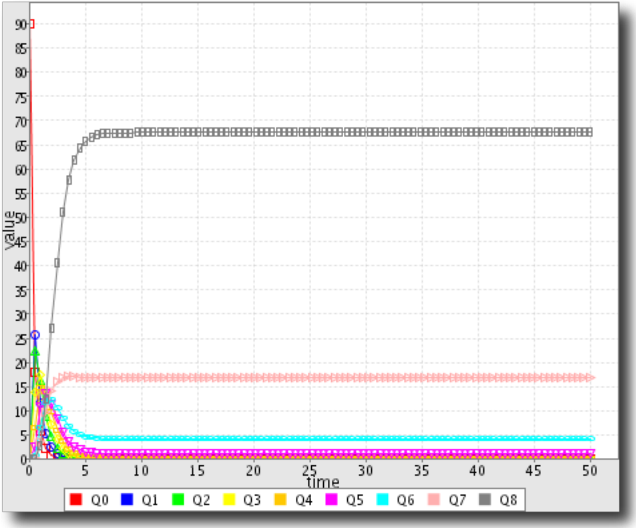
Computing performance measures: ODEs



$\lambda = 2$

$\mu = 1$

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.

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

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at $t = 50$)
1	4	0.3333

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at $t = 50$)
1	4	0.3333
1	2	0.9824

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at $t = 50$)
1	4	0.3333
1	2	0.9824
1	1	3.9914

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at $t = 50$)
1	4	0.3333
1	2	0.9824
1	1	3.9914
2	1	7.0176

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at $t = 50$)
1	4	0.3333
1	2	0.9824
1	1	3.9914
2	1	7.0176
4	1	7.6667

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.

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

Arrival rate (λ)	Service rate (μ)	Av. queue length (at $t = 50$)
1	4	0.3333
1	2	0.9824
1	1	3.9914
2	1	7.0176
4	1	7.6667

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 50$)	Difference
1	4	0.333299009029	0.333298624889	3.8×10^{-7}

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 50$)	Difference
1	4	0.333299009029	0.333298624889	3.8×10^{-7}
1	2	0.982387959648	0.982387242222	7.1×10^{-7}

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 50$)	Difference
1	4	0.333299009029	0.333298624889	3.8×10^{-7}
1	2	0.982387959648	0.982387242222	7.1×10^{-7}
1	1	4.000000000000	3.991409877780	8.6×10^{-3}

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 50$)	Difference
1	4	0.333299009029	0.333298624889	3.8×10^{-7}
1	2	0.982387959648	0.982387242222	7.1×10^{-7}
1	1	4.000000000000	3.991409877780	8.6×10^{-3}
2	1	7.017612040350	7.017612412220	-3.7×10^{-7}

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 50$)	Difference
1	4	0.333299009029	0.333298624889	3.8×10^{-7}
1	2	0.982387959648	0.982387242222	7.1×10^{-7}
1	1	4.000000000000	3.991409877780	8.6×10^{-3}
2	1	7.017612040350	7.017612412220	-3.7×10^{-7}
4	1	7.666700990970	7.666701341490	-3.5×10^{-7}

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 50$)	Difference
1	4	0.333299009029	0.333298624889	3.8×10^{-7}
1	2	0.982387959648	0.982387242222	7.1×10^{-7}
1	1	4.000000000000	3.991409877780	8.6×10^{-3}
2	1	7.017612040350	7.017612412220	-3.7×10^{-7}
4	1	7.666700990970	7.666701341490	-3.5×10^{-7}

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 100$)	Difference
1	4	0.333299009029	0.333298736822	2.7×10^{-7}
1	2	0.982387959648	0.982387201111	7.6×10^{-7}
1	1	4.000000000000	3.999979511110	2.0×10^{-5}
2	1	7.017612040350	7.017613132220	-1.1×10^{-6}
4	1	7.666700990970	7.666701089580	-9.8×10^{-8}

Comparison of computed measures

λ	μ	Av. queue length (CTMCs at equilibrium)	Av. queue length (ODEs at $t = 200$)	Difference
1	4	0.333299009029	0.333298753978	2.5×10^{-7}
1	2	0.982387959648	0.982386995556	9.6×10^{-7}
1	1	4.000000000000	4.000000266670	-2.6×10^{-7}
2	1	7.017612040350	7.017613704440	-1.6×10^{-6}
4	1	7.666700990970	7.666701306580	-3.2×10^{-7}

Small queue example: CTMCs

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

Small queue example: CTMCs

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

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

Small queue example: CTMCs

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

$$\mathbf{Q} = \begin{bmatrix} -\lambda & \lambda & 0 \\ \mu & -\lambda-\mu & \lambda \\ 0 & \mu & -\mu \end{bmatrix} \qquad \boxed{\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}}$$

Small queue example: CTMCs

Small queue example

$Q_0 \stackrel{def}{=} (arrive, \lambda).Q_1 \qquad Q_1 \stackrel{def}{=} (arrive, \lambda).Q_2 + (serve, \mu).Q_0$

$Q_2 \stackrel{def}{=} (serve, \mu).Q_1$

$$\mathbf{Q} = \begin{bmatrix} -\lambda & \lambda & 0 \\ \mu & -\lambda-\mu & \lambda \\ 0 & \mu & -\mu \end{bmatrix} \qquad \boxed{\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}} \qquad \boxed{\sum \boldsymbol{\pi} = 1}$$

Small queue example: CTMCs

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

$$\mathbf{Q} = \begin{bmatrix} -\lambda & \lambda & 0 \\ \mu & -\lambda-\mu & \lambda \\ 0 & \mu & -\mu \end{bmatrix} \quad \boxed{\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}} \quad \boxed{\sum \boldsymbol{\pi} = 1}$$

$$\boldsymbol{\pi} = \left[\frac{\mu^2}{\lambda^2 + \mu\,\lambda + \mu^2}, \frac{\mu\,\lambda}{\lambda^2 + \mu\,\lambda + \mu^2}, \frac{\lambda^2}{\lambda^2 + \mu\,\lambda + \mu^2}\right]$$

Small queue example: ODEs

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

Small queue example: ODEs

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

$$\frac{dQ_0}{dt} = -\lambda Q_0 + \mu Q_1$$

Small queue example: ODEs

Small queue example

$Q_0 \stackrel{def}{=} (arrive, \lambda).Q_1 \qquad Q_1 \stackrel{def}{=} (arrive, \lambda).Q_2 + (serve, \mu).Q_0$
$Q_2 \stackrel{def}{=} (serve, \mu).Q_1$

$$\begin{aligned} \frac{dQ_0}{dt} &= -\lambda Q_0 + \mu Q_1 \\ \frac{dQ_1}{dt} &= \lambda Q_0 - \lambda Q_1 - \mu Q_1 + \mu Q_2 \end{aligned}$$

Small queue example: ODEs

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

$$
\begin{aligned}
\frac{dQ_0}{dt} &= -\lambda Q_0 + \mu Q_1 \\
\frac{dQ_1}{dt} &= \lambda Q_0 - \lambda Q_1 - \mu Q_1 + \mu Q_2 \\
\frac{dQ_2}{dt} &= \lambda Q_1 - \mu Q_2
\end{aligned}
$$

Small queue example: ODEs (stationary points)

Small queue example

$$Q_0 \stackrel{\text{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\text{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$$

$$Q_2 \stackrel{\text{def}}{=} (\mathit{serve}, \mu).Q_1$$

$$\begin{aligned}
0 &= -\lambda Q_0 + \mu Q_1 \\
0 &= \lambda Q_0 - \lambda Q_1 - \mu Q_1 + \mu Q_2 \\
0 &= \lambda Q_1 - \mu Q_2
\end{aligned}$$

Small queue example: ODEs (stationary points)

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

$$\mathbf{0} = [Q_0 \quad Q_1 \quad Q_2] \begin{bmatrix} -\lambda & \lambda & 0 \\ \mu & -\lambda - \mu & \lambda \\ 0 & \mu & -\mu \end{bmatrix}$$

Small queue example: ODEs (and CTMC solution)

Small queue example

$$Q_0 \stackrel{def}{=} (arrive, \lambda).Q_1 \qquad Q_1 \stackrel{def}{=} (arrive, \lambda).Q_2 + (serve, \mu).Q_0$$
$$Q_2 \stackrel{def}{=} (serve, \mu).Q_1$$

$$\mathbf{p} \quad = \quad [Q_0 \quad \frac{\lambda}{\mu} Q_0 \quad \frac{\lambda^2}{\mu^2} Q_0]$$

Small queue example: ODEs (and CTMC solution)

Small queue example

$Q_0 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_1 \qquad Q_1 \stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).Q_2 + (\mathit{serve}, \mu).Q_0$

$Q_2 \stackrel{\mathit{def}}{=} (\mathit{serve}, \mu).Q_1$

$$
\begin{aligned}
\mathbf{p} &= [Q_0 \quad \frac{\lambda}{\mu} Q_0 \quad \frac{\lambda^2}{\mu^2} Q_0] \\
\boldsymbol{\pi} &= \left[\frac{\mu^2}{\lambda^2 + \mu\,\lambda + \mu^2}, \frac{\mu\,\lambda}{\lambda^2 + \mu\,\lambda + \mu^2}, \frac{\lambda^2}{\lambda^2 + \mu\,\lambda + \mu^2}\right]
\end{aligned}
$$

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

$$\frac{d\pi(t)}{dt} = \pi(t)Q \text{ ?}$$

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

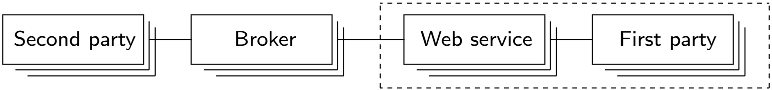
$$\mathbf{Q} = \begin{bmatrix} -2\,\lambda & \lambda & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & -2\,\lambda - \mu & 0 & \lambda & \lambda & 0 & 0 & 0 & 0 \\ \mu & 0 & -2\,\lambda - \mu & 0 & \lambda & 0 & 0 & 0 & \lambda \\ 0 & \mu & 0 & -\lambda - \mu & 0 & \lambda & 0 & 0 & 0 \\ 0 & \mu & \mu & 0 & -2\,\lambda - 2\,\mu & \lambda & 0 & \lambda & 0 \\ 0 & 0 & 0 & \mu & \mu & -\lambda - 2\,\mu & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu & -2\,\mu & \mu & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 & \lambda & -\lambda - 2\,\mu & \mu \\ 0 & 0 & \mu & 0 & 0 & 0 & 0 & \lambda & -\lambda - \mu \end{bmatrix}$$

Steady-state for two parallel queues

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

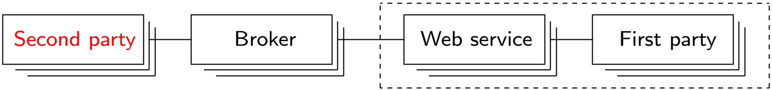
Outline

Example: Secure Web Service use



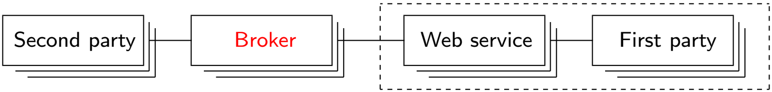
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service requests via trusted brokers.

PEPA model: Second party clients



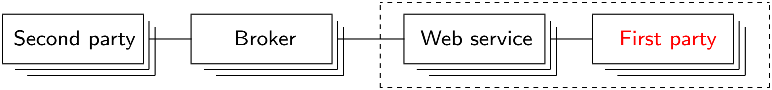
$$
\begin{aligned}
SPC_{idle} &\stackrel{def}{=} (compose_{sp}, r_{sp\_cmp}).SPC_{enc} \\
SPC_{enc} &\stackrel{def}{=} (encrypt_{b}, r_{sp\_encb}).SPC_{sending} \\
SPC_{sending} &\stackrel{def}{=} (request_{b}, r_{sp\_req}).SPC_{waiting} \\
SPC_{waiting} &\stackrel{def}{=} (response_{b}, \top).SPC_{dec} \\
SPC_{dec} &\stackrel{def}{=} (decrypt_{b}, r_{sp\_decb}).SPC_{idle}
\end{aligned}
$$

PEPA model: Brokers



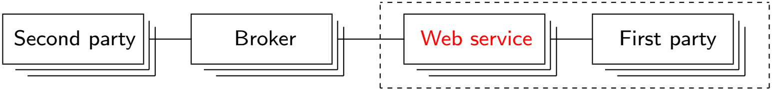
$$
\begin{aligned}
Broker_{idle} &\stackrel{def}{=} (request_b, \top).Broker_{dec\_input} \\
Broker_{dec\_input} &\stackrel{def}{=} (decrypt_{sp}, r_{b\_dec\_sp}).Broker_{enc\_input} \\
Broker_{enc\_input} &\stackrel{def}{=} (encrypt_{ws}, r_{b\_enc\_ws}).Broker_{sending} \\
Broker_{sending} &\stackrel{def}{=} (request_{ws}, r_{b\_req}).Broker_{waiting} \\
Broker_{waiting} &\stackrel{def}{=} (response_{ws}, \top).Broker_{dec\_resp} \\
Broker_{dec\_resp} &\stackrel{def}{=} (decrypt_{ws}, r_{b\_dec\_ws}).Broker_{enc\_resp} \\
Broker_{enc\_resp} &\stackrel{def}{=} (encrypt_{sp}, r_{b\_enc\_sp}).Broker_{replying} \\
Broker_{replying} &\stackrel{def}{=} (response_b, r_{b\_resp}).Broker_{idle}
\end{aligned}
$$

PEPA model: First party clients



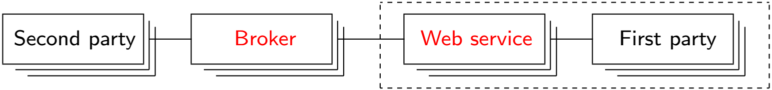
$$
\begin{aligned}
FPC_{idle} &\stackrel{def}{=} (compose_{fp}, r_{fp\_cmp}).FPC_{calling} \\
FPC_{calling} &\stackrel{def}{=} (invoke_{ws}, r_{fp\_inv}).FPC_{blocked} \\
FPC_{blocked} &\stackrel{def}{=} (result_{ws}, \top).FPC_{idle}
\end{aligned}
$$

PEPA model: Web service



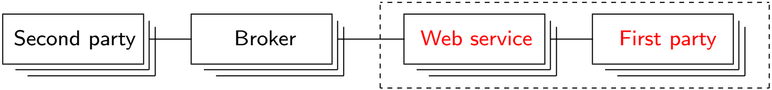
$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\phantom{\stackrel{def}{=}}+ (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



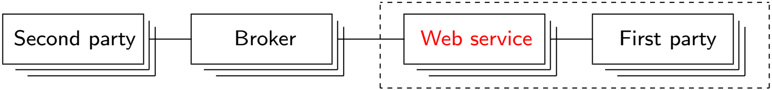
$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

PEPA model: Web service



$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&+ (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing} &\stackrel{def}{=} (encryptResp_{ws}, r_{ws\_enc\_b}).WS_{responding} \\
WS_{responding} &\stackrel{def}{=} (response_{ws}, r_{ws\_resp\_b}).WS_{idle} \\
WS_{method} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{returning} \\
WS_{returning} &\stackrel{def}{=} (result_{ws}, r_{ws\_res}).WS_{idle}
\end{aligned}
$$

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.

$$System \stackrel{\mathit{def}}{=} (SPC_{idle} \underset{\mathcal{K}}{\bowtie} Broker_{idle}) \underset{\mathcal{L}}{\bowtie} (WS_{idle} \underset{\mathcal{M}}{\bowtie} FPC_{idle})$$

where
$$\begin{aligned} \mathcal{K} &= \{\, request_b, response_b \,\} \\ \mathcal{L} &= \{\, request_{ws}, response_{ws} \,\} \\ \mathcal{M} &= \{\, invoke_{ws}, result_{ws} \,\} \end{aligned}$$

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.

$$
\mathit{System} \stackrel{\mathit{def}}{=} (\mathit{SPC}_{\mathit{idle}} \underset{\mathcal{K}}{\bowtie} \mathit{Broker}_{\mathit{idle}}) \underset{\mathcal{L}}{\bowtie} (\mathit{WS}_{\mathit{idle}} \underset{\mathcal{M}}{\bowtie} \mathit{FPC}_{\mathit{idle}})
$$

$$
\begin{array}{lrcl}
\text{where} & \mathcal{K} & = & \{\, \mathit{request}_b, \mathit{response}_b \,\} \\
 & \mathcal{L} & = & \{\, \mathit{request}_{ws}, \mathit{response}_{ws} \,\} \\
 & \mathcal{M} & = & \{\, \mathit{invoke}_{ws}, \mathit{result}_{ws} \,\}
\end{array}
$$

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)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81

Running times from analyses (in seconds)

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

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	$>$234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78

Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77

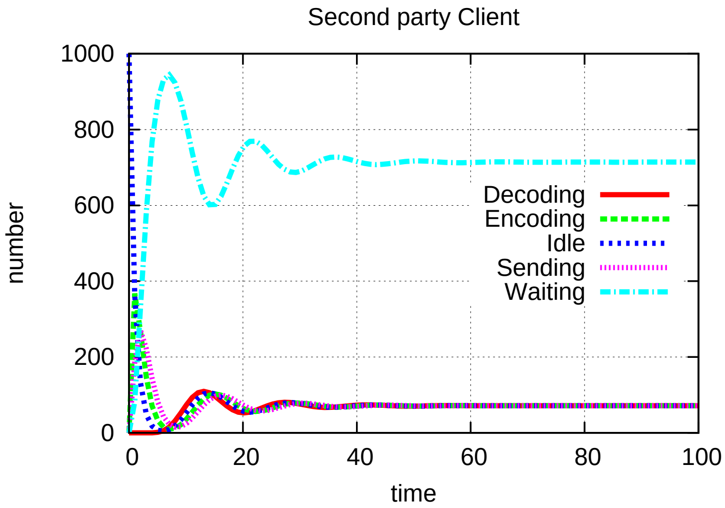
Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77
1000	1000	1000	1000	–	–	–	–	–	5.44	2.77

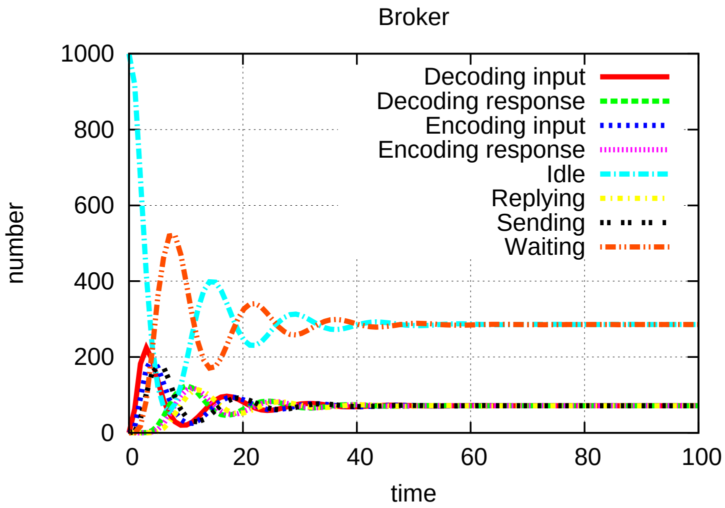
Running times from analyses (in seconds)

Second party clients	Brokers	Web service instances	First party clients	Number of states in the full state-space	Number of states in the aggregated state-space	Sparse matrix steady-state	Matrix/MTBDD steady-state	Transient solution for time $t = 100$	MCMC simulation one run to $t = 100$	ODE solution
1	1	1	1	48	48	1.04	1.10	1.01	2.47	2.81
2	2	2	2	6,304	860	2.15	2.26	2.31	2.45	2.81
3	3	3	3	1,130,496	161,296	172.48	255.48	588.80	2.48	2.83
4	4	4	4	>234M	–	–	–	–	2.44	2.85
100	100	100	100	–	–	–	–	–	2.78	2.78
1000	100	500	1000	–	–	–	–	–	3.72	2.77
1000	1000	1000	1000	–	–	–	–	–	5.44	2.77

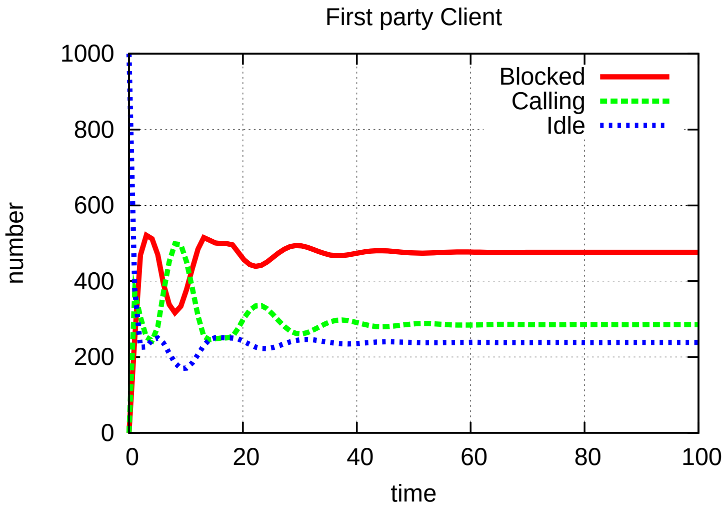
Second party clients



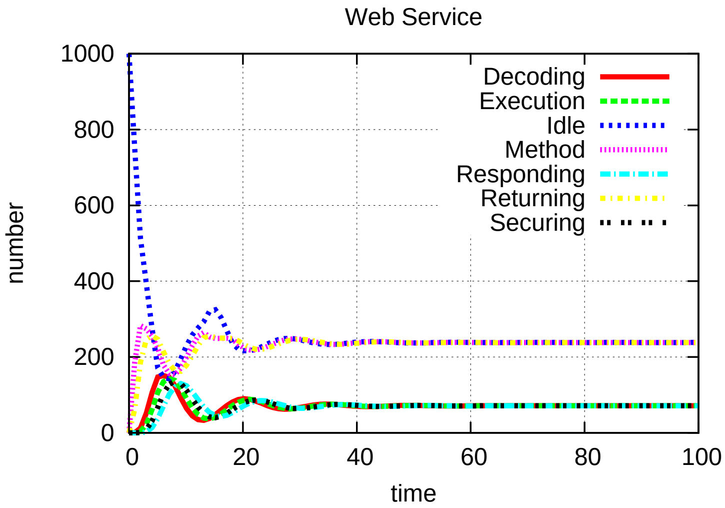
Brokers


Broker
number
time
0
200
400
600
800
1000
0
20
40
60
80
100
Decoding input
Decoding response
Encoding input
Encoding response
Idle
Replying
Sending
Waiting

First party clients



Web service


Web Service
1000
800
600
400
200
0
number
0
20
40
60
80
100
time
Decoding
Execution
Idle
Method
Responding
Returning
Securing

Outline

Commentary and comparison

- Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit *steady-state* and *transient* analysis (by solving the CTMC).

Commentary and comparison

- 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.

Commentary and comparison

- 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.

Commentary and comparison

- 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.

Commentary and comparison

- Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).

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.

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.

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.
Cambridge University Press, 1996.

J. Hillston.
Tuning systems: From composition to performance.
The Computer Journal, 48(4):385–400, May 2005.
The Needham Lecture paper.

ODE-based modelling with PEPA

J. Hillston.
Fluid flow approximation of PEPA models.
In *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems*, pages 33–43, Torino, Italy, September 2005. IEEE Computer Society Press.

Mario Bravetti, Stephen Gilmore, Claudio Guidi, and Mirco Tribastone.
Replicating web services for scalability.
In G. Barthe and C. Fournet, editors, *Proceedings of the Third International Conference on Trustworthy Global Computing (TGC'07)*, volume 4912 of *LNCS*, pages 204–221. Springer-Verlag, 2008.