Fluid Flow Approximation of PEPA Models

Jane Hillston.
LFCS, University of Edinburgh

20th September 2005

Outline

Outline

Performance evaluation: new mathematical structures

When constructing a model performance modellers usually make a choice between:

- Closed form analytical models (queueing networks);

Performance evaluation: new mathematical structures

When constructing a model performance modellers usually make a choice between:

- Closed form analytical models (queueing networks);
- Simulations; or

Performance evaluation: new mathematical structures

When constructing a model performance modellers usually make a choice between:

- Closed form analytical models (queueing networks);
- Simulations; or
- Numerical solution of continuous time Markov chains (CTMC) (Stochastic Petri nets, Stochastic process algebras etc.)

Performance evaluation: new mathematical structures

When constructing a model performance modellers usually make a choice between:

- Closed form analytical models (queueing networks);
- Simulations; or
- Numerical solution of continuous time Markov chains (CTMC) (Stochastic Petri nets, Stochastic process algebras etc.)

In each case the model has a discrete state space and continuous time.

Performance evaluation: new mathematical structures

When constructing a model performance modellers usually make a choice between:

- Closed form analytical models (queueing networks);
- Simulations; or
- Numerical solution of continuous time Markov chains (CTMC) (Stochastic Petri nets, Stochastic process algebras etc.)

In each case the model has a discrete state space and continuous time.

The major limitations of the CTMC approach are the state space explosion problem and the reliance on exponential distributions.

Performance evaluation: new mathematical structures

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Performance evaluation: new mathematical structures

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Is there an alternative?

Performance evaluation: new mathematical structures

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Is there an alternative?

Use continuous state variables to approximate the discrete state space.

Performance evaluation: new mathematical structures

State space explosion becomes an ever more challenging problem as the scale and complexity of modern systems increase.

Is there an alternative?

Use continuous state variables to approximate the discrete state space.

Use ordinary differential equations to represent the evolution of those variables over time.

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha,\ r).P$$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a CTMC for performance modelling.

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a CTMC for performance modelling.

PEPA MODEL

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a CTMC for performance modelling.

PEPA MODEL $\xrightarrow{\text{SOS rules}}$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a CTMC for performance modelling.

PEPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a CTMC for performance modelling.

PEPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM $\xrightarrow[\text{diagram}]{\text{state transition}}$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

The language may be used to generate a CTMC for performance modelling.

$$\text{PEPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}} \text{CTMC } \mathbf{Q}$$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate

component/ derivative

The language may be used to generate a system of ordinary differential equations (ODEs) for performance modelling.

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate

component/ derivative

The language may be used to generate a system of ordinary differential equations (ODEs) for performance modelling.

PEPA MODEL

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate

component/ derivative

The language may be used to generate a system of ordinary differential equations (ODEs) for performance modelling.

PEPA MODEL $\xrightarrow[\text{analysis}]{\text{syntactic}}$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate

component/ derivative

The language may be used to generate a system of ordinary differential equations (ODEs) for performance modelling.

$$\text{PEPA MODEL} \xrightarrow[\text{analysis}]{\text{syntactic}} \text{ACTIVITY MATRIX}$$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name

activity rate

component/ derivative

The language may be used to generate a system of ordinary differential equations (ODEs) for performance modelling.

$$\text{PEPA MODEL} \xrightarrow[\text{analysis}]{\text{syntactic}} \text{ACTIVITY MATRIX} \xrightarrow[\text{interpretation}]{\text{continuous}}$$

Performance Evaluation Process Algebra

- Models are constructed from components which engage in activities.

$$(\alpha, r).P$$

action type or name — α
activity rate — r
component/derivative — P

The language may be used to generate a system of ordinary differential equations (ODEs) for performance modelling.

$$\text{PEPA MODEL} \xrightarrow[\text{analysis}]{\text{syntactic}} \text{ACTIVITY MATRIX} \xrightarrow[\text{interpretation}]{\text{continuous}} \text{ODEs}$$

PEPA

$$
\begin{aligned}
S &::= (\alpha, r).S \mid S + S \mid A \\
P &::= S \mid P \underset{L}{\bowtie} P \mid P/L
\end{aligned}
$$

PEPA

$$
\begin{aligned}
S & ::= (\alpha, r).S \mid S + S \mid A \\
P & ::= S \mid P \underset{L}{\bowtie} P \mid P/L
\end{aligned}
$$

PREFIX: $(\alpha, r).S$ designated first action

PEPA

$$
\begin{aligned}
S &::= (\alpha, r).S \mid S + S \mid A \\
P &::= S \mid P \underset{L}{\bowtie} P \mid P/L
\end{aligned}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)

PEPA

$$
\begin{aligned}
S &::= (\alpha, r).S \mid S + S \mid A \\
P &::= S \mid P \underset{L}{\bowtie} P \mid P/L
\end{aligned}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)
CONSTANT:	$A \stackrel{\mathit{def}}{=} S$	assigning names

PEPA

$$
\begin{aligned}
S &::= (\alpha, r).S \mid S + S \mid A \\
P &::= S \mid P \underset{L}{\bowtie} P \mid P/L
\end{aligned}
$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)
CONSTANT:	$A \stackrel{\mathit{def}}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ concurrent activity (*individual actions*) $\alpha \in L$ cooperative activity (*shared actions*)

PEPA

$$\begin{array}{rcl} S & ::= & (\alpha, r).S \mid S + S \mid A \\ P & ::= & S \mid P \bowtie_{L} P \mid P/L \end{array}$$

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components (race policy)
CONSTANT:	$A \stackrel{def}{=} S$	assigning names
COOPERATION:	$P \bowtie_{L} P$	$\alpha \notin L$ concurrent activity (*individual actions*) $\alpha \in L$ cooperative activity (*shared actions*)
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$

Outline

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.

New mathematical structures: differential equations

- Use a more abstract state representation rather than the CTMC complete state space.
- Assume that these state variables are subject to continuous rather than discrete change.
- No longer aim to calculate the probability distribution over the entire state space of the model.

Appropriate for models in which there are large numbers of components of the same type.

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation

Differential equations from PEPA models

- In a PEPA model the state at any current time is the local derivative or state of each component of the model.
- We can represent the state of the system as the count of the current number of each possible local derivative or component type.
- We can approximate the behaviour of the model by treating each count as a continuous variable, and the state of the model as a whole as the set of such variables.
- The evolution of each count variable can then be described by an ordinary differential equation (assuming rates are deterministic).

Differential equations from PEPA models

- The PEPA definitions of the component specify the activities which can increase or decrease the number of components exhibited in the current state.
- The cooperations show when the number of instances of another component will have an influence on the evolution of this component.

Differential equations from PEPA models

- The PEPA definitions of the component specify the activities which can increase or decrease the number of components exhibited in the current state.
- The cooperations show when the number of instances of another component will have an influence on the evolution of this component.

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t + \delta t) \; - \; N(\mathcal{C}_{i_j}, t) =$$

$$- \underbrace{\sum_{(\alpha,r) \in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l} \in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t)) \, \delta t}_{\text{exit activities}}$$

$$+ \underbrace{\sum_{(\alpha,r) \in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l} \in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t)) \, \delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t + \delta t) \; - \; N(\mathcal{C}_{i_j}, t) =$$

$$\underbrace{- \sum_{(\alpha,r) \in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l} \in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{exit activities}}$$

$$+ \underbrace{\sum_{(\alpha,r) \in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l} \in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t+\delta t) \; - \; N(\mathcal{C}_{i_j}, t) =$$

$$- \underbrace{\sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{exit activities}}$$

$$+ \underbrace{\sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t+\delta t) \;-\; N(\mathcal{C}_{i_j}, t) =$$

$$-\underbrace{\sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\,\delta t}_{\text{exit activities}}$$

$$+\underbrace{\sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\,\delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Consider the change in a small time δt:

$$N(\mathcal{C}_{i_j}, t+\delta t) \; - \; N(\mathcal{C}_{i_j}, t) =$$

$$- \underbrace{\sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{exit activities}}$$

$$+ \underbrace{\sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(\mathcal{C}_{k_l}, t))\, \delta t}_{\text{entry activities}}$$

Differential equations from PEPA models

Let $N(\mathcal{C}_{i_j}, t)$ denote the number of $\mathcal{C}_{i_j}$ type components at time t.

Dividing by δt and taking the limit, $\delta t \longrightarrow 0$:

$$\frac{dN(C_{i_j}, t)}{dt} = -\sum_{(\alpha,r)\in Ex(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t)) + \sum_{(\alpha,r)\in En(\mathcal{C}_{i_j})} r \times \min_{\mathcal{C}_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t))$$

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Activity matrix

Derivation of the system of ODEs representing the PEPA model then proceeds via an activity matrix which records the influence of each activity on each component type/derivative.

The matrix has one row for each component type and one column for each activity type.

One ODE is generated corresponding to each row of the matrix, taking into account the negative entries in the non-zero columns as these are the components for which this is an exit activity.

Modelling with quantified process algebras

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Modelling with quantified process algebras

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

This example defines a system with nine reachable states:

1. $P_1 \parallel P_1$
2. $P_1 \parallel P_2$
3. $P_1 \parallel P_3$
4. $P_2 \parallel P_1$
5. $P_2 \parallel P_2$
6. $P_2 \parallel P_3$
7. $P_3 \parallel P_1$
8. $P_3 \parallel P_2$
9. $P_3 \parallel P_3$

The transitions between states have quantified duration r which can be evaluated against a CTMC or ODE interpretation.

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 0$:

1. 1.0000
2. 0.0000
3. 0.0000
4. 0.0000
5. 0.0000
6. 0.0000
7. 0.0000
8. 0.0000
9. 0.0000

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 1$:

1. 0.1642
2. 0.1567
3. 0.0842
4. 0.1567
5. 0.1496
6. 0.0804
7. 0.0842
8. 0.0804
9. 0.0432

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 2$:

1. 0.1056
2. 0.1159
3. 0.1034
4. 0.1159
5. 0.1272
6. 0.1135
7. 0.1034
8. 0.1135
9. 0.1012

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{\textit{def}}{=} (\textit{start}, r).P_2 \qquad P_2 \stackrel{\textit{def}}{=} (\textit{run}, r).P_3 \qquad P_3 \stackrel{\textit{def}}{=} (\textit{stop}, r).P_1$$

$$\textit{System} \stackrel{\textit{def}}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 3$:

1. 0.1082
2. 0.1106
3. 0.1100
4. 0.1106
5. 0.1132
6. 0.1125
7. 0.1100
8. 0.1125
9. 0.1119

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (stop, r).P_1$$

$$System \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 4$:

1. 0.1106
2. 0.1108
3. 0.1111
4. 0.1108
5. 0.1110
6. 0.1113
7. 0.1111
8. 0.1113
9. 0.1116

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 5$:

1. 0.1111
2. 0.1110
3. 0.1111
4. 0.1110
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 6$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Continuous-time Markov Chains

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (stop, r).P_1$$

$$System \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 7$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1111
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (stop, r).P_1$$

$$System \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 0$:

P_1	2.0000
P_2	0.0000
P_3	0.0000

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{\text{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\text{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\text{def}}{=} (stop, r).P_1$$

$$System \stackrel{\text{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 1$:

P_1	0.8121
P_2	0.7734
P_3	0.4144

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 2$:

P_1	0.6490
P_2	0.7051
P_3	0.6457

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 3$:

P_1	0.6587
P_2	0.6719
P_3	0.6692

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 4$:

P_1	0.6648
P_2	0.6665
P_3	0.6685

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 5$:

P_1	0.6666
P_2	0.6663
P_3	0.6669

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (stop, r).P_1$$

$$System \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 6$:

P_1	0.6666
P_2	0.6666
P_3	0.6666

Analysis based on Ordinary Differential Equations

Tiny example

$$P_1 \stackrel{\mathit{def}}{=} (\mathit{start}, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (\mathit{run}, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (\mathit{stop}, r).P_1$$

$$\mathit{System} \stackrel{\mathit{def}}{=} (P_1 \parallel P_1)$$

Using the ordinary differential equation semantics we can compute the expected number of each type of component.

For $t = 7$:

P_1	0.6666
P_2	0.6666
P_3	0.6666

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 0$:

P_1	3.0000
P_2	0.0000
P_3	0.0000

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 1$:

P_1	1.1782
P_2	1.1628
P_3	0.6590

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (stop, r).P_1$$

$$System \stackrel{\mathit{def}}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 2$:

P_1	0.9766
P_2	1.0754
P_3	0.9479

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{\text{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\text{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\text{def}}{=} (stop, r).P_1$$

$$System \stackrel{\text{def}}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 3$:

P_1	0.9838
P_2	1.0142
P_3	1.0020

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{\mathit{def}}{=} (start, r).P_2 \qquad P_2 \stackrel{\mathit{def}}{=} (run, r).P_3 \qquad P_3 \stackrel{\mathit{def}}{=} (stop, r).P_1$$

$$System \stackrel{\mathit{def}}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 4$:

P_1	0.9981
P_2	0.9995
P_3	1.0023

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 5$:

P_1	1.0001
P_2	0.9996
P_3	1.0003

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 6$:		
	P_1	1.0001
	P_2	0.9999
	P_3	1.0000

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 7$:

P_1	1.0000
P_2	0.9999
P_3	0.9999

Analysis based on Ordinary Differential Equations

Slightly larger example

$$P_1 \stackrel{def}{=} (start, r).P_2 \qquad P_2 \stackrel{def}{=} (run, r).P_3 \qquad P_3 \stackrel{def}{=} (stop, r).P_1$$

$$System \stackrel{def}{=} (P_1 \parallel P_1 \parallel P_1)$$

A slightly larger example with a third copy of the process also initiated in state P_1.

For $t = 8$:

P_1	1.0000
P_2	1.0000
P_3	1.0000

Isn't this just the Chapman-Kolmogorov equations?

It is possible to perform transient analysis of a continuous-time Markov chain by solving the Chapman-Kolmogorov differential equations:

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

[Stewart, 1994]

Isn't this just the Chapman-Kolmogorov equations?

It is possible to perform transient analysis of a continuous-time Markov chain by solving the Chapman-Kolmogorov differential equations:

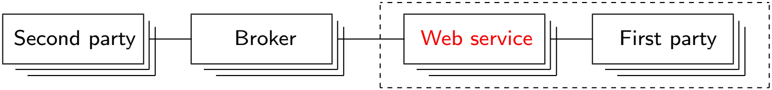
$$\frac{d\pi(t)}{dt} = \pi(t)Q$$

[Stewart, 1994]

That's not what we're doing. We go directly to ODEs.

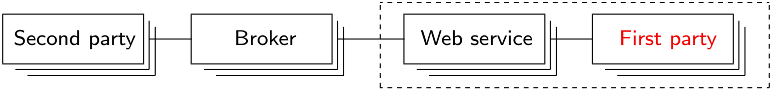
Outline

Example: Secure Web Service use



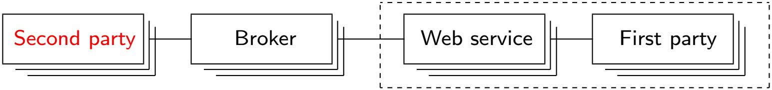
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service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

Example: Secure Web Service use



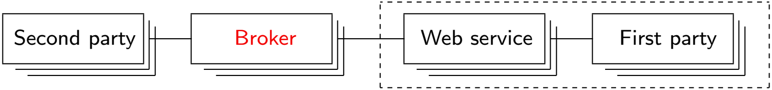
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service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

Example: Secure Web Service use



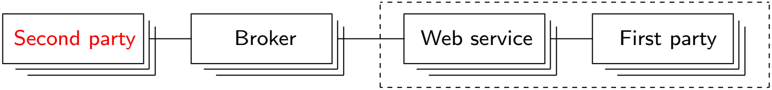
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service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

Example: Secure Web Service use



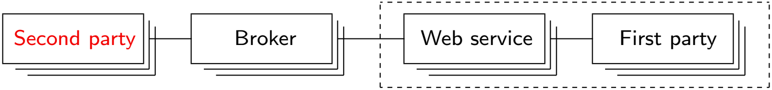
- The example which we consider is a Web service which has two types of clients:
  - first party application clients which access the web service across a secure intranet, and
  - second party browser clients which access the Web service across the Internet.
- Second party clients route their service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

PEPA model: Second party clients and Brokers



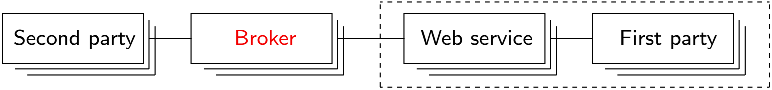
- A second party client composes service requests, encrypts these and sends them to its broker.

PEPA model: Second party clients and Brokers



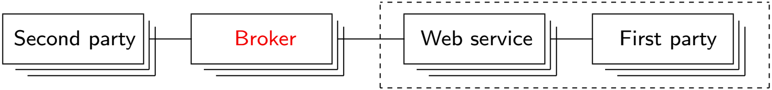
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.

PEPA model: Second party clients and Brokers



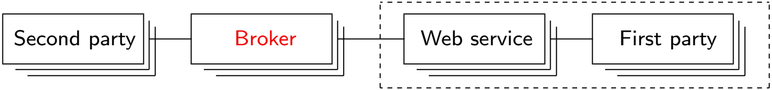
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.
- Brokers add decryption and encryption steps to build end-to-end security from point-to-point security.

PEPA model: Second party clients and Brokers



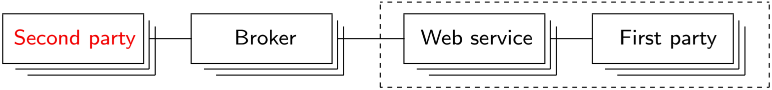
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.
- Brokers add decryption and encryption steps to build end-to-end security from point-to-point security.
- The broker forwards the request to the Web service and then waits for a response.

PEPA model: Second party clients and Brokers



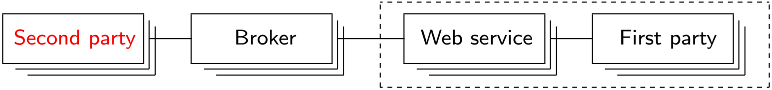
- A second party client composes service requests, encrypts these and sends them to its broker.
- It then waits for a response from the broker.
- Brokers add decryption and encryption steps to build end-to-end security from point-to-point security.
- The broker forwards the request to the Web service and then waits for a response.
- Decryption and re-encrytion are performed before returning the response to the client.

PEPA model: Second party clients



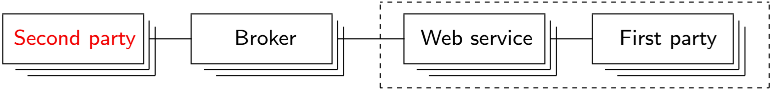
$$
\begin{aligned}
SPC_{idle} &\stackrel{\mathit{def}}{=} (compose_{sp}, r_{sp\_cmp}).SPC_{enc} \\
SPC_{enc} &\stackrel{\mathit{def}}{=} (encrypt_{b}, r_{sp\_encb}).SPC_{sending} \\
SPC_{sending} &\stackrel{\mathit{def}}{=} (request_{b}, r_{sp\_req}).SPC_{waiting} \\
SPC_{waiting} &\stackrel{\mathit{def}}{=}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

PEPA model: Second party clients



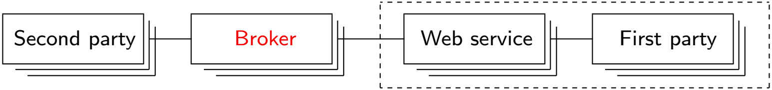
$$
\begin{aligned}
SPC_{idle} &\stackrel{def}{=} (compose_{sp}, r_{sp\_cmp}).SPC_{enc} \\
SPC_{enc} &\stackrel{def}{=} (encrypt_{b}, r_{sp\_encb}).SPC_{sending} \\
SPC_{sending} &\stackrel{def}{=} (request_{b}, r_{sp\_req}).SPC_{waiting} \\
SPC_{waiting} &\stackrel{def}{=} (response_{b}, \top).SPC_{dec}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

PEPA model: Second party clients



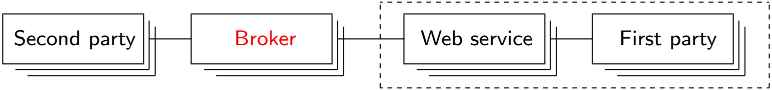
$$
\begin{aligned}
SPC_{idle} &\stackrel{def}{=} (compose_{sp}, r_{sp\_cmp}).SPC_{enc} \\
SPC_{enc} &\stackrel{def}{=} (encrypt_{b}, r_{sp\_encb}).SPC_{sending} \\
SPC_{sending} &\stackrel{def}{=} (request_{b}, r_{sp\_req}).SPC_{waiting} \\
SPC_{waiting} &\stackrel{def}{=} (response_{b}, \top).SPC_{dec}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

PEPA model: Brokers



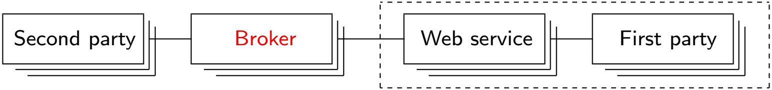
$$
\begin{aligned}
\color{red}{Broker_{idle}} & \color{red}{\stackrel{def}{=}} \color{red}{(request_b, \top).Broker_{dec\_input}} \\
Broker_{dec\_input} & \stackrel{def}{=} (decrypt_{sp}, r_{b\_dec\_sp}).Broker_{enc\_input} \\
Broker_{enc\_input} & \stackrel{def}{=} (encrypt_{ws}, r_{b\_enc\_ws}).Broker_{sending}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

PEPA model: Brokers



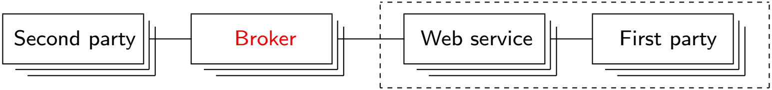
$$
\begin{aligned}
Broker_{idle} &\stackrel{def}{=} (request_b, \top).Broker_{dec\_input} \\
Broker_{dec\_input} &\stackrel{def}{=} (decrypt_{sp}, r_{b\_dec\_sp}).Broker_{enc\_input} \\
Broker_{enc\_input} &\stackrel{def}{=} (encrypt_{ws}, r_{b\_enc\_ws}).Broker_{sending} \\
Broker_{sending} &\stackrel{def}{=}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

PEPA model: Brokers



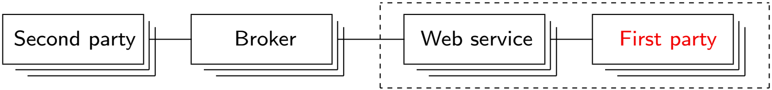
$$
\begin{aligned}
Broker_{idle} &\stackrel{def}{=} (request_b, \top).Broker_{dec\_input} \\
Broker_{dec\_input} &\stackrel{def}{=} (decrypt_{sp}, r_{b\_dec\_sp}).Broker_{enc\_input} \\
Broker_{enc\_input} &\stackrel{def}{=} (encrypt_{ws}, r_{b\_enc\_ws}).Broker_{sending} \\
Broker_{sending} &\stackrel{def}{=}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

PEPA model: Brokers



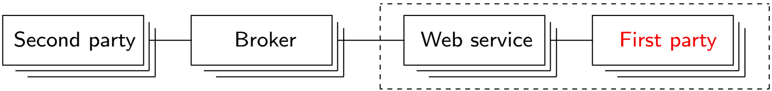
$$
\begin{aligned}
Broker_{idle} &\stackrel{def}{=} (request_b, \top).Broker_{dec\_input} \\
Broker_{dec\_input} &\stackrel{def}{=} (decrypt_{sp}, r_{b\_dec\_sp}).Broker_{enc\_input} \\
Broker_{enc\_input} &\stackrel{def}{=} (encrypt_{ws}, r_{b\_enc\_ws}).Broker_{sending} \\
Broker_{sending} &\stackrel{def}{=}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

PEPA model: First party clients



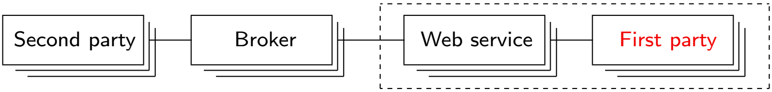
- The lifetime of a first party client mirrors that of a second party client except that encryption need not be used when all of the communication is conducted across a secure intranet.

PEPA model: First party clients



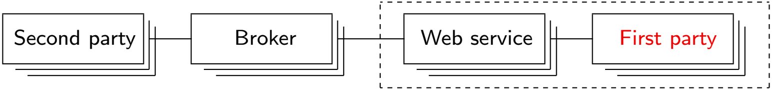
- The lifetime of a first party client mirrors that of a second party client except that encryption need not be used when all of the communication is conducted across a secure intranet.
- Also the service may be invoked by a remote method invocation to the host machine instead of via HTTP.

PEPA model: First party clients



- The lifetime of a first party client mirrors that of a second party client except that encryption need not be used when all of the communication is conducted across a secure intranet.
- Also the service may be invoked by a remote method invocation to the host machine instead of via HTTP.
- Thus the first party client experiences the Web service as a blocking remote method invocation.

PEPA model: First party clients

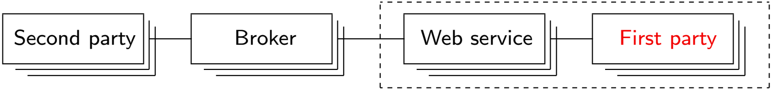


$$FPC_{idle} \stackrel{def}{=} (compose_{fp}, r_{fp\_cmp}).FPC_{calling}$$

$$FPC_{calling} \stackrel{def}{=} (invoke_{ws}, r_{fp\_inv}).FPC_{blocked}$$

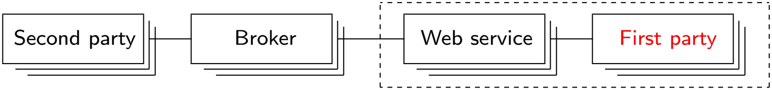
$$FPC_{blocked} \stackrel{def}{=} (result_{ws}, \top).FPC_{idle}$$

PEPA model: First party clients



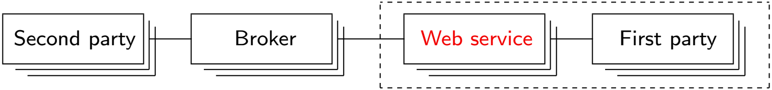
$$
\begin{aligned}
FPC_{idle} &\stackrel{def}{=} (compose_{fp}, r_{fp\_cmp}).FPC_{calling} \\
FPC_{calling} &\stackrel{def}{=} (invoke_{ws}, r_{fp\_inv}).FPC_{blocked} \\
FPC_{blocked} &\stackrel{def}{=} (result_{ws}, \top).FPC_{idle}
\end{aligned}
$$

PEPA model: First party clients



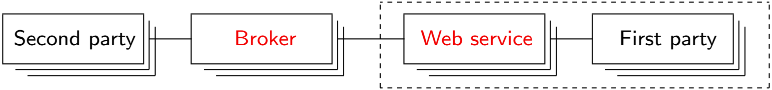
$$
\begin{aligned}
FPC_{idle} &\stackrel{def}{=} (compose_{fp}, r_{fp\_cmp}).FPC_{calling}\\
FPC_{calling} &\stackrel{def}{=} (invoke_{ws}, r_{fp\_inv}).FPC_{blocked}\\
FPC_{blocked} &\stackrel{def}{=} (result_{ws}, \top).FPC_{idle}
\end{aligned}
$$

PEPA model: Web service



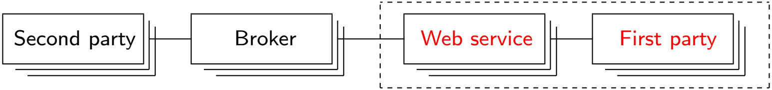
$$
\begin{array}{rcl}
WS_{idle} & \stackrel{def}{=} & (request_{ws}, \top).WS_{decoding} \\
 & + & (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} & \stackrel{def}{=} & (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} & \stackrel{def}{=} & (execute_{ws}, r_{ws\_exec}).WS_{securing}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

PEPA model: Web service



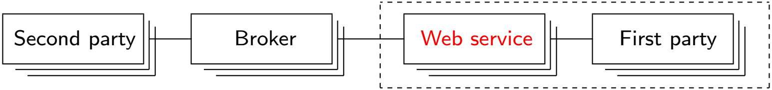
$$
\begin{aligned}
WS_{idle} &\stackrel{def}{=} (request_{ws}, \top).WS_{decoding} \\
&\quad + (invoke_{ws}, \top).WS_{method} \\
WS_{decoding} &\stackrel{def}{=} (decryptReq_{ws}, r_{ws\_dec\_b}).WS_{execution} \\
WS_{execution} &\stackrel{def}{=} (execute_{ws}, r_{ws\_exec}).WS_{securing} \\
WS_{securing}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

PEPA model: Web service



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

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: System composition

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

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

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

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.

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

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

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

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

Cost of analysis

- We compare ODE-based evaluation against other techniques which could be used to analyse the model.
  - Steady-state and transient analysis as implemented by the PRISM probabilistic model-checker.

Cost of analysis

- We compare ODE-based evaluation against other techniques which could be used to analyse the model.
  - Steady-state and transient analysis as implemented by the PRISM probabilistic model-checker.
  - Monte Carlo Markov Chain simulation (a Java implementation of Gillespie's Direct Method).

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

Time series analysis via ODEs

- We assume a system in which the number of clients of both kinds, brokers, and web service instances are all 1000.

Time series analysis via ODEs

- We assume a system in which the number of clients of both kinds, brokers, and web service instances are all 1000.
- We present the results from our ODE integrator as time-series plots of the number of each type of component behaviour as a function of time, as time runs from $t = 0$ to $t = 100$.

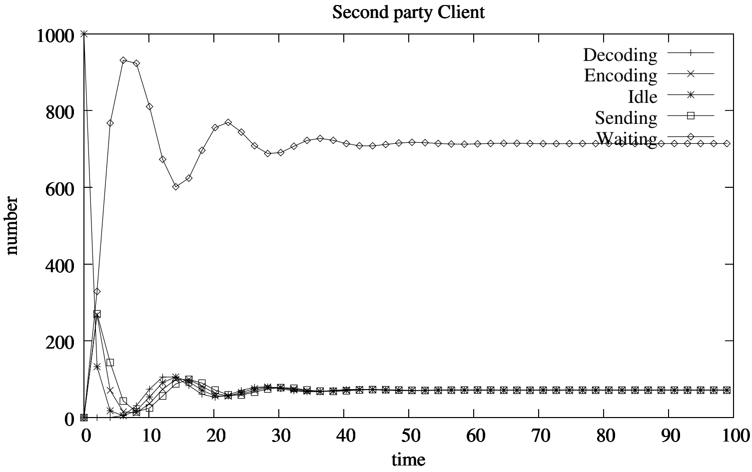
Time series analysis via ODEs

- We assume a system in which the number of clients of both kinds, brokers, and web service instances are all 1000.
- We present the results from our ODE integrator as time-series plots of the number of each type of component behaviour as a function of time, as time runs from $t = 0$ to $t = 100$.
- The graphs show fluctuations in the numbers of components with respect to time.

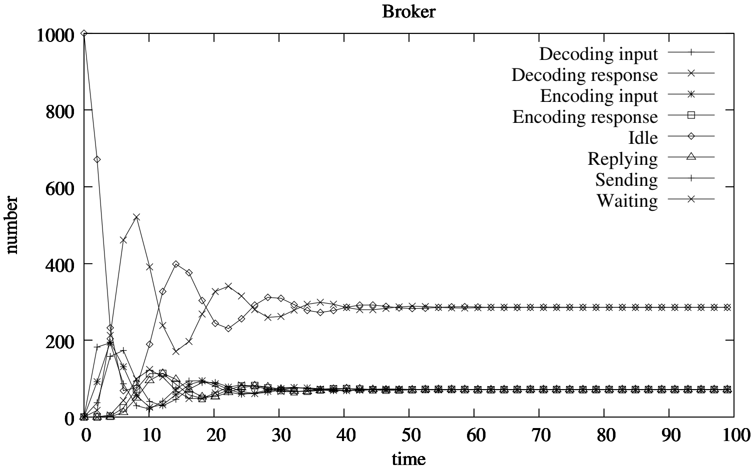
Time series analysis via ODEs

- We assume a system in which the number of clients of both kinds, brokers, and web service instances are all 1000.
- We present the results from our ODE integrator as time-series plots of the number of each type of component behaviour as a function of time, as time runs from $t = 0$ to $t = 100$.
- The graphs show fluctuations in the numbers of components with respect to time.
- We can observe an initial flurry of activity until the system stabilises into its steady-state equilibrium at time (around) $t = 50$.

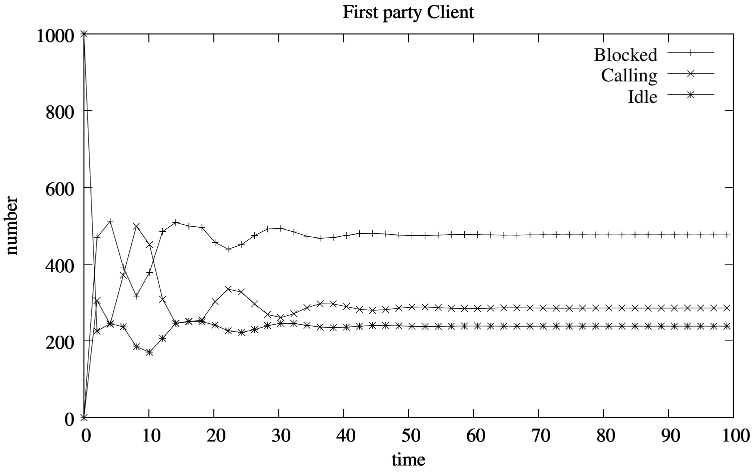
Second party clients



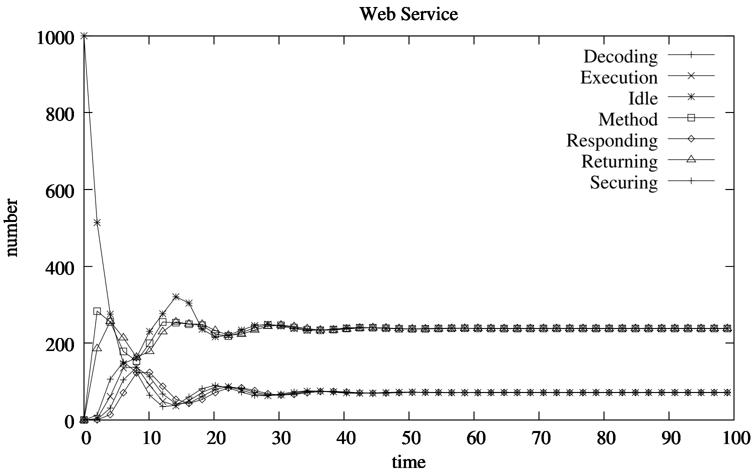
Brokers


Broker
number
time
Decoding input
Decoding response
Encoding input
Encoding response
Idle
Replying
Sending
Waiting
0
200
400
600
800
1000
0
10
20
30
40
50
60
70
80
90
100

First party clients



Web service


Web Service
1000
800
600
400
200
0
number
0
10
20
30
40
50
60
70
80
90
100
time
Decoding
Execution
Idle
Method
Responding
Returning
Securing

Comparison with Continuous-Time Markov Chain solution

- Direct comparison of results is not possible.

Comparison with Continuous-Time Markov Chain solution

- Direct comparison of results is not possible.
- For systems with the same blocking characteristics (1000 instances of all components vs 1 instance of each component) we compared values of performance measures derived via each of the two models.

Comparison with Continuous-Time Markov Chain solution

- Direct comparison of results is not possible.
- For systems with the same blocking characteristics (1000 instances of all components vs 1 instance of each component) we compared values of performance measures derived via each of the two models.
- We found good agreement between the CTMC results and those results obtained by ODE solution after the system stabilises into its steady-state equilibrium.

Comparison with Continuous-Time Markov Chain solution

- Direct comparison of results is not possible.
- For systems with the same blocking characteristics (1000 instances of all components vs 1 instance of each component) we compared values of performance measures derived via each of the two models.
- We found good agreement between the CTMC results and those results obtained by ODE solution after the system stabilises into its steady-state equilibrium. Less than 1% difference.

Outline

Conclusions

- Unlike fluid queues and fluid stochastic Petri nets, these are wholly continuous models.

Conclusions

- Unlike fluid queues and fluid stochastic Petri nets, these are wholly continuous models.
- Closest to diffusion approximations in queueing networks or continuous Petri nets.

Conclusions

- Unlike fluid queues and fluid stochastic Petri nets, these are wholly continuous models.
- Closest to diffusion approximations in queueing networks or continuous Petri nets.
- Based on an assumption that there are large numbers of components of the same type within the model.

Conclusions

- Unlike fluid queues and fluid stochastic Petri nets, these are wholly continuous models.
- Closest to diffusion approximations in queueing networks or continuous Petri nets.
- Based on an assumption that there are large numbers of components of the same type within the model.
- Particularly good for when we want to model the behaviour of individuals but study the behaviour of populations.

Conclusions

- Unlike fluid queues and fluid stochastic Petri nets, these are wholly continuous models.
- Closest to diffusion approximations in queueing networks or continuous Petri nets.
- Based on an assumption that there are large numbers of components of the same type within the model.
- Particularly good for when we want to model the behaviour of individuals but study the behaviour of populations.
- The models are no longer stochastic, since service rates are assumed to be deterministic.

Future Work

- Relax some of the restrictions which are placed on the model:

Future Work

- Relax some of the restrictions which are placed on the model:
  - Repeated components do not need to be independent of each other;

Future Work

- Relax some of the restrictions which are placed on the model:
  - Repeated components do not need to be independent of each other;
  - Partners in cooperations can have different rates for the shared activity;

Future Work

- Relax some of the restrictions which are placed on the model:
  - Repeated components do not need to be independent of each other;
  - Partners in cooperations can have different rates for the shared activity;
- Reintroduce a stochastic element:

Future Work

- Relax some of the restrictions which are placed on the model:
  - Repeated components do not need to be independent of each other;
  - Partners in cooperations can have different rates for the shared activity;
- Reintroduce a stochastic element:
  - Use of random or stochastic differential equations;

Future Work

- Relax some of the restrictions which are placed on the model:
  - Repeated components do not need to be independent of each other;
  - Partners in cooperations can have different rates for the shared activity;
- Reintroduce a stochastic element:
  - Use of random or stochastic differential equations;
  - Limit the continuous element to a few continuous component types, with others having usual CTMC semantics (c.f. fluid models)