The Tao of PEPA Nets

Stephen Gilmore

Joint work with

Jane Hillston, Leïla Kloul and Marina Ribaudo

November 21, 2005
Background

Modern enterprise software design assumes a heterogeneous distributed system model where mobile objects are sent from host to host.
Background

Modern enterprise software design assumes a heterogeneous distributed system model where mobile objects are sent from host to host.

Systems are usually structured in this way to separate trusted participants with access rights from untrusted participants without access rights but also to achieve required levels of performance.
Background

Modern enterprise software design assumes a heterogeneous distributed system model where mobile objects are sent from host to host. Systems are usually structured in this way to separate trusted participants with access rights from untrusted participants without access rights but also to achieve required levels of performance.

Several formalisms exist to analyse security in mobile code systems (Secure Ambients, Spi-calculus) but what about the performance analysis of such systems?
PEPA nets

The **PEPA nets** language is a high-level modelling formalism for performance analysis of mobile object systems.
PEPA nets

The **PEPA nets** language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a **stochastic Petri net with coloured tokens**. The tokens represent mobile objects with state and behaviour.
PEPA nets

The **PEPA nets** language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston’s Performance Evaluation Process Algebra (PEPA).
PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston’s Performance Evaluation Process Algebra (PEPA).

\[ P ::= (\alpha, r).P \]

prefix
**PEPA nets**

The **PEPA nets** language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a **stochastic** Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a **stochastic process algebra**, Jane Hillston’s Performance Evaluation Process Algebra (PEPA).

\[ P ::= (\alpha, r).P | P + P \]

- **prefix**
- **choice**
**PEPA nets**

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston’s Performance Evaluation Process Algebra (PEPA).

\[
P ::= (\alpha, r).P \mid P + P \mid P \bowtie_P P
\]

- **prefix**
- **choice**
- **cooperation**
PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston’s Performance Evaluation Process Algebra (PEPA).

\[ P ::= (\alpha, r).P \mid P + P \mid P \bowtie_L P \mid P/L \]

- prefix
- choice
- cooperation
- hiding
PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston’s Performance Evaluation Process Algebra (PEPA).

\[ P ::= (\alpha, r).P \mid P + P \mid P \boxplus_P P \mid P/L \mid X \]

- prefix
- choice
- cooperation
- hiding
- variable
Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.
Objects as tokens

Consider a File class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a File object.
Objects as tokens

Consider a File class with methods openRead(), openWrite(), read(), write() and close().

The order in which the methods can be applied defines a protocol for a File object.
Objects as tokens

Consider a File class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a File object.
Objects as tokens

Consider a File class with methods openRead(), openWrite(), read(), write() and close().

The order in which the methods can be applied defines a protocol for a File object.
Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a `File` object.
Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a `File` object.
Objects as tokens

Consider a File class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`. The order in which the methods can be applied defines a protocol for a File object.
Objects as tokens

Consider a File class with methods openRead(), openWrite(), read(), write() and close().

The order in which the methods can be applied defines a protocol for a File object.
Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a `File` object.

We can express this as a PEPA component.

```
File  def = (openRead, r_o).InStream + (openWrite, r_o).OutStream
InStream  def = (read, r_r).InStream + (close, r_c).File
OutStream def = (write, r_w).OutStream + (close, r_c).File
```
Contexts and cells

A PEPA net is made up of PEPA **contexts**, one at each place in the net.
Contexts and cells

A PEPA net is made up of PEPA contexts, one at each place in the net.

Contexts contain static components and cells, which store tokens.
Contexts and cells

A PEPA net is made up of PEPA contexts, one at each place in the net.

Contexts contain static components and cells, which store tokens.

A typical context might be the following:

\[\text{File[\_]} \overset{L}{\sqsupseteq} \text{FileReader}\]

where the synchronisation set \(L\) in this case is \(\overline{A}(\text{File})\), the complete action type set of the component, \((\text{openRead}, \text{read}, \text{close}, \ldots)\).
Token movement

Tokens move by participating in firings of the net.
Token movement

Tokens move by participating in firings of the net.

Continuing our example, we introduce an instant message as a type of transmissible file.

\[ \text{InstantMessage} \overset{\text{def}}{=} (\text{transmit}, r_t).\text{File} \]
Token movement

Tokens move by participating in firings of the net. Continuing our example, we introduce an instant message as a type of transmissible file.

\[ \text{InstantMessage} \overset{\text{def}}{=} (\text{transmit}, r_t).\text{File} \]

Part of a PEPA net which models the passage of instant messages is shown below.
Token movement

Tokens move by participating in firings of the net.

Continuing our example, we introduce an instant message as a type of transmissible file.

\[ \text{InstantMessage} \overset{\text{def}}{=} (\text{transmit}, r_t).\text{File} \]

Part of a PEPA net which models the passage of instant messages is shown below.
Token movement

Tokens move by participating in firings of the net. Continuing our example, we introduce an instant message as a type of transmissible file.

\[ \text{InstantMessage} \stackrel{\text{def}}{=} (\text{transmit}, r_t).\text{File} \]

Part of a PEPA net which models the passage of instant messages is shown below.

An instant message \( IM \) can be moved by the \text{transmit} firing. In moving it changes state to a \text{File} derivative, which can be read by the \text{FileReader}.
Semantics: enabling and output

An enabling is a mapping of input places to tokens and an output is a mapping of output places to vacant cells.
Semantics: enabling and output

An enabling is a mapping of input places to tokens and an output is a mapping of output places to vacant cells.

A transition $t$ has an enabling of firing type $\alpha$, $E(t, \alpha)$, if for each input place $P_i$ of $t$ there is a token $T$, in the current marking of $P_i$, which has a one-step $\alpha$-derivative, $T'$. 
Semantics: enabling and output

An enabling is a mapping of input places to tokens and an output is a mapping of output places to vacant cells.

A transition $t$ has an enabling of firing type $\alpha$, $E(t, \alpha)$, if for each input place $P_i$ of $t$ there is a token $T$, in the current marking of $P_i$, which has a one-step $\alpha$-derivative, $T'$.

For any transition $t$, an output, denoted $O(t)$, is a mapping from the output places of $t$ to vacant cells in the current mapping.
Semantics: concession and enabling

A transition $t$ has **concession** for a firing of type $\alpha$ if there is an enabling $E(t, \alpha)$ such that there is a bijective mapping from $E(t, \alpha)$ to an output $O(t)$, which preserves the types of tokens.
Semantics: concession and enabling

A transition $t$ has concession for a firing of type $\alpha$ if there is an enabling $E(t, \alpha)$ such that there is a bijective mapping from $E(t, \alpha)$ to an output $O(t)$, which preserves the types of tokens.

Such a firing is enabled if no other transition of higher priority is enabled.
Example: enabling, output and firing

\[ P_1 \xrightarrow{\alpha, r} (\alpha, r) \xrightarrow{t} P_2 \]

\[ P_1 = [P, Q] \]
\[ P_2 = [R] \]
\[ P_3 = [\_], R[\_] \]
\[ P_4 = [Q', R] \]
Example: enabling, output and firing

\[ P[\alpha, r] \rightarrow P'[\alpha, r] \]

Diagram:

- \( P_1 \): \( P[P], Q[Q] \) → \( P[\_], R[\_] \) \( P_3 \)
- \( P_2 \): \( R[R] \) → \( Q[Q'], R[\_] \) \( P_4 \)

\( P \stackrel{(\alpha, r)}{\rightarrow} P' \),
Example: enabling, output and firing

\[ P \xrightarrow{\langle \alpha, r \rangle} P', \quad Q \xrightarrow{\langle \alpha, r \rangle} Q' , \]

\[ P_1 \xrightarrow{(\alpha, r)} [P], Q_1 \xrightarrow{(\alpha, r)} [Q] \]

\[ P_2 \xrightarrow{t} R \]

\[ P_3 \xrightarrow{\alpha, r} [P], R_3 \xrightarrow{\alpha, r} [R] \]

\[ P_4 \xrightarrow{\alpha, r} [Q'], R_4 \xrightarrow{\alpha, r} [R] \]
Example: enabling, output and firing

\[ P_1 \quad P[P], Q[Q] \quad \xrightarrow{(\alpha, r)} \quad P[\_], R[\_] \quad P_3 \]

\[ P_2 \quad R[R] \quad \xrightarrow{t} \quad Q[Q'], R[\_] \quad P_4 \]

\[ P \xrightarrow{(\alpha, r)} P', \quad Q \xrightarrow{(\alpha, r)} Q', \quad R \xrightarrow{(\alpha, r)} R' \]
Example: enabling, output and firing

\[ P(\alpha, r) \rightarrow P', \quad Q(\alpha, r) \rightarrow Q', \quad R(\alpha, r) \rightarrow R' \]

Enablings: \(((P_1, P), (P_2, R)), ((P_1, Q), (P_2, R))\)
Example: enabling, output and firing

Enablings: \(((P_1, P), (P_2, R)), ((P_1, Q), (P_2, R))\)

Outputs: \(((P_3, P), (P_4, R)), ((P_3, R), (P_4, R))\)
Example: enabling, output and firing

\[ P_1 \xrightarrow{(\alpha, r)} P', \quad Q \xrightarrow{(\alpha, r)} Q', \quad R \xrightarrow{(\alpha, r)} R' \]

Enablings: \(((P_1, P), (P_2, R)), ((P_1, Q), (P_2, R))\)

Outputs: \(((P_3, P), (P_4, R)), ((P_3, R), (P_4, R))\)

Firing: \(((P_1, P), (P_2, R)) \rightarrow ((P_3, P), (P_4, R))\)
Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.
Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.
Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.

The state space of the model is typically too large to allow us to consider every state individually.
Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.

The state space of the model is typically too large to allow us to consider every state individually.

Many types of analysis (steady-state, transient, passage time) begin by identifying distinguished subsets of the state space.
Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.
Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.
Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.
Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.
Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.
Tokens

\[ \text{SoapMessage} \stackrel{\text{def}}{=} (send_{clr}, r_{sc}).\text{SentClearMessage} \]
\[ + (encrypt, r_{e}).\text{EncryptedMsg} \]
\[ + (parse, r_{p}).\text{DOMtree} \]
**Tokens**

\[\text{SoapMessage} \overset{\text{def}}{=} (send_{clr}, r_{sc}).\text{SentClearMessage} + (encrypt, r_{e}).\text{EncryptedMsg} + (parse, r_{p}).\text{DOMtree}\]

\[\text{SentClearMessage} \overset{\text{def}}{=} (\text{copyClear}, \top).\text{SoapMessage}\]
Tokens

\[
\text{SoapMessage} \overset{\text{def}}{=} (send_{\text{clr}}, r_{\text{sc}}).\text{SentClearMessage} \\
+ (encrypt, r_e).\text{EncryptedMsg} \\
+ (parse, r_p).\text{DOMtree}
\]

\[
\text{SentClearMessage} \overset{\text{def}}{=} (\text{copyClear}, \top).\text{SoapMessage}
\]

\[
\text{EncryptedMsg} \overset{\text{def}}{=} (\text{decrypt}, r_d).\text{SoapMessage} \\
+ (send_{\text{enc}}, r_{\text{se}}).\text{SentEncMessage}
\]
Tokens

\[
\text{SoapMessage} \overset{\text{def}}{=} (send_{\text{clr}}, r_{\text{sc}}).\text{SentClearMessage} + (encrypt, r_{\text{e}}).\text{EncryptedMsg} + (parse, r_{\text{p}}).\text{DOMtree}
\]

\[
\text{SentClearMessage} \overset{\text{def}}{=} (\text{copyClear}, \top).\text{SoapMessage}
\]

\[
\text{EncryptedMsg} \overset{\text{def}}{=} (decrypt, r_{\text{d}}).\text{SoapMessage} + (send_{\text{enc}}, r_{\text{se}}).\text{SentEncMessage}
\]

\[
\text{SentEncMessage} \overset{\text{def}}{=} (\text{copyEncrypted}, \top).\text{EncryptedMsg}
\]


**Tokens**

\[
\begin{align*}
\text{SoapMessage} & \overset{\text{def}}{=} (\text{send}_{clr}, r_{sc}).\text{SentClearMessage} \\
& + (\text{encrypt}, r_{e}).\text{EncryptedMsg} \\
& + (\text{parse}, r_{p}).\text{DOMtree}
\end{align*}
\]

\[
\begin{align*}
\text{SentClearMessage} & \overset{\text{def}}{=} (\text{copyClear}, \top).\text{SoapMessage}
\end{align*}
\]

\[
\begin{align*}
\text{EncryptedMsg} & \overset{\text{def}}{=} (\text{decrypt}, r_{d}).\text{SoapMessage} \\
& + (\text{send}_{enc}, r_{se}).\text{SentEncMessage}
\end{align*}
\]

\[
\begin{align*}
\text{SentEncMessage} & \overset{\text{def}}{=} (\text{copyEncrypted}, \top).\text{EncryptedMsg}
\end{align*}
\]

\[
\begin{align*}
\text{DOMtree} & \overset{\text{def}}{=} (\text{read}, r_{r}).\text{DOMtree} \\
& + (\text{modify}, r_{m}).\text{DOMtree} \\
& + (\text{export}, r_{x}).\text{SoapMessage}
\end{align*}
\]
Static components

\[
\text{User} \overset{\text{def}}{=} \\
(\text{encrypt, } \top).(\text{send}_{\text{enc}, } \top).\text{User} \\
+ (\text{decrypt, } \top).(\text{parse, } \top).(\text{read, } \top).(\text{modify, } \top).(\text{export, } \top).\text{User}
\]
Static components

User \[\overset{\text{def}}{=}\]
\[(\text{encrypt}, \top).(\text{send}_{\text{enc}}, \top).\text{User}\]
\[+ (\text{decrypt}, \top).(\text{parse}, \top).(\text{read}, \top).(\text{modify}, \top).(\text{export}, \top).\text{User}\]

GateKeeper \[\overset{\text{def}}{=}\]
\[(\text{decrypt}, \top).(\text{send}_{\text{clr}}, \top).\text{GateKeeper}\]
\[+ (\text{decrypt}, \top).(\text{encrypt}, \top).(\text{send}_{\text{enc}}, \top).\text{GateKeeper}\]
\[+ (\text{encrypt}, \top).(\text{send}_{\text{enc}}, \top).\text{GateKeeper}\]
Static components

\begin{align*}
\text{User} & \overset{\text{def}}{=} \\
& (\text{encrypt}, T).(\text{send}_{\text{enc}}, T).\text{User} \\
& + (\text{decrypt}, T).(\text{parse}, T).(\text{read}, T).(\text{modify}, T).(\text{export}, T).\text{User} \\
\text{GateKeeper} & \overset{\text{def}}{=} \\
& (\text{decrypt}, T).(\text{send}_{\text{clr}}, T).\text{GateKeeper} \\
& + (\text{decrypt}, T).(\text{encrypt}, T).(\text{send}_{\text{enc}}, T).\text{GateKeeper} \\
& + (\text{encrypt}, T).(\text{send}_{\text{enc}}, T).\text{GateKeeper} \\
\text{WebService} & \overset{\text{def}}{=} (\text{parse}, T).(\text{read}, T). \\
& (\text{modify}, T).(\text{export}, T).(\text{send}_{\text{clr}}, T).\text{WebService}
\end{align*}
Client side

\[ \text{User} \overset{L}{\sqsupseteq} \text{SoapMessage}[\text{SoapMessage}] \]

\( (\text{copyEncrypted}, r_{ce}) \) \quad \text{GateKeeper} \overset{L}{\sqsupseteq} \text{EncryptedMsg}[\ldots] \quad (\text{copyEncrypted}, r_{ce}) 

\( (\text{copyClear}, r_{cc}) \) \quad \text{WebService} \overset{L}{\sqsupseteq} \text{SoapMessage}[\ldots] 

Server side
Client side

\[(\text{send}_{\text{enc}}, \top).\text{User} \parallel_{L} \text{SoapMessage}[\text{EncryptedMsg}]\]

\[(\text{copyEncrypted}, r_{ce}) \quad (\text{copyEncrypted}, r_{ce})\]

\[\text{GateKeeper} \parallel_{L} \text{EncryptedMsg}[\_\_\_\_\_]\]

\[(\text{copyClear}, r_{cc}) \quad (\text{copyClear}, r_{cc})\]

\[\text{WebService} \parallel_{L} \text{SoapMessage}[\_\_\_\_\_]\]

Server side
PEPA net

Client side

\[ \text{User} \xrightleftharpoons[\text{L}] \text{SoapMessage}[\text{SentEncMessage}] \]

\[ \text{(copyEncrypted,} r_{ce}) \]

\[ \Downarrow \]

\[ \text{GateKeeper} \xrightleftharpoons[\text{L}] \text{EncryptedMsg}[\text{]]} \]

\[ \text{(copyClear,} r_{cc}) \]

\[ \Downarrow \]

\[ \text{WebService} \xrightleftharpoons[\text{L}] \text{SoapMessage}[\text{]} \]

Server side
**PEPA net**

Client side

\[
\text{User} \overset{L}{\bowtie} \text{SoapMessage}[\_\_\_\_\_\_\_\_\_\_]
\]

\[
\text{(copyEncrypted}, r_{ce}) \quad \downarrow \quad (\text{copyEncrypted}, r_{ce})
\]

\[
(\text{encrypt}, \top).\text{GateKeeper'} \overset{L}{\bowtie} \text{EncryptedMsg}[\text{SoapMessage}]
\]

\[
(\text{copyClear}, r_{cc}) \quad \downarrow \quad (\text{copyClear}, r_{cc})
\]

\[
\text{WebService} \overset{L}{\bowtie} \text{SoapMessage}[\_\_\_\_\_\_\_\_\_\_]
\]

Server side
**Client side**

\[ \text{User} \overset{\text{L}}{\otimes} \text{SoapMessage} \]

\( (\text{copyEncrypted}, r_{ce}) \)

\( \uparrow \)

\( (\text{copyEncrypted}, r_{ce}) \)

\( \downarrow \)

\( (\text{send}_{enc}, \top).\text{GateKeeper} \overset{\text{L}}{\otimes} \text{EncryptedMsg} \)

\( (\text{copyClear}, r_{cc}) \)

\( \uparrow \)

\( (\text{copyClear}, r_{cc}) \)

\( \downarrow \)

\[ \text{WebService} \overset{\text{L}}{\otimes} \text{SoapMessage} \]

**Server side**
PEPA net

Client side

\[ User \overset{L}{\boxright} \text{SoapMessage}[\text{EncryptedMsg}] \]

\[ (\text{copyEncrypted}, r_{ce}) \]

\[ \downarrow \]

\[ \text{GateKeeper} \overset{L}{\boxright} \text{EncryptedMsg} \]

\[ (\text{copyClear}, r_{cc}) \]

\[ \downarrow \]

\[ \text{WebService} \overset{L}{\boxright} \text{SoapMessage} \]

Server side
Client side

\[(\text{parse}, \top). User' \parallel_L SoapMessage[SoapMessage]\]

\[(\text{copyEncrypted}, r_{ce}) \Downarrow \quad (\text{copyEncrypted}, r_{ce})\]

\[GateKeeper \parallel_L EncryptedMsg[\_\_\_\_]\]

\[(\text{copyClear}, r_{cc}) \Downarrow \quad (\text{copyClear}, r_{cc})\]

\[WebService \parallel_L SoapMessage[\_\_\_\_]\]

Server side
Client side

\((\text{read, } \top).\text{User}'' \otimes L\text{SoapMessage}[\text{DOMtree}]\)

\((\text{copyEncrypted}, r_{ce}) \downarrow \quad \uparrow (\text{copyEncrypted}, r_{ce})\)

\(\text{GateKeeper} \otimes L\text{EncryptedMsg}[\phantom{\text{DOMtree}}]\)

\((\text{copyClear}, r_{cc}) \downarrow \quad \uparrow (\text{copyClear}, r_{cc})\)

\(\text{WebService} \otimes L\text{SoapMessage}[\phantom{\text{DOMtree}}]\)

Server side
PEPA net

Client side

\[(\text{modify, } \top).\text{User}'''. \bowtie \text{SoapMessage}[\text{DOMtree}]\]

\[(\text{copyEncrypted}, r_{ce}) \downarrow \quad \uparrow \quad (\text{copyEncrypted}, r_{ce})\]

\[\text{GateKeeper} \bowtie \text{EncryptedMsg}[\quad \quad \quad \quad \quad]\]

\[(\text{copyClear}, r_{cc}) \downarrow \quad \uparrow \quad (\text{copyClear}, r_{cc})\]

\[\text{WebService} \bowtie \text{SoapMessage}[\quad \quad \quad \quad \quad]\]

Server side
Client side

\[ (\text{export}, \top).\text{User} \overset{\text{copyEncrypted}, r_{ce}}{\bowtie} \text{SoapMessage}[\text{DOMtree}] \]

\[ \text{GateKeeper} \overset{\text{copyClear}, r_{cc}}{\bowtie} \text{EncryptedMsg}[\_\_\_] \]

Server side

\[ \text{WebService} \overset{\text{copyClear}, r_{cc}}{\bowtie} \text{SoapMessage}[\_\_\_] \]
PEPA net

Client side

\[ User_\bowtie L SoapMessage[SoapMessage] \]

\[ \text{(copyEncrypted}, r_{ce}) \]

\[ \text{(copyEncrypted}, r_{ce}) \]

\[ GateKeeper_\bowtie L EncryptedMsg[_____________________] \]

\[ \text{(copyClear}, r_{cc}) \]

\[ \text{(copyClear}, r_{cc}) \]

\[ WebService_\bowtie L SoapMessage[_____________________] \]

Server side
(send_{enc}, \top).User \times_{L} SoapMessage[EncryptedMsg]

\begin{align*}
& (\text{copyEncrypted}, r_{ce}) \\
& \Downarrow \\
& \text{GateKeeper} \times_{L} EncryptedMsg[\_\_\_\_\_\_\_] \\
& \Downarrow \\
& (\text{copyClear}, r_{cc}) \\
& \Uparrow \\
& \text{WebService} \times_{L} SoapMessage[\_\_\_\_\_\_] 
\end{align*}
Client side

\[ \text{User} \sqsupset \text{SoapMessage}[\text{SentEncMessage}] \]

\[ (\text{copyEncrypted,} r_{ce}) \downarrow \]  

\[ (\text{copyEncrypted,} r_{ce}) \uparrow \]

\[ \text{GateKeeper} \sqsupset \text{EncryptedMsg}[\_\_\_] \]

\[ (\text{copyClear,} r_{cc}) \downarrow \]  

\[ (\text{copyClear,} r_{cc}) \uparrow \]

\[ \text{WebService} \sqsupset \text{SoapMessage}[\_\_\_] \]

Server side
**Client side**

\[ \text{User} \xrightarrow{L} \text{SoapMessage}[\ ] \]

\[(\text{copyEncrypted}, r_{ce}) \]

\[\text{GateKeeper} \xrightarrow{L} \text{EncryptedMsg}[\text{EncryptedMsg}]\]

\[(\text{copyClear}, r_{cc}) \]

**Server side**

\[ \text{WebService} \xrightarrow{L} \text{SoapMessage}[\ ] \]
Client side

\[
\text{User } \overset{L}{\rightarrow} \text{SoapMessage}[\_\_\_\_\_\_\_\_\_]
\]

\[
(\text{copyEncrypted}, r_{ce}) \quad \downarrow \quad \uparrow \quad (\text{copyEncrypted}, r_{ce})
\]

\[
(\text{send}_{clr}, \top).\text{GateKeeper } \overset{L}{\rightarrow} \text{EncryptedMsg[SoapMessage]}
\]

\[
(\text{copyClear}, r_{cc}) \quad \downarrow \quad \uparrow \quad (\text{copyClear}, r_{cc})
\]

\[
\text{WebService } \overset{L}{\rightarrow} \text{SoapMessage}[\_\_\_\_\_\_\_\_\_]
\]

Server side
**PEPA net**

**Client side**

\[ \text{User } \overset{L}{\triangleright} \text{SoapMessage}[\_\_\_] \]

\text{(copyEncrypted,} r_{ce}) \downarrow \quad (\text{copyEncrypted,} r_{ce}) \uparrow \quad \text{(copyEncrypted,} r_{ce})

\[ \text{GateKeeper } \overset{L}{\triangleright} \text{EncryptedMsg}[\text{SentClearMessage}] \]

\text{(copyClear,} r_{cc}) \downarrow \quad (\text{copyClear,} r_{cc}) \uparrow \quad (\text{copyClear,} r_{cc})

\[ \text{WebService } \overset{L}{\triangleright} \text{SoapMessage}[\_\_\_] \]

**Server side**
Client side

\[ \text{User} \overset{L}{\otimes} \text{SoapMessage}[\text{SoapMessage}] \]

\( (\text{copyEncrypted}, r_{ce}) \)

\[ \text{GateKeeper} \overset{L}{\otimes} \text{EncryptedMsg}[\text{EncryptedMsg}] \]

\( (\text{copyClear}, r_{cc}) \)

\[ \text{WebService} \overset{L}{\otimes} \text{SoapMessage}[\text{SoapMessage}] \]

Server side
Client side

\[ User \xleftarrow{L} SoapMessage[\text{[ ]}] \]

\((\text{copyEncrypted},r_{ce}) \xrightarrow{} \text{[ ]} \) \(\xleftarrow{} (\text{copyEncrypted},r_{ce})\)

\[ GateKeeper \xleftarrow{L} EncryptedMsg[\text{[ ]}] \]

\((\text{copyClear},r_{cc}) \xrightarrow{} \text{[ ]} \) \(\xleftarrow{} (\text{copyClear},r_{cc})\)

\((\text{read}, \top).\ WebService' \xleftarrow{L} SoapMessage[\text{DOMtree}]\)

Server side
PEPA net

**Client side**

User \( △ L \) SoapMessage[__________]

\((\text{copyEncrypted}, r_{ce}) \downarrow \)  
\((\text{copyEncrypted}, r_{ce}) \uparrow \)

GateKeeper \( △ L \) EncryptedMsg[__________]

\((\text{copyClear}, r_{cc}) \downarrow \)  
\((\text{copyClear}, r_{cc}) \uparrow \)

\((\text{modify}, ⊤). \text{WebService}'' \ △ L \) SoapMessage[DOMtree]

**Server side**
Client side

\[ User \otimes_{L} SoapMessage[\_\_\_\_\_\_\_] \]

\( (\text{copyEncrypted}, r_{ce}) \)

\[ GateKeeper \otimes_{L} EncryptedMsg[\_\_\_\_\_\_\_] \]

\( (\text{copyClear}, r_{cc}) \)

\( (\text{export}, \top).WebService'''' \otimes_{L} SoapMessage[DOMtree] \)

Server side
Client side

User ⦿ SoapMessage[---------------]

(copyEncrypted,r_{ce}) ↓  (copyEncrypted,r_{ce})

GateKeeper ⦿ EncryptedMsg[---------------]

(copyClear,r_{cc}) ↓  (copyClear,r_{cc})

(send_{clr}, ⊤).WebService ⦿ SoapMessage[SoapMessage]

Server side
PEPA net

Client side

User $\nabla_L SoapMessage[\ ]$

(copyEncrypted, $r_{ce}$) $\downarrow$ (copyEncrypted, $r_{ce}$)

GateKeeper $\nabla_L EncryptedMsg[\ ]$

(copyClear, $r_{cc}$) $\downarrow$ (copyClear, $r_{cc}$)

WebService $\nabla_L SoapMessage[SentClearMessage]$
PEPA net

Client side

User \(\overset{L}{\trieve}\) SoapMessage\[\Box\]

\((\text{copyEncrypted}, r_{ce})\) \(\downarrow\) \(\uparrow\) \((\text{copyEncrypted}, r_{ce})\)

GateKeeper \(\overset{L}{\trieve}\) EncryptedMsg[SoapMessage]

\((\text{copyClear}, r_{cc})\) \(\downarrow\) \(\uparrow\) \((\text{copyClear}, r_{cc})\)

WebService \(\overset{L}{\trieve}\) SoapMessage\[\Box\]

Server side
Client side

\[ User_{L} \triangleright SoaMessage[\_\_\_\_\_\_] \]

\( \text{(copyEncrypted, } r_{ce} \text{)} \)

\( \text{(copyEncrypted, } r_{ce} \text{)} \)

\( (send_{enc}, \top) \cdot GateKeeper_{L} \triangleright EncryptedMsg[EncryptedMsg] \)

\( \text{(copyClear, } r_{cc} \text{)} \)

\( \text{(copyClear, } r_{cc} \text{)} \)

\[ WebService_{L} \triangleright SoaMessage[\_\_\_\_\_] \]

Server side
PEPA net

Client side

User $\boxdot L$ SoapMessage[___________]  
(copyEncrypted,$r_{ce}$) ↓  
GateKeeper $\boxdot L$ EncryptedMsg[SentEncMessage]  
(copyClear,$r_{cc}$) ↓  
WebService $\boxdot L$ SoapMessage[___________]

Server side
Client side

User \( \otimes \) SoapMessage[EncryptedMsg]

\((\text{copyEncrypted}, r_{ce})\) \hspace{2cm} \hspace{2cm} \text{(copyEncrypted}, r_{ce})

GateKeeper \( \otimes \) EncryptedMsg[\hspace{1cm}]

\((\text{copyClear}, r_{cc})\) \hspace{2cm} \hspace{2cm} \text{(copyClear}, r_{cc})

WebService \( \otimes \) SoapMessage[\hspace{1cm}]

Server side
Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.
Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.

Usually the high-level model is used to derive a Continuous-Time Markov Chain (CTMC) for performance analysis.
Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.

Usually the high-level model is used to derive a Continuous-Time Markov Chain (CTMC) for performance analysis.

We can derive a CTMC directly from a PEPA net using the PEPA Workbench for PEPA nets.
Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.

Usually the high-level model is used to derive a Continuous-Time Markov Chain (CTMC) for performance analysis.

We can derive a CTMC directly from a PEPA net using the PEPA Workbench for PEPA nets.

An alternative is to compile a PEPA net to an equivalent PEPA model and then use one of the PEPA tools.
Compiling PEPA nets to PEPA

The **PEPA net compiler** compiles a PEPA net to a PEPA model. Activities are renamed to enforce the PEPA net idiom that components at different places cannot synchronise on transitions.
Compiling PEPA nets to PEPA

The PEPA net compiler compiles a PEPA net to a PEPA model. Activities are renamed to enforce the PEPA net idiom that components at different places cannot synchronise on transitions.

The given net and the generated PEPA model produce isomorphic CTMCs (but via different labelled transition systems).
Compiling PEPA nets to PEPA

The PEPA net compiler compiles a PEPA net to a PEPA model. Activities are renamed to enforce the PEPA net idiom that components at different places cannot synchronise on transitions.

The given net and the generated PEPA model produce isomorphic CTMCs (but via different labelled transition systems).

The renaming of activities is systematic so that it is possible to recover the transition system of the PEPA net from the transition system of the PEPA model.
Solving larger PEPA nets

The motivation for compiling PEPA nets to PEPA models is to use the range of tools available for PEPA.
Solving larger PEPA nets

The motivation for compiling PEPA nets to PEPA models is to use the range of tools available for PEPA.

We solved the secure web service model using

- the PEPA net compiler;
- Jeremy Bradley’s Imperial PEPA compiler; and
- Will Knottenbelt’s DNAmaca Petri net analyser.
Solving larger PEPA nets

The motivation for compiling PEPA nets to PEPA models is to use the range of tools available for PEPA.

We solved the secure web service model using

- the PEPA net compiler;
- Jeremy Bradley’s Imperial PEPA compiler; and
- Will Knottenbelt’s DNAmaca Petri net analyser.

Alternatives: Möbius, PRISM.
Secure web service model results

We investigated the utilisation of the web service while varying the probability that user requests are bounced by the gatekeeper and the strength of the encryption used, captured as an overhead.
Summary

- PEPA nets are a high-level modelling language addressing the performance aspects of the design of modern software systems.

- Unlike a Petri net, tokens are programmable components, allowing direct modelling of stateful objects.

- Evaluation contexts at the places of the net allow the modeller to represent different areas of computation.

- Link: www.dcs.ed.ac.uk/pepa
end of slide show
Future work

- It is possible that the PEPA nets language could be extended, necessitating extensions to the existing tool support.

- One possibility would be to add a type system which ensures a consistent interface for tokens.

- Preliminary work on the use of logics to express performance queries should be further explored.

- Undertaking real-world examples and case studies is a good way to drive this process.