BCS Joint BCS-FACS and BCS Women Evening Seminar

Process Algebra for Collective Dynamics

Jane Hillston

Laboratory for Foundations of Computer Science
University of Edinburgh

10th December 2007

Outline

1. Introduction
   - Collective Dynamics
   - Process Algebra

2. Continuous Approximation
   - State variables
   - Numerical illustration

3. Examples
   - Case Study: Scalable Web Services
   - Internet worms

4. On-going and Future Work
   - Alternative Models
   - New Languages

Outline

1 Introduction
- Collective Dynamics
- Process Algebra

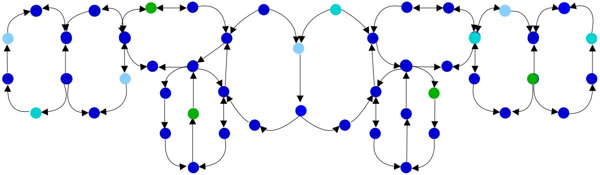
2 Continuous Approximation
- State variables
- Numerical illustration

3 Examples
- Case Study: Scalable Web Services
- Internet worms

4 On-going and Future Work
- Alternative Models
- New Languages

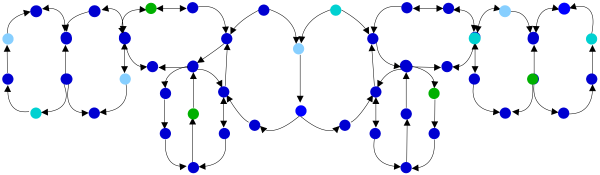
Collective Dynamics

The behaviour of many systems can be interpreted as the result of the collective behaviour of a large number of interacting entities.



Collective Dynamics

The behaviour of many systems can be interpreted as the result of the collective behaviour of a large number of interacting entities.



For such systems we are often as interested in the population level behaviour as we are in the behaviour of the individual entities.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

Process Algebra and Collective Dynamics

Process algebra are well-suited to modelling such systems

- Developed to represent concurrent behaviour compositionally;
- Capture the interactions between individuals explicitly;
- Incorporate formal apparatus for reasoning about the behaviour of systems;
- Stochastic extensions, such as PEPA, enable quantified behaviour of the dynamics of systems.

In the CODA project we are developing stochastic process algebras and associated theory, tailored to the construction and evaluation of the collective dynamics of large systems of interacting entities.

Novelty

The novelty in this project is twofold:

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Large scale software systems
Issues of scalability are important for user satisfaction and resource efficiency but such issues are difficult to investigate using discrete state models.

Novelty

The novelty in this project is twofold:

- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Biochemical signalling pathways

Understanding these pathways has the potential to improve the quality of life through enhanced drug treatment and better drug design.

Novelty

The novelty in this project is twofold:

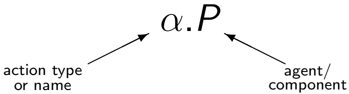
- Linking process algebra and continuous mathematical models for dynamic evaluation represents a paradigm shift in how such systems are studied.
- The prospect of formally-based quantified evaluation of dynamic behaviour could have significant impact in application domains such as:

Epidemiological systems

Improved modelling of these systems could lead to improved disease prevention and treatment in nature and better security in computer systems.

Process Algebra

- Models consist of agents which engage in actions.



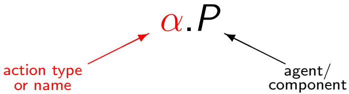
$\alpha.P$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.



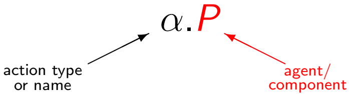
$\alpha.P$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.



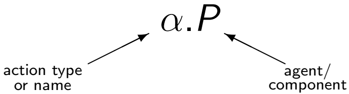
$\alpha.P$

action type or name

agent/ component

Process Algebra

- Models consist of agents which engage in actions.



$$\alpha.P$$

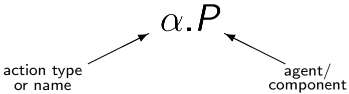
action type or name

agent/ component

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

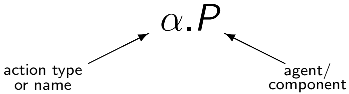
agent/ component

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model

Process Algebra

- Models consist of agents which engage in actions.



$\alpha.P$

action type or name

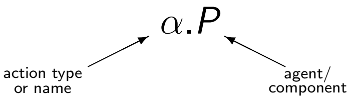
agent/ component

- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

Process algebra model $\xrightarrow{\text{SOS rules}}$

Process Algebra

- Models consist of agents which engage in actions.



- The structured operational (interleaving) semantics of the language is used to generate a labelled transition system.

$$\text{Process algebra model} \xrightarrow{\text{SOS rules}} \text{Labelled transition system}$$

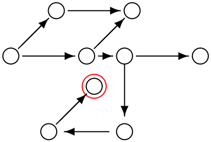
Qualitative Analysis

- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

Qualitative Analysis

- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

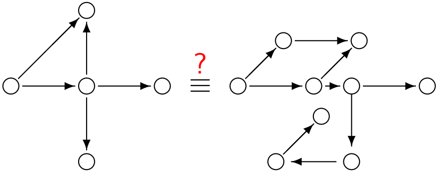
Will the system arrive in a particular state?



Qualitative Analysis

- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

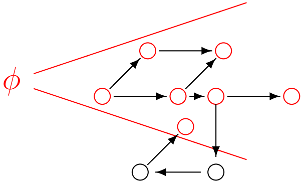
Does system behaviour match its specification?



Qualitative Analysis

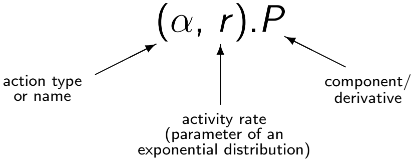
- The labelled transition system underlying a process algebra model can be used for functional verification e.g.: reachability analysis, specification matching and model checking.

Does a given property ϕ hold within the system?



Stochastic Process Algebra

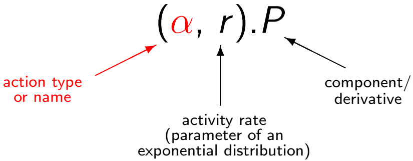
- Models are constructed from components which engage in activities.



$(\alpha, r).P$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

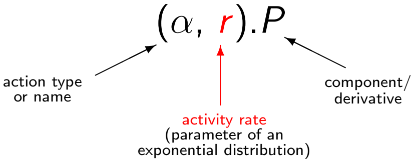
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

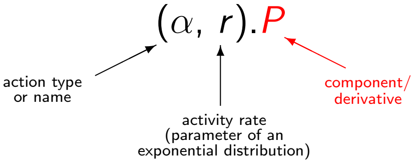
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

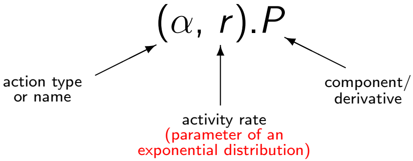
action type
or name

activity rate
(parameter of an
exponential distribution)

component/
derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



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

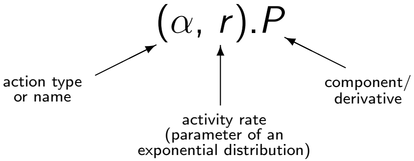
action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



$(\alpha, r).P$

action type or name

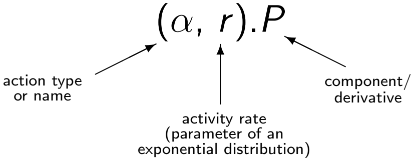
activity rate (parameter of an exponential distribution)

component/ derivative

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

Stochastic 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)

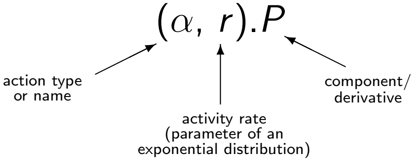
component/
derivative

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

PEPA
MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

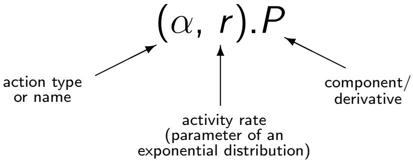


The language may be used to generate a CTMC for performance modelling.

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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



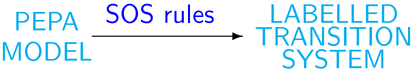
$(\alpha, r).P$

action type or name

activity rate (parameter of an exponential distribution)

component/ derivative

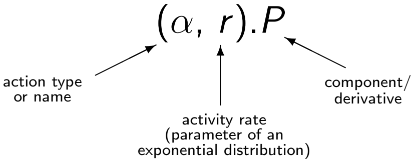
The language may be used to generate a CTMC for performance modelling.



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

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



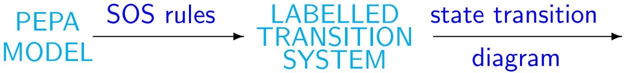
$(\alpha, r).P$

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

SOS rules

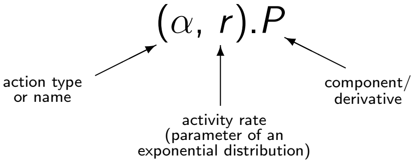
LABELLED TRANSITION SYSTEM

state transition

diagram

Stochastic 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

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.

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

Performance Evaluation Process Algebra

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

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

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

Performance Evaluation Process Algebra

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

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

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

When working with large numbers of entities, we write $P[n]$ to denote an array of n copies of P executing in parallel.

Performance Evaluation Process Algebra

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

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

$P_1 \parallel P_2$ is a derived form for $P_1 \underset{\emptyset}{\bowtie} P_2$.

When working with large numbers of entities, we write $P[n]$ to denote an array of n copies of P executing in parallel.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$

Outline

1 Introduction
- Collective Dynamics
- Process Algebra

2 Continuous Approximation
- State variables
- Numerical illustration

3 Examples
- Case Study: Scalable Web Services
- Internet worms

4 On-going and Future Work
- Alternative Models
- New Languages

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

Alternatively they may be studied using stochastic simulation. Each run generates a single trajectory through the state space. Many runs are needed in order to obtain average behaviours.

Solving discrete state models

Under the SOS semantics a PEPA model is mapped to a Continuous Time Markov Chain (CTMC) with global states determined by the local states of all the participating components.

When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

Alternatively they may be studied using stochastic simulation. Each run generates a single trajectory through the state space. Many runs are needed in order to obtain average behaviours.

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

○ ○ ○

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

Continuous Approximation

The major limitation of the CTMC approach is the state space explosion problem.

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

Use continuous state variables to approximate the discrete state space.

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

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, i.e. models of populations.

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, \top).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_1).Res_1 \\
Res_1 &\overset{def}{=} (reset, s).Res_0
\end{aligned}
$$

$$Proc_0[P] \underset{\{task1\}}{\bowtie} Res_0[R]$$

A simple example: processors and resources

$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, \top).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_1).Res_1 \\
Res_1 &\overset{def}{=} (reset, s).Res_0
\end{aligned}
$$

$$Proc_0[P] \underset{\{task1\}}{\bowtie} Res_0[R]$$

CTMC interpretation

Processors (P)	Resources (R)	States (2^{P+R})
1	1	4
2	1	8
2	2	16
3	2	32
3	3	64
4	3	128
4	4	256
5	4	512
5	5	1024
6	5	2048
6	6	4096
7	6	8192
7	7	16384
8	7	32768
8	8	65536
9	8	131072
9	9	262144
10	9	524288
10	10	1048576

A simple example: processors and resources

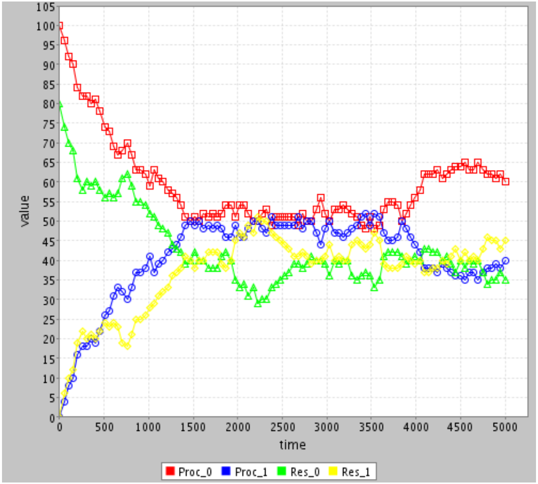
$$
\begin{aligned}
Proc_0 &\overset{def}{=} (task1, \top).Proc_1 \\
Proc_1 &\overset{def}{=} (task2, r_2).Proc_0 \\
Res_0 &\overset{def}{=} (task1, r_1).Res_1 \\
Res_1 &\overset{def}{=} (reset, s).Res_0
\end{aligned}
$$

$$Proc_0[P] \underset{\{task1\}}{\bowtie} Res_0[R]$$

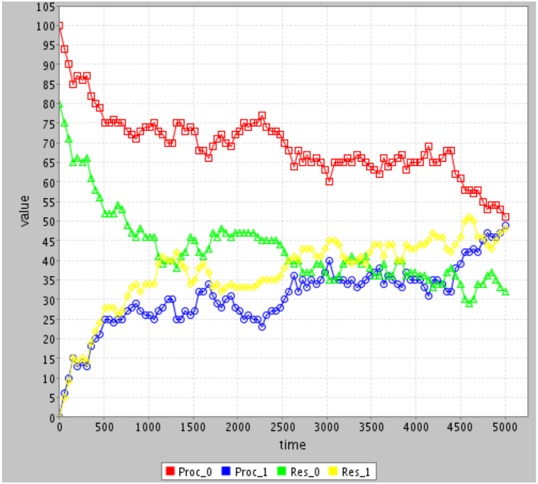
ODE interpretation

$$
\begin{aligned}
\frac{\mathrm{d}Proc_0}{\mathrm{d}t} &= -r_1 \min(Proc_0, Res_0) \\
&\quad +r_2\, Proc_1 \\
\frac{\mathrm{d}Proc_1}{\mathrm{d}t} &= r_1 \min(Proc_0, Res_0) \\
&\quad -r_2\, Proc_1 \\
\frac{\mathrm{d}Res_0}{\mathrm{d}t} &= -r_1 \min(Proc_0, Res_0) \\
&\quad +s\, Res_1 \\
\frac{\mathrm{d}Res_1}{\mathrm{d}t} &= r_1 \min(Proc_0, Res_0) \\
&\quad -s\, Res_1
\end{aligned}
$$

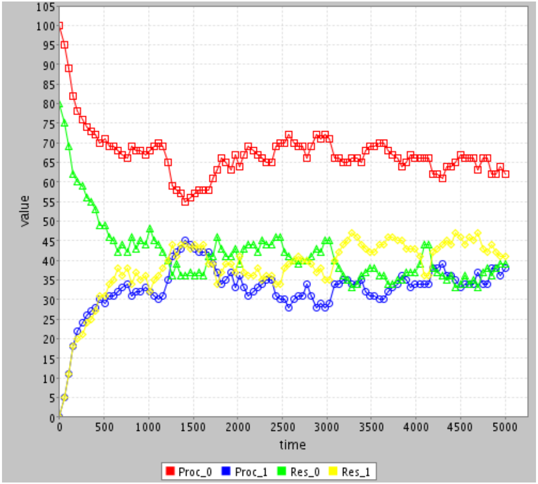
Processors and resources (simulation run A)



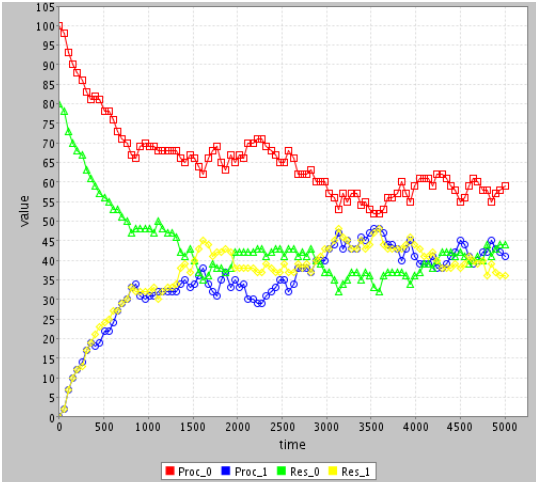
Processors and resources (simulation run B)



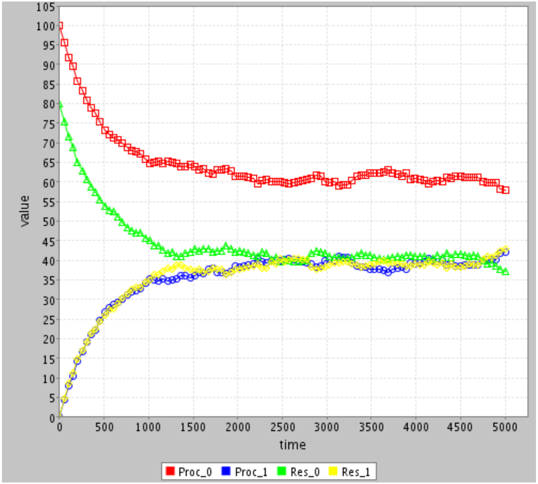
Processors and resources (simulation run C)



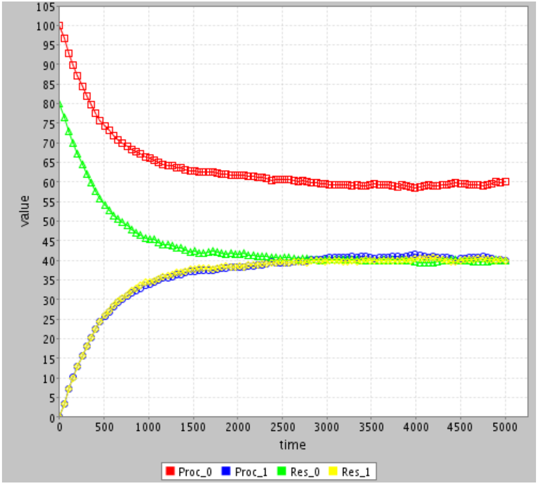
Processors and resources (simulation run D)



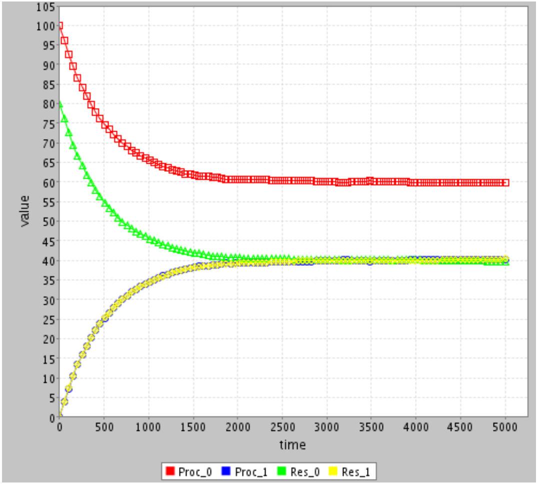
Processors and resources (average of 10 runs)



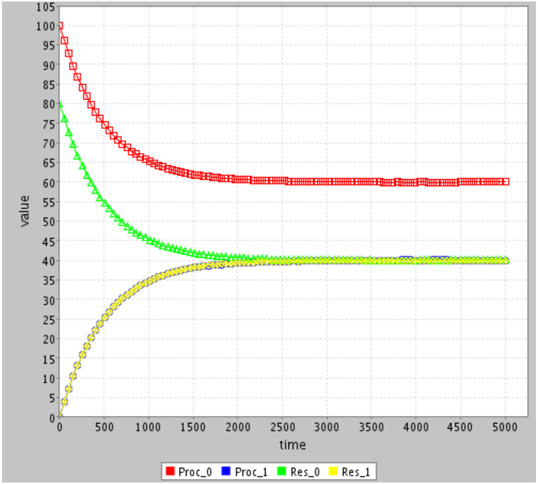
Processors and resources (average of 100 runs)



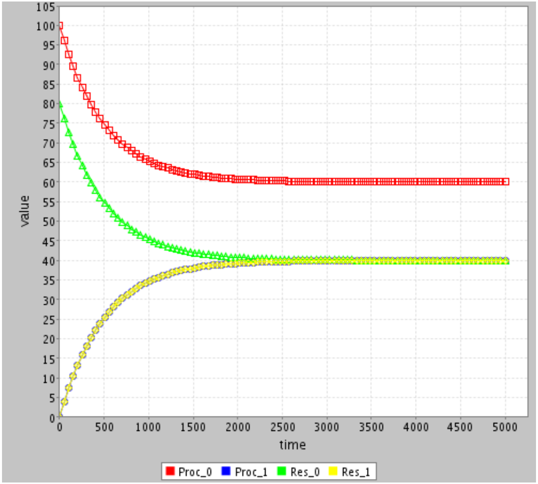
Processors and resources (average of 1000 runs)



Processors and resources (average of 10000 runs)



Processors and resources (ODE solution)



Relation to transient solution

- It is important to understand what this solution represents compared to say, traditional transient analysis of a Markov chain: an ODE represents a deterministic view of a system, that is, a particular mean trajectory.

Relation to transient solution

- It is important to understand what this solution represents compared to say, traditional transient analysis of a Markov chain: an ODE represents a deterministic view of a system, that is, a particular mean trajectory.
- This compares to a transient Markov model solution which maintains the stochastic information in the solution and shows a particular trajectory's probability of occurring at a time t.

Outline

1 Introduction
- Collective Dynamics
- Process Algebra

2 Continuous Approximation
- State variables
- Numerical illustration

3 Examples
- Case Study: Scalable Web Services
- Internet worms

4 On-going and Future Work
- Alternative Models
- New Languages

Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.

Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.
- Sharing of knowledge is promoted by providing students with a wider selection of subjects.

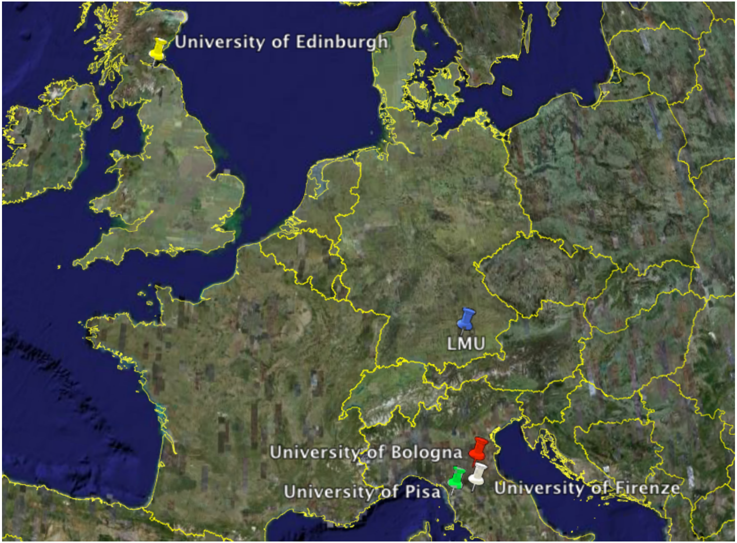
Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.
- Sharing of knowledge is promoted by providing students with a wider selection of subjects.
- Services are replicated across the physical sites.

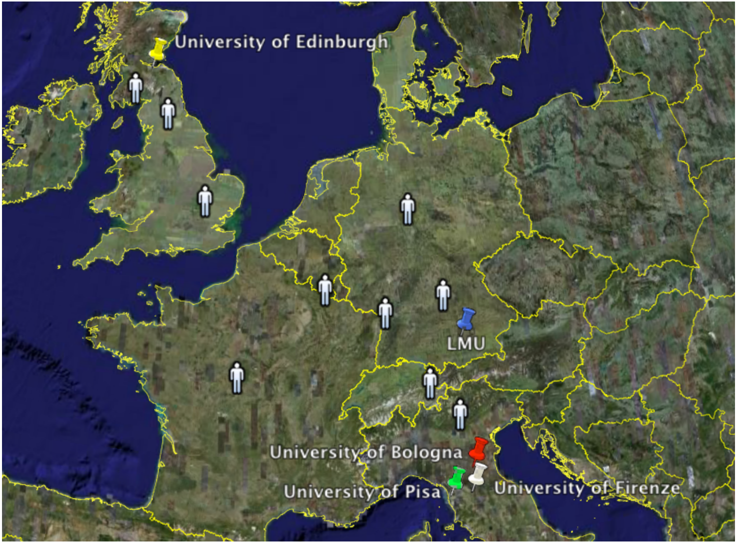
Virtual University Scenario

- A Virtual University is a federation of *real* universities, each contributing courses and degrees.
- Sharing of knowledge is promoted by providing students with a wider selection of subjects.
- Services are replicated across the physical sites.
- By agreement in the university, students may connect to any site to download content and use services, not just the one which is geographically closest.

Case Study: A Virtual University



Location, Time, and Size



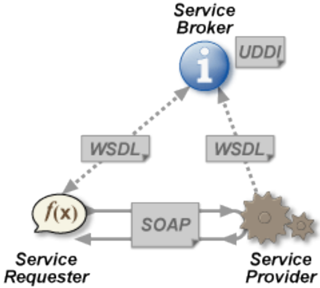
Replicating Web Services

Two viable approaches to cope with increasing user demand:

Replicating Web Services

Two viable approaches to cope with increasing user demand:

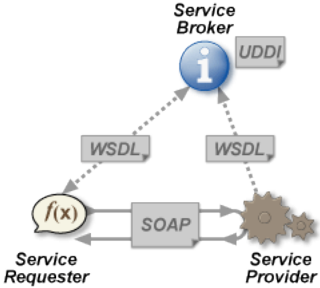
- use a service broker for routing



Replicating Web Services

Two viable approaches to cope with increasing user demand:

- use a service broker for routing

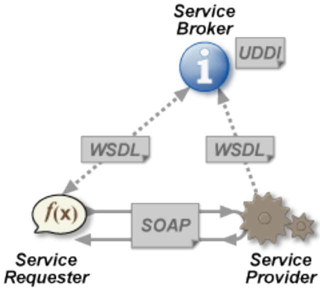


- decentralised routing

Replicating Web Services

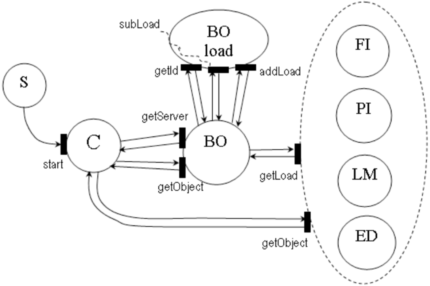
Two viable approaches to cope with increasing user demand:

- use a service broker for routing



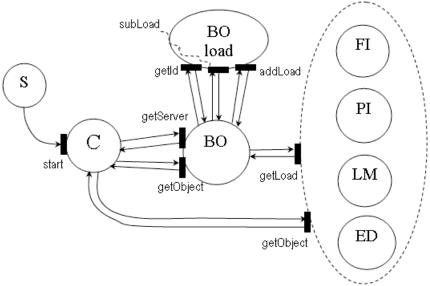
- decentralised routing

Decentralised Routing



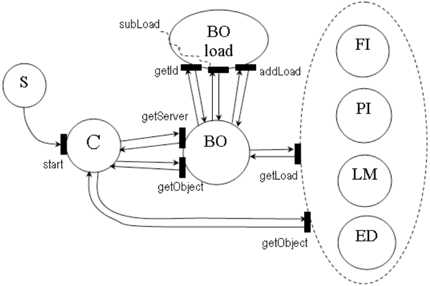
1. A client contacts a university site to download content.

Decentralised Routing



1. A client contacts a university site to download content.
2. The site either serves the request or forwards it to another site.

Decentralised Routing



1. A client contacts a university site to download content.
2. The site either serves the request or forwards it to another site.
3. The decision in made in accord with the local service policy.

Model in PEPA

Clients

$$
\begin{array}{rcl}
Client_i & \overset{def}{=} & (connect_1, c_{1,i}).(download_1, d_{1,i}).Idle_i \\
 & + & (connect_2, c_{2,i}).(download_2, d_{2,i}).Idle_i \\
 & \ldots & \\
 & + & (connect_m, c_{m,i}).(download_m, d_{m,i}).Idle_i \\
 & + & (overload, \top).Client_i \\
Idle_i & \overset{def}{=} & (idle, r_{idle,i}).Client_i
\end{array}
$$

$$(1 \leq i \leq k)$$

Model in PEPA

Clients

$$\begin{array}{rcl}
Client_i & \stackrel{\mathit{def}}{=} & (connect_1, c_{1,i}).(download_1, d_{1,i}).Idle_i \\
 & + & (connect_2, c_{2,i}).(download_2, d_{2,i}).Idle_i \\
 & \ldots & \\
 & + & (connect_m, c_{m,i}).(download_m, d_{m,i}).Idle_i \\
 & + & (overload, \top).Client_i \\
Idle_i & \stackrel{\mathit{def}}{=} & (idle, r_{idle,i}).Client_i
\end{array}$$

$$(1 \leq i \leq k)$$

Model in PEPA

Clients

$$
\begin{array}{rcl}
Client_i & \overset{def}{=} & (connect_1, \textcolor{red}{c_{1,i}}).(download_1, \textcolor{red}{d_{1,i}}).Idle_i \\
 & + & (connect_2, \textcolor{red}{c_{2,i}}).(download_2, \textcolor{red}{d_{2,i}}).Idle_i \\
 & \ldots & \\
 & + & (connect_m, \textcolor{red}{c_{m,i}}).(download_m, \textcolor{red}{d_{m,i}}).Idle_i \\
 & + & (overload, \top).Client_i \\
Idle_i & \overset{def}{=} & (idle, \textcolor{red}{r_{idle,i}}).Client_i
\end{array}
$$

$$(1 \leq i \leq k)$$

Model in PEPA

Content mirrors

$$\begin{aligned} \mathit{Mirror}_j &\overset{\mathit{def}}{=} \left(\mathit{connect}_j, f_j(s)\right).\mathit{MirrorUploading}_j \\ \mathit{MirrorUploading}_j &\overset{\mathit{def}}{=} \left(\mathit{download}_j, \top\right).\mathit{Mirror}_j \end{aligned}$$

$$(1 \leq j \leq m)$$

Model in PEPA

Content mirrors

$$
\begin{aligned}
Mirror_j &\overset{def}{=} (connect_j, f_j(s)).MirrorUploading_j \\
MirrorUploading_j &\overset{def}{=} (download_j, \top).Mirror_j
\end{aligned}
$$

$$(1 \leq j \leq m)$$

Service policies as functional rates in PEPA

The Bologna policy

Serve all requests while load is less than 75%. If more, and the loads at UNIFI, UPISA, LMU and UEDIN are at least 60%, 60%, 40% and 20% then serve the request if load is less than 95%.

Service policies as functional rates in PEPA

The Bologna policy

Serve all requests while load is less than 75%. If more, and the loads at UNIFI, UPISA, LMU and UEDIN are at least 60%, 60%, 40% and 20% then serve the request if load is less than 95%.

$$f_{\mathrm{UNIBO}} = \begin{cases} \top & \textit{if MirrorUploading}_{\mathrm{UNIBO}} < 75 \\ \top & \textit{if MirrorUploading}_{\mathrm{UNIBO}} < 95, \\ & \quad \textit{MirrorUploading}_{\mathrm{UNIFI}} \geq 60, \\ & \quad\quad \textit{MirrorUploading}_{\mathrm{UPISA}} \geq 60, \\ & \quad\quad\quad \textit{MirrorUploading}_{\mathrm{LMU}} \geq 40, \\ & \quad\quad\quad\quad \textit{MirrorUploading}_{\mathrm{UEDIN}} \geq 20 \\ 0 & \textit{otherwise} \end{cases}$$

Model in PEPA

Dealing with overload

$$Overload \overset{def}{=} (overload, o(s)).Overload$$

$$o(s) = \begin{cases} \top & f_i(s) = 0, \quad 1 \leq i \leq m \\ 0 & \text{otherwise} \end{cases}$$

Model in PEPA

Dealing with overload

$$Overload \overset{\mathit{def}}{=} (overload, o(s)).Overload$$

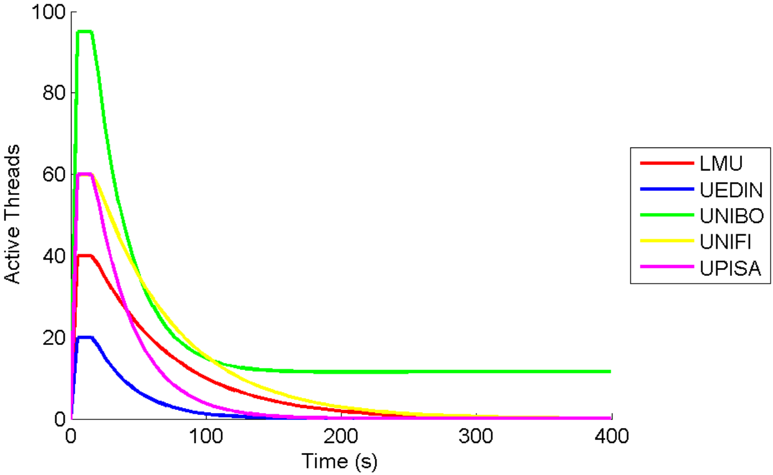
$$o(s) = \begin{cases} \top & f_i(s) = 0, \quad 1 \leq i \leq m \\ 0 & \text{otherwise} \end{cases}$$

The system as a whole with client and mirror site populations

$$\left(Client_1[p_1] \parallel Client_2[p_2] \parallel \ldots \parallel Client_k[p_k]\right) \underset{L}{\bowtie} \left(Mirror_1[q_1] \parallel Mirror_2[q_2] \parallel \ldots \parallel Mirror_m[q_m]\right)$$

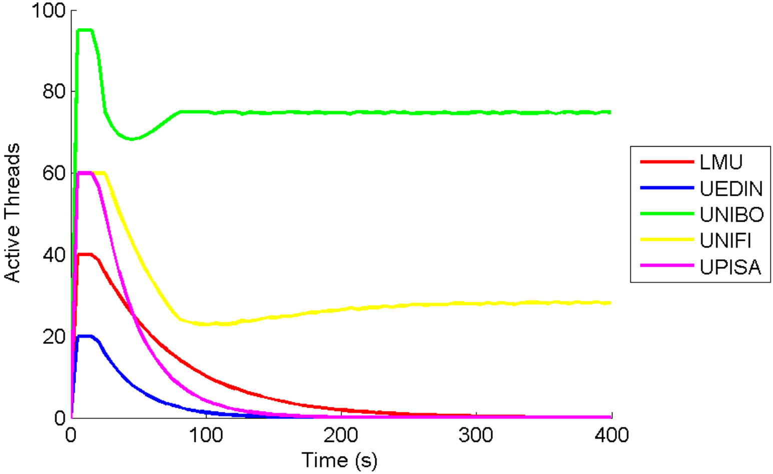
Numerical Results

$r_{idle} = 0.001$



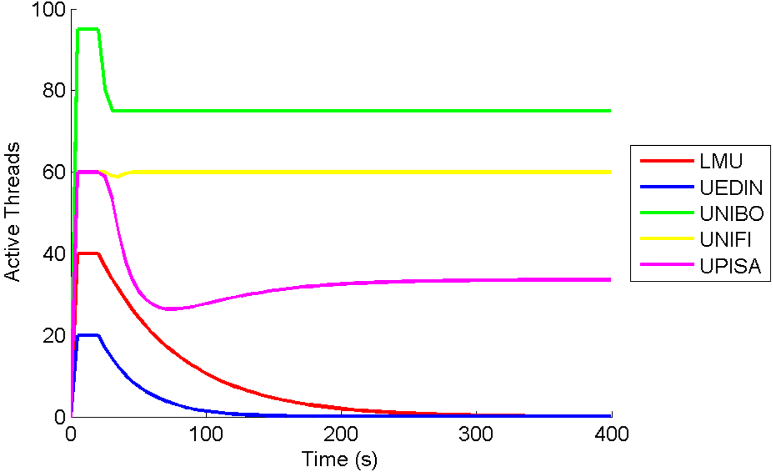
Numerical Results

$r_{idle} = 0.01$



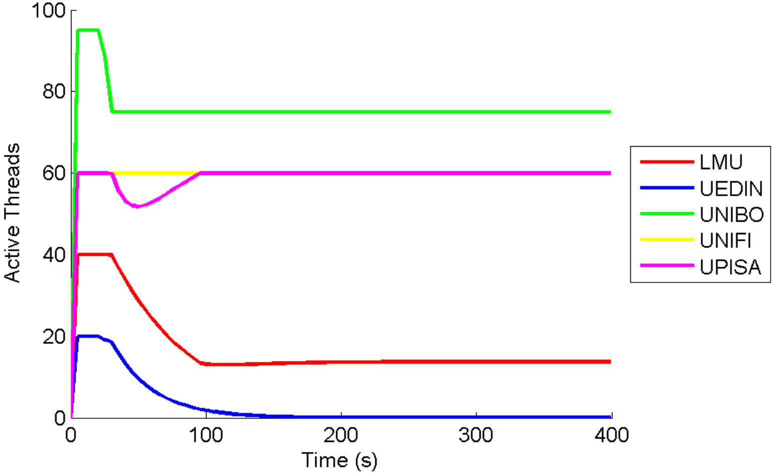
Numerical Results

$r_{idle} = 0.02$



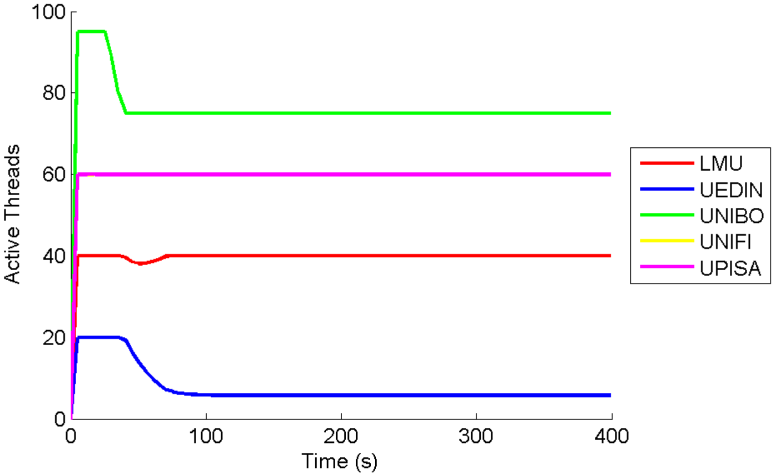
Numerical Results

$r_{idle} = 0.03$



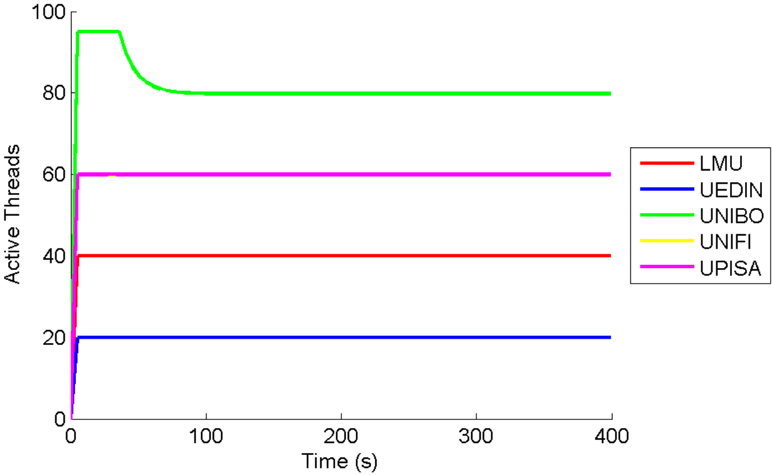
Numerical Results

$r_{idle} = 0.04$



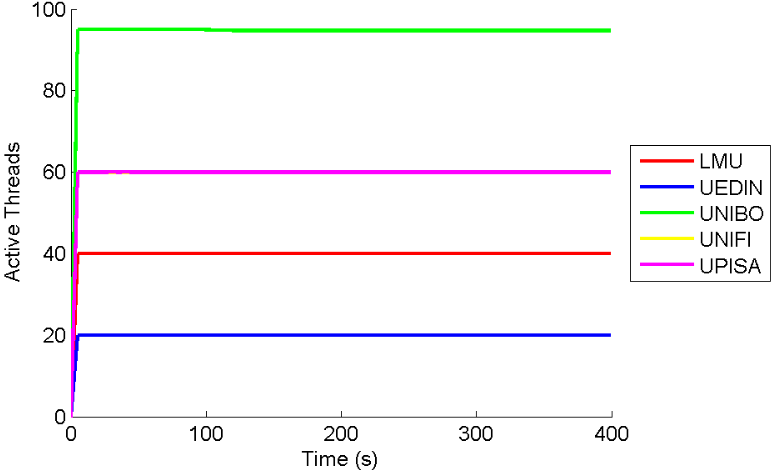
Numerical Results

$r_{idle} = 0.05$



Numerical Results

$r_{idle} = 0.06$



Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.

Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.
- PEPA gives us insights into the performance of the system.

Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.
- PEPA gives us insights into the performance of the system.
    - We applied the continuous-space semantics of PEPA and were able to see service policies at work.

Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.
- PEPA gives us insights into the performance of the system.
    - We applied the continuous-space semantics of PEPA and were able to see service policies at work.
    - Analysis carried out on a system of 17 ODEs as opposed to an underlying CTMC of over 270 million states.

Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.
- PEPA gives us insights into the performance of the system.
    - We applied the continuous-space semantics of PEPA and were able to see service policies at work.
    - Analysis carried out on a system of 17 ODEs as opposed to an underlying CTMC of over 270 million states.
- This framework is general. Elements that may be changed are

Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.
- PEPA gives us insights into the performance of the system.
    - We applied the continuous-space semantics of PEPA and were able to see service policies at work.
    - Analysis carried out on a system of 17 ODEs as opposed to an underlying CTMC of over 270 million states.
- This framework is general. Elements that may be changed are
    - Classes of clients

Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.
- PEPA gives us insights into the performance of the system.
  - We applied the continuous-space semantics of PEPA and were able to see service policies at work.
  - Analysis carried out on a system of 17 ODEs as opposed to an underlying CTMC of over 270 million states.
- This framework is general. Elements that may be changed are
  - Classes of clients
  - Deployment of service providers

Comments

- We have a modelling approach which captures both functional and non-functional properties of large-scale systems.
- PEPA gives us insights into the performance of the system.
    - We applied the continuous-space semantics of PEPA and were able to see service policies at work.
    - Analysis carried out on a system of 17 ODEs as opposed to an underlying CTMC of over 270 million states.
- This framework is general. Elements that may be changed are
    - Classes of clients
    - Deployment of service providers
    - Different load balancing policies

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.
- Far more destructive is the worms' effect on the Internet routing infrastructure, as the worms tend to overload the connecting routers with nonexistent IP lookups.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.
- Far more destructive is the worms' effect on the Internet routing infrastructure, as the worms tend to overload the connecting routers with nonexistent IP lookups.
- Worms like Nimbda, Slammer, Code Red, Sasser and Code Red 2 have caused the Internet to become unusable for many hours at a time until security patches could be applied and routers fixed.

Internet worms: Background

- Internet worms are malicious programs that exploit operating system security weaknesses to propagate themselves.
- While the security flaws go unpatched, the worm spreads epidemic-like and uses large amounts of available bandwidth.
- Far more destructive is the worms' effect on the Internet routing infrastructure, as the worms tend to overload the connecting routers with nonexistent IP lookups.
- Worms like Nimbda, Slammer, Code Red, Sasser and Code Red 2 have caused the Internet to become unusable for many hours at a time until security patches could be applied and routers fixed.
- The estimated cost of computer worms and related activities is about $50 billion a year.

An Internet-scale Problem

- We wish to study the emergent behaviour of Internet worms as they spread to thousands and then hundreds-of-thousands of hosts.

An Internet-scale Problem

- We wish to study the emergent behaviour of Internet worms as they spread to thousands and then hundreds-of-thousands of hosts.
- Explicit state-based methods for calculating steady-state, transient or passage-time measures are limited to state-spaces of the order of 10^9.

An Internet-scale Problem

- We wish to study the emergent behaviour of Internet worms as they spread to thousands and then hundreds-of-thousands of hosts.
- Explicit state-based methods for calculating steady-state, transient or passage-time measures are limited to state-spaces of the order of 10^9.
- By transforming our stochastic process algebra model into a set of ODEs, we can obtain a plot of model behaviour against time for models with global state spaces in excess of 10^{10000} states.

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

$$\frac{\mathrm{d}s(t)}{\mathrm{d}t} = -\beta\, s(t)\, i(t)$$

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

$$\begin{aligned}\frac{\mathrm{d}s(t)}{\mathrm{d}t} &= -\beta\, s(t)\, i(t) \\ \frac{\mathrm{d}i(t)}{\mathrm{d}t} &= \beta\, s(t)\, i(t) - \gamma\, i(t)\end{aligned}$$

Susceptible-Infective-Removed (SIR) model

- We apply a version of an SIR model of infection to various computer worm attack models.
- An SIR model explicitly represents the total number of susceptible, infective and removed hosts in a system and is more commonly used to model disease epidemics.

$$
\begin{aligned}
\frac{\mathrm{d}s(t)}{\mathrm{d}t} &= -\beta\, s(t)\, i(t) \\
\frac{\mathrm{d}i(t)}{\mathrm{d}t} &= \beta\, s(t)\, i(t) - \gamma\, i(t) \\
\frac{\mathrm{d}r(t)}{\mathrm{d}t} &= \gamma\, i(t)
\end{aligned}
$$

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.
- As the system evolves more susceptible computers become infected from the growing infective population.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.
- As the system evolves more susceptible computers become infected from the growing infective population.
- An infected computer can be patched so that it is no longer infected or susceptible to infection.

Susceptible-Infective-Removed over a network

- This is our most basic infection model and is used to verify that we get recognisable qualitative results.
- Initially, there are N susceptible computers and one infected computer.
- As the system evolves more susceptible computers become infected from the growing infective population.
- An infected computer can be patched so that it is no longer infected or susceptible to infection.
- This state is termed removed and is an absorbing state for that component in the system.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.
- Additionally, an attempted network connection can fail or timeout as indicated by the *fail* action.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.
- Additionally, an attempted network connection can fail or timeout as indicated by the *fail* action.
- This might be due to network contention or the lack of availability of a susceptible machine to infect.

Susceptible-Infective-Removed over a network

- The capacity of the network is dictated by the parameter M, the number of concurrent, independent connections that the network can sustain.
- Additionally, an attempted network connection can fail or timeout as indicated by the *fail* action.
- This might be due to network contention or the lack of availability of a susceptible machine to infect.
- As large scale worm infections tend not to waste time determining whether a given host is already infected or not, we assume that a certain number of infections will attempt to reinfect hosts; in this instance, the host is unaffected.

Susceptible-Infective-Removed over a network

$$
\begin{aligned}
S &\overset{def}{=} (infectS, \top).I \\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R \\
R &\overset{def}{=} Stop
\end{aligned}
$$

Susceptible-Infective-Removed over a network

$$
\begin{aligned}
S &\overset{def}{=} (infectS, \top).I \\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R \\
R &\overset{def}{=} Stop
\end{aligned}
$$

$$
\begin{aligned}
Net &\overset{def}{=} (infectI, \top).Net' \\
Net' &\overset{def}{=} (infectS, \beta).Net + (fail, \delta).Net
\end{aligned}
$$

Susceptible-Infective-Removed over a network

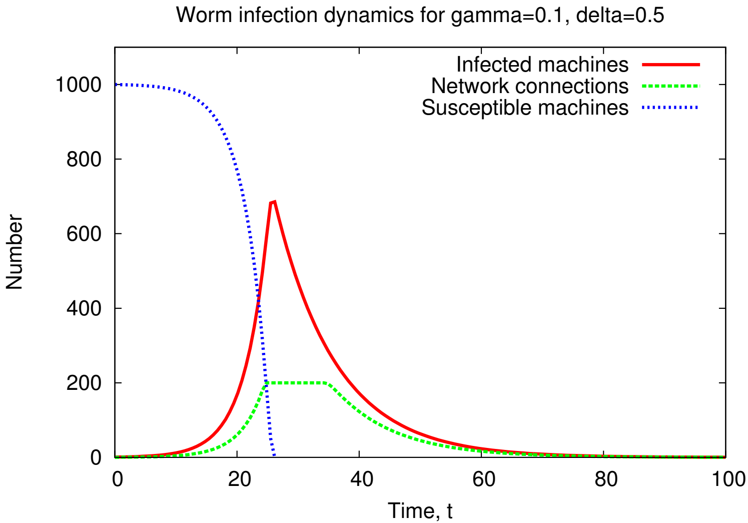
$$\begin{aligned} S &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \top).I \\ I &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\ R &\stackrel{\mathit{def}}{=} \mathit{Stop} \end{aligned}$$

$$\begin{aligned} \mathit{Net} &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' \\ \mathit{Net}' &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net} \end{aligned}$$

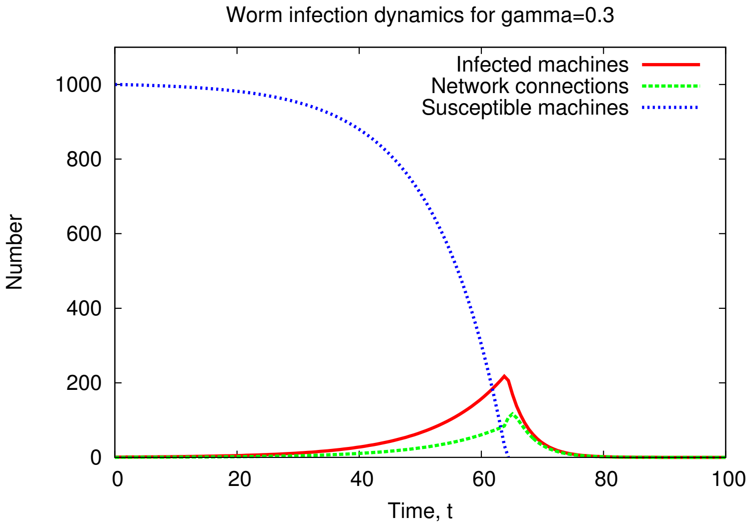
$$\mathit{Sys} \stackrel{\mathit{def}}{=} (S[N] \parallel I) \underset{L}{\bowtie} \mathit{Net}[M]$$

$$\text{where } L = \{\, \mathit{infectI}, \mathit{infectS} \,\}$$

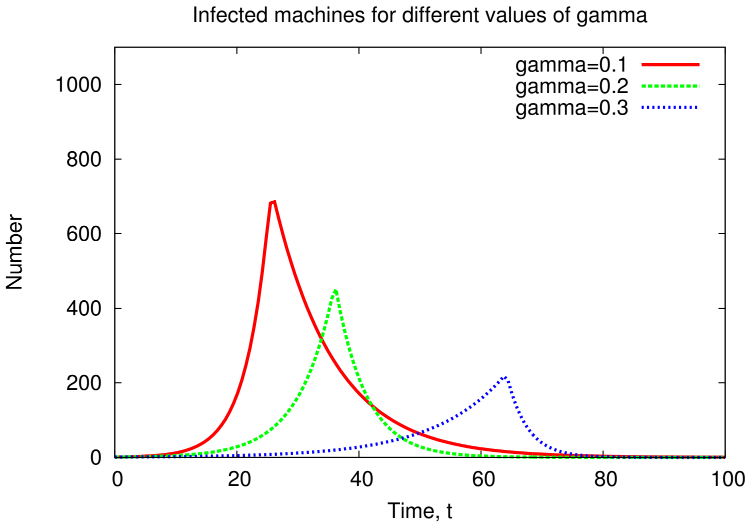
Patch rate $\gamma = 0.1$. Connection failure rate $\delta = 0.5$



Patch rate $\gamma = 0.3$. Connection failure rate $\delta = 0.5$



Increasing machine patch rate γ from 0.1 to 0.3



Susceptible-Infective-Removed-Reinfection (SIRR) model

- As with the SIR model, we constrain infection to occur over a limited network resource, constrained by the number of independent network connections in the system, M.

Susceptible-Infective-Removed-Reinfection (SIRR) model

- As with the SIR model, we constrain infection to occur over a limited network resource, constrained by the number of independent network connections in the system, M.
- A small modification in the process model of infection allows for removed computers to become susceptible again after a delay.

Susceptible-Infective-Removed-Reinfection (SIRR) model

- As with the SIR model, we constrain infection to occur over a limited network resource, constrained by the number of independent network connections in the system, M.
- A small modification in the process model of infection allows for removed computers to become susceptible again after a delay.
- We use this to model a faulty or incomplete security upgrade or the mistaken removal of security patches which had previously defended the machine against attack.

Susceptible-Infective-Removed-Reinfection (SIRR) model

$$
\begin{aligned}
S &\overset{def}{=} (infectS, \top).I \\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R \\
R &\overset{def}{=} (unsecure, \mu).S
\end{aligned}
$$

Susceptible-Infective-Removed-Reinfection (SIRR) model

$$
\begin{aligned}
S &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \top).I \\
I &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \beta).I + (\mathit{infectS}, \top).I + (\mathit{patch}, \gamma).R \\
R &\stackrel{\mathit{def}}{=} (\mathit{unsecure}, \mu).S
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Net} &\stackrel{\mathit{def}}{=} (\mathit{infectI}, \top).\mathit{Net}' \\
\mathit{Net}' &\stackrel{\mathit{def}}{=} (\mathit{infectS}, \beta).\mathit{Net} + (\mathit{fail}, \delta).\mathit{Net}
\end{aligned}
$$

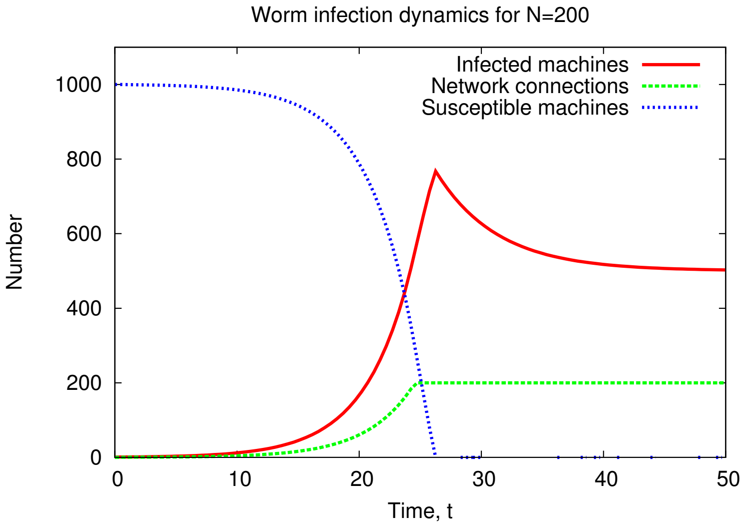
Susceptible-Infective-Removed-Reinfection (SIRR) model

$$
\begin{aligned}
S &\overset{def}{=} (infectS, \top).I \\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R \\
R &\overset{def}{=} (unsecure, \mu).S
\end{aligned}
$$

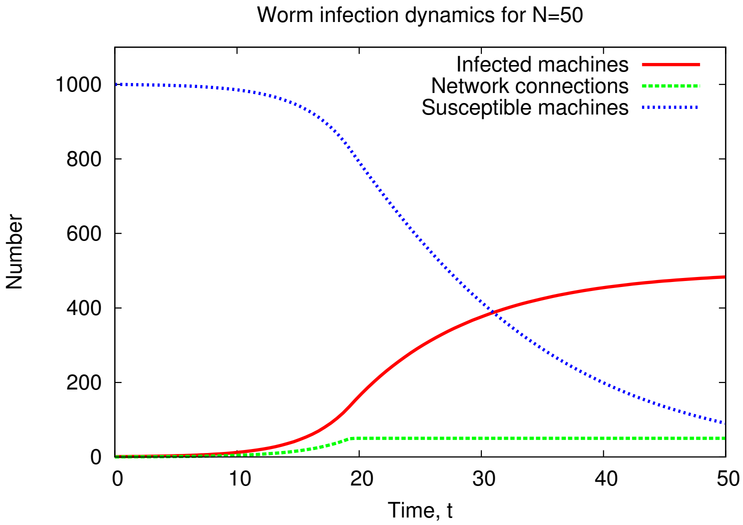
$$
\begin{aligned}
Net &\overset{def}{=} (infectI, \top).Net' \\
Net' &\overset{def}{=} (infectS, \beta).Net + (fail, \delta).Net
\end{aligned}
$$

$$
\begin{aligned}
Sys &\overset{def}{=} (S[1000] \parallel I) \underset{L}{\bowtie} Net[M] \\
& \qquad \text{where } L = \{infectI, infectS\}
\end{aligned}
$$

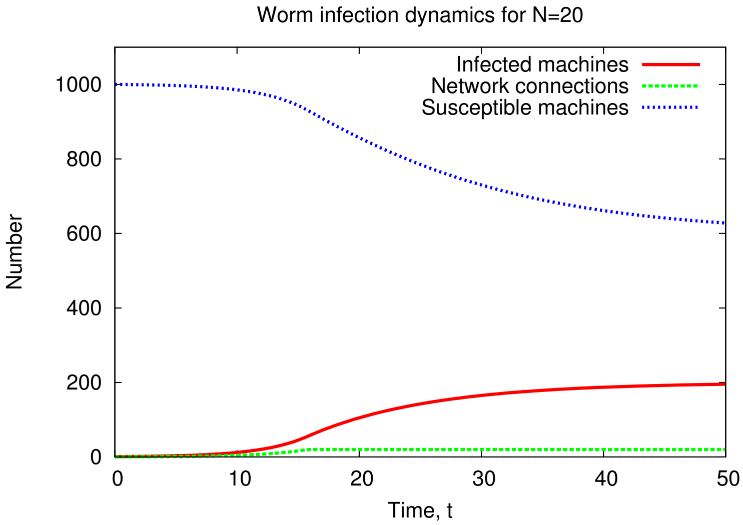
Unsecured SIR model (200 network channels)



Unsecured SIR model (50 network channels)



Unsecured SIR model (20 network channels)



Susceptible-Infective-Removed-Attack (SIR-Attack) model

- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.
- In some worms it is known that there is a bimodal behaviour to the worm, either a worm can infect another computer or it can start an attack on a victim computer.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.
- In some worms it is known that there is a bimodal behaviour to the worm, either a worm can infect another computer or it can start an attack on a victim computer.
- The attack need not itself exploit any particular security flaw, but can be something as simple as requesting a specific web page, or issuing a *ping* request.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

- This example describes a modified SIR-Attack model. This simulates a possible *distributed denial-of-service* (DDOS) attack mode of an Internet worm.
- In some worms it is known that there is a bimodal behaviour to the worm, either a worm can infect another computer or it can start an attack on a victim computer.
- The attack need not itself exploit any particular security flaw, but can be something as simple as requesting a specific web page, or issuing a *ping* request.
- The combination of perhaps millions of machines making such requests quickly overwhelms the target computer, which either crashes under the huge load, or becomes unusably slow.

Susceptible-Infective-Removed-Attack (SIR-Attack) model

$$
\begin{aligned}
S &\overset{def}{=} (infectS, \top).I \\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R + (attack, \chi).A \\
A &\overset{def}{=} (attackA, \lambda).A + (patch, \gamma).R \\
R &\overset{def}{=} Stop
\end{aligned}
$$

Susceptible-Infective-Removed-Attack (SIR-Attack) model

$$
\begin{aligned}
S &\overset{def}{=} (infectS, \top).I \\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R + (attack, \chi).A \\
A &\overset{def}{=} (attackA, \lambda).A + (patch, \gamma).R \\
R &\overset{def}{=} Stop
\end{aligned}
$$

$$
\begin{aligned}
Net &\overset{def}{=} (infectI, \top).Net' + (attackA, \top).Net'' \\
Net' &\overset{def}{=} (infectS, \beta).Net + (fail, \delta).Net \\
Net'' &\overset{def}{=} (attackV, \rho).Net + (fail, \delta).Net
\end{aligned}
$$

Susceptible-Infective-Removed-Attack (SIR-Attack) model

$$\begin{aligned}
S &\overset{def}{=} (infectS, \top).I \\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R + (attack, \chi).A \\
A &\overset{def}{=} (attackA, \lambda).A + (patch, \gamma).R \\
R &\overset{def}{=} Stop
\end{aligned}$$

$$\begin{aligned}
Net &\overset{def}{=} (infectI, \top).Net' + (attackA, \top).Net'' \\
Net' &\overset{def}{=} (infectS, \beta).Net + (fail, \delta).Net \\
Net'' &\overset{def}{=} (attackV, \rho).Net + (fail, \delta).Net
\end{aligned}$$

$$\begin{aligned}
V &\overset{def}{=} (attackV, \top).V' \\
V' &\overset{def}{=} (release, \sigma).V
\end{aligned}$$

Susceptible-Infective-Removed-Attack (SIR-Attack) model

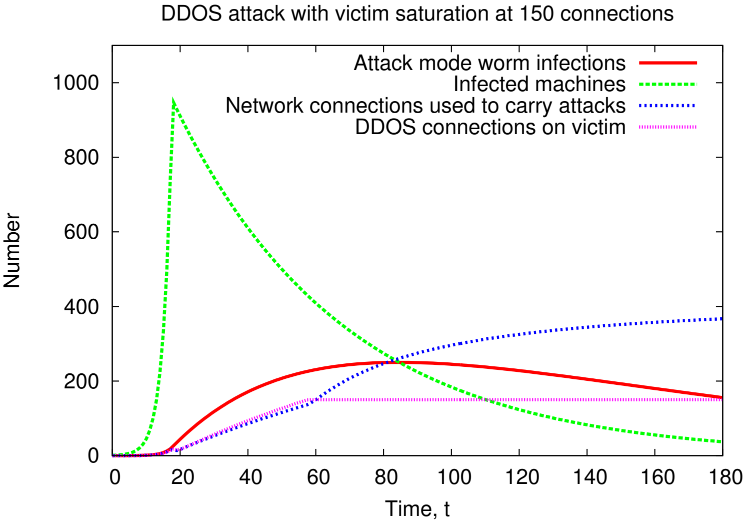
$$
\begin{aligned}
S &\overset{def}{=} (infectS, \top).I\\
I &\overset{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R + (attack, \chi).A\\
A &\overset{def}{=} (attackA, \lambda).A + (patch, \gamma).R\\
R &\overset{def}{=} Stop
\end{aligned}
$$

$$
\begin{aligned}
Net &\overset{def}{=} (infectI, \top).Net' + (attackA, \top).Net''\\
Net' &\overset{def}{=} (infectS, \beta).Net + (fail, \delta).Net\\
Net'' &\overset{def}{=} (attackV, \rho).Net + (fail, \delta).Net
\end{aligned}
$$

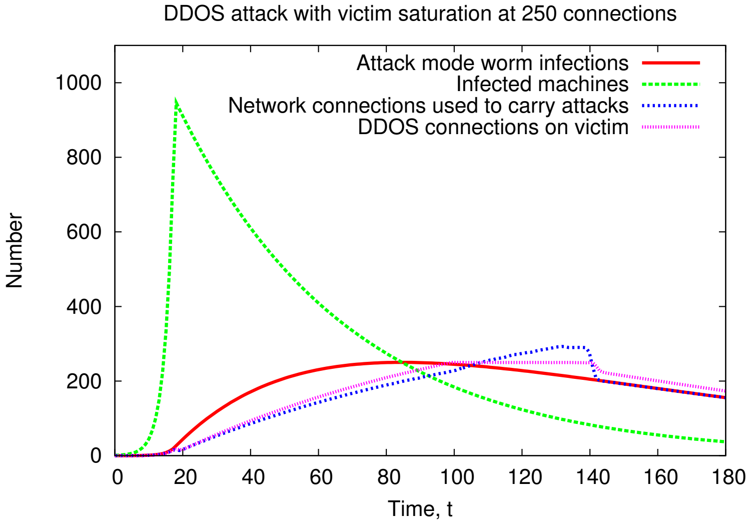
$$
\begin{aligned}
V &\overset{def}{=} (attackV, \top).V'\\
V' &\overset{def}{=} (release, \sigma).V
\end{aligned}
$$

$$
\begin{aligned}
Sys \overset{def}{=} (S[1000] \parallel I \parallel V) \underset{L}{\bowtie} Net[M]\\
\text{where } L = \{infectI, infectS, attackA, attackV\}
\end{aligned}
$$

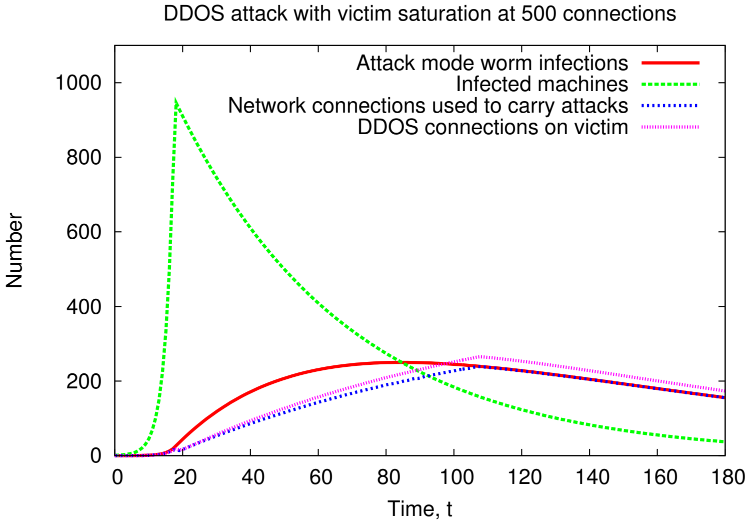
Impact of the network capacity on the DDOS attack



Impact of the network capacity on the DDOS attack



Impact of the network capacity on the DDOS attack



Conclusions

- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.

Conclusions

- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.
- Process algebra modelling allows the details of interactions to be recorded on the individual level but then mapped to abstracted away into appropriate population-based representations.

Conclusions

- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.
- Process algebra modelling allows the details of interactions to be recorded on the individual level but then mapped to abstracted away into appropriate population-based representations.
- The scale of problems which can be modelled in this way vastly exceeds those which are founded on explicit state representations.

Conclusions

- The scale of the effects of Internet worms defeats attempts to model their behaviour in very close detail, and thus impedes the analysis which has the potential to bring understanding of their function and distribution.
- Process algebra modelling allows the details of interactions to be recorded on the individual level but then mapped to abstracted away into appropriate population-based representations.
- The scale of problems which can be modelled in this way vastly exceeds those which are founded on explicit state representations.
- We believe the modelling methods exemplified here to be generally useful for analysing the behaviour of populations of interacting processes with complex dynamics.

Outline

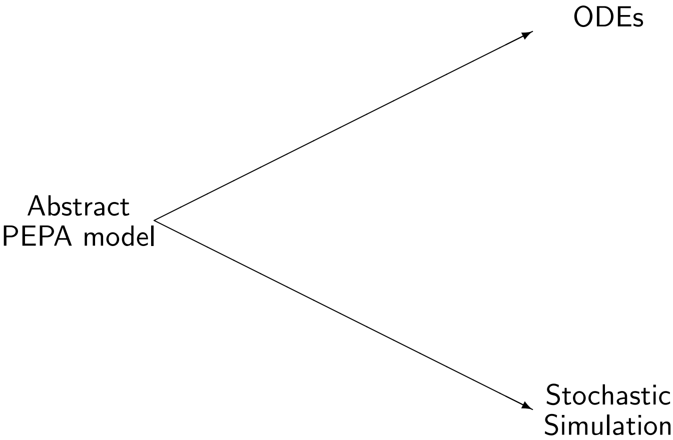
1 Introduction
- Collective Dynamics
- Process Algebra

2 Continuous Approximation
- State variables
- Numerical illustration

3 Examples
- Case Study: Scalable Web Services
- Internet worms

4 On-going and Future Work
- Alternative Models
- New Languages

Alternative Representations

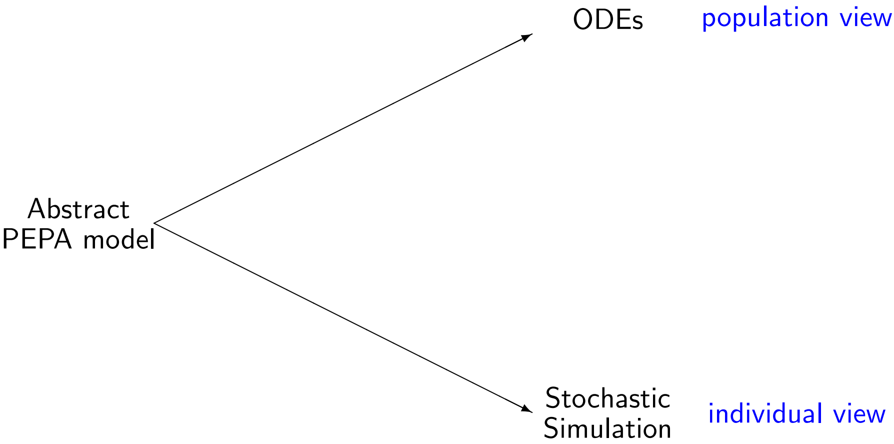


Abstract PEPA model

ODEs

Stochastic Simulation

Alternative Representations



Abstract PEPA model

ODEs

population view

Stochastic Simulation

individual view

Discretisation

In some cases observations of the system are made not at the level of the individuals but at the level of the populations.

Discretisation

In some cases observations of the system are made not at the level of the individuals but at the level of the populations.

This more naturally lends itself to description in ordinary differential equations (ODEs).

Discretisation

In some cases observations of the system are made not at the level of the individuals but at the level of the populations.

This more naturally lends itself to description in ordinary differential equations (ODEs).

Luckily the process algebra descriptions are the same.

Discretisation

In some cases observations of the system are made not at the level of the individuals but at the level of the populations.

This more naturally lends itself to description in ordinary differential equations (ODEs).

Luckily the process algebra descriptions are the same.

However we may be interested in analyses such as model checking which cannot be readily applied in the context of ODEs.

Discretisation

In some cases observations of the system are made not at the level of the individuals but at the level of the populations.

This more naturally lends itself to description in ordinary differential equations (ODEs).

Luckily the process algebra descriptions are the same.

However we may be interested in analyses such as model checking which cannot be readily applied in the context of ODEs.

This can be accomodated if we consider an abstract, intermediate level model in which transitions represent discretised steps within the continuous population range.

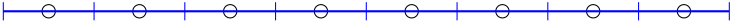
Discretising continuous variables

We can discretise the continuous range of possible concentration values into a number of distinct states. These form the possible states of the component representing the entity.

Discretising continuous variables

We can discretise the continuous range of possible concentration values into a number of distinct states. These form the possible states of the component representing the entity.

Discretising continuous variables

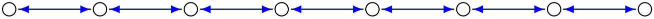


We can discretise the continuous range of possible concentration values into a number of distinct states. These form the possible states of the component representing the entity.

Discretising continuous variables

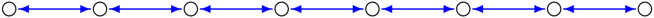
We can discretise the continuous range of possible concentration values into a number of distinct states. These form the possible states of the component representing the entity.

Discretising continuous variables



We can discretise the continuous range of possible concentration values into a number of distinct states. These form the possible states of the component representing the entity.

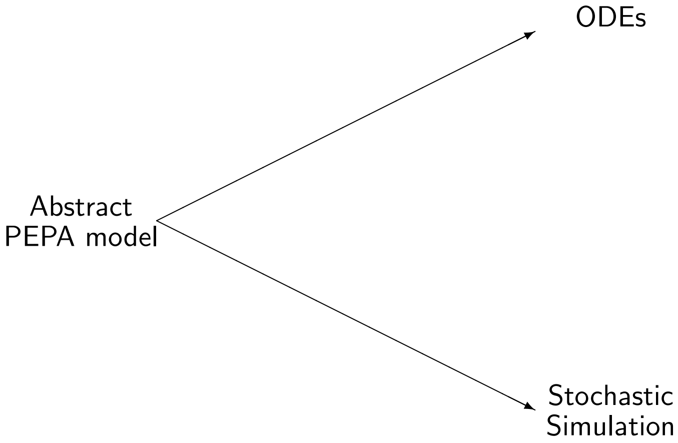
Discretising continuous variables



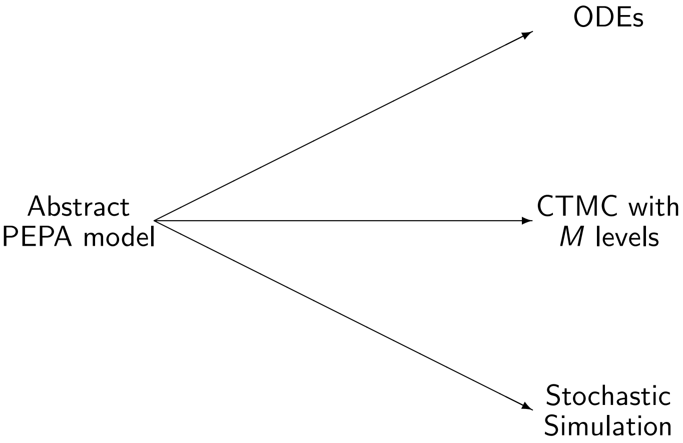
We can discretise the continuous range of possible concentration values into a number of distinct states. These form the possible states of the component representing the entity.

Or alternatively each copy of an entity might represent a range of concentration values.

Alternative Representations

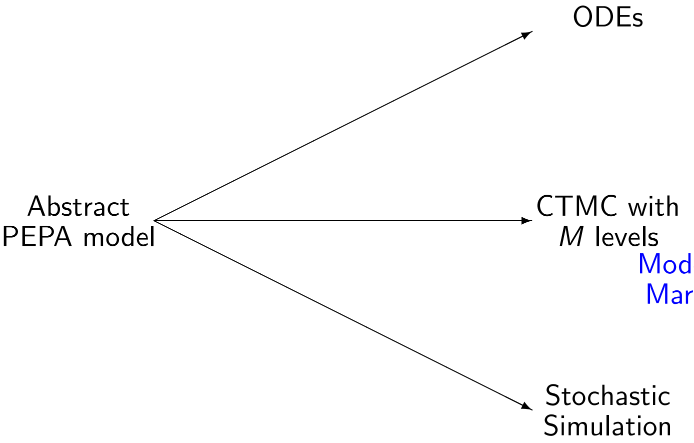


Alternative Representations



abstract view

Alternative Representations



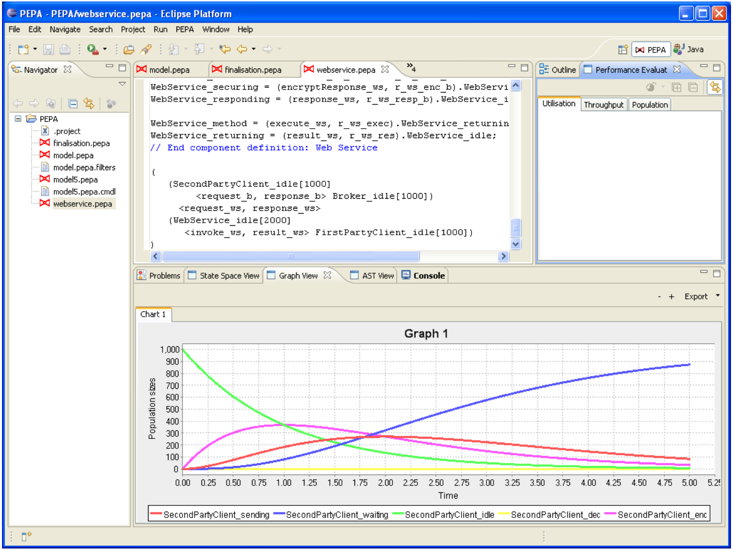
Model checking and Markovian analysis

Software Tool Support

PEPA is supported by several tool suites:

- The PRISM probabilistic model checker
  - Kwiatkowska, Norman, Parker – Oxford
- The IPC/Hydra tool chain
  - Bradley, Knottenbelt, Dingle – Imperial College, London
- The Eclipse Plug-in for PEPA
  - Tribastone, Clark and Duguid – Edinburgh

Eclipse Plug-in for PEPA



```
WebService_securing = (encryptResponse_ws, r_ws_enc_b).WebServi
WebService_responding = (response_ws, r_ws_resp_b).WebService_i

WebService_method = (execute_ws, r_ws_exec).WebService_returnin
WebService_returning = (result_ws, r_ws_res).WebService_idle;
// End component definition: Web Service

(
   (SecondPartyClient_idle[1000]
        <request_b, response_b> Broker_idle[1000])
     <request_ws, response_ws>
   (WebService_idle[2000]
      <invoke_ws, result_ws> FirstPartyClient_idle[1000])
)
```

New languages

In addition to using PEPA in this novel way we are also exploring the use of new process algebras, specifically designed with collective dynamics in mind:

Bio-PEPA: Designed for modelling biochemical processes such as those that arise in signal transduction pathways.

HYPE: Designed for modelling hybrid systems combining discrete and continuous behaviour.

$\mathcal{L}$: Taking an alternative approach to modelling systems from the population perspective.

Bio-PEPA

- Particularly suited to modelling biochemical processes Bio-PEPA is designed for abstract modelling with parameters capturing state changes in terms of abstract levels.

Bio-PEPA

- Particularly suited to modelling biochemical processes Bio-PEPA is designed for abstract modelling with parameters capturing state changes in terms of abstract levels.
- Moreover the notion of stoichiometry is fully supported: different participants in a reaction (action) may be modified to different degrees.

HYPE

- The continuous approximation approach currently assumes that it is valid to consider populations of all component types so that all actions can be modelled by ODEs.

HYPE

- The continuous approximation approach currently assumes that it is valid to consider populations of all component types so that all actions can be modelled by ODEs.
- HYPE is designed to describe hybrid systems in which there are two types of actions: instantaneous, discrete events and continuously acting influences.

$\mathcal{L}$

- Targetted to population level models, $\mathcal{L}$ takes a dual view of systems with components representing processes or flows and actions representing the aspects of the system which modify flows.

$\mathcal{L}$

- Targetted to population level models, $\mathcal{L}$ takes a dual view of systems with components representing processes or flows and actions representing the aspects of the system which modify flows.
- This offers an alternative formalisation of ODE-based models, the potential of which is currently being explored.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.

Conclusions

- Many interesting and important systems can be regarded as examples of collective dynamics and emergent behaviour.
- Process algebras, such as PEPA, are well-suited to modelling the behaviour of such systems in terms of the individuals and their interactions.
- Continuous approximation allows a rigorous mathematical analysis of the average behaviour of such systems.
- This alternative view of systems has opened up many and exciting new research directions.

Thanks!

Thanks!

Acknowledgements: collaborators

Thanks to many co-authors and collaborators: Jeremy Bradley, Muffy Calder, Federica Ciocchetta, Allan Clark, Adam Duguid, Vashti Galpin, Stephen Gilmore, Marco Stenico, Mirco Tribastone, and others.

Thanks!

Acknowledgements: collaborators

Thanks to many co-authors and collaborators: Jeremy Bradley, Muffy Calder, Federica Ciocchetta, Allan Clark, Adam Duguid, Vashti Galpin, Stephen Gilmore, Marco Stenico, Mirco Tribastone, and others.

Acknowledgements: funding

Thanks to EPRSC for the Process Algebra for Collective Dynamics grant.

Thanks!

Acknowledgements: collaborators
Thanks to many co-authors and collaborators: Jeremy Bradley, Muffy Calder, Federica Ciocchetta, Allan Clark, Adam Duguid, Vashti Galpin, Stephen Gilmore, Marco Stenico, Mirco Tribastone, and others.

Acknowledgements: funding
Thanks to EPRSC for the Process Algebra for Collective Dynamics grant.

More information:

http://www.dcs.ed.ac.uk/pepa