System behaviour in the presence of failures

Matthew Hennessy, U of Sussex
joint work with Adrian Francalanza.

Background:

- Wide-area distributed networks
- Location aware programming
- Network failures
  - nodes
  - links
- Programming in the presence of failures

Details: University of Sussex Technical Report 2005:01
Anonymous agents migrating between sites in a distributed network

\[ \text{SERV}[d? (x @ z) T] \mid \text{CL}[(\text{new } a) \text{ goto SERV}.d!(a @ k) \mid P] \]
Anonymous agents migrating between sites in a distributed network

\[
\text{SERV}[d?(x@z) \ T] \mid \text{CL}[(\text{new } a) \ \text{goto} \ \text{SERV}.d!(a@k) \mid P]
\]

- Mobility between CL and SERV depends on state of underlying network
- Calculus requires explicit representation of network
Network representations

- **l** live node
- **k** dead node
- Two way communication links
Network representations

- $l$ live node
- $k$ dead node
- two way communication links

many different possible choices
Network representations - $\Delta$

$$\Delta = \langle \mathcal{N}, \mathcal{D}, \mathcal{L} \rangle$$

where

- $\mathcal{N}$ a set of names; $\text{loc}(\mathcal{N})$ the subset of $\mathcal{N}$ which are locations
- $\mathcal{D} \subseteq \text{loc}(\mathcal{N})$ - dead locations
- $\mathcal{L} \subseteq \text{loc}(\mathcal{N}) \times \text{loc}(\mathcal{N})$ - connections between locations

$\mathcal{L}$ is reflexive symmetric - a *link set*
Network representations - $\Delta$

$\Delta = \langle \mathcal{N}, \mathcal{D}, \mathcal{L} \rangle$ where

- $\mathcal{N}$ a set of names; $\text{loc}(\mathcal{N})$ the subset of $\mathcal{N}$ which are locations
- $\mathcal{D} \subseteq \text{loc}(\mathcal{N})$ - dead locations
- $\mathcal{L} \subseteq \text{loc}(\mathcal{N}) \times \text{loc}(\mathcal{N})$ - connections between locations
  $\mathcal{L}$ is reflexive symmetric - a link set

Look up functions:

- $\Delta \vdash l : \text{alive}$
- $\Delta \vdash l \leftrightarrow k$ - a link
- $\Delta \vdash l \leftrightarrow k$ - active link
  if $\Delta \vdash l, k : \text{alive}, \ l \leftrightarrow k$
Process mobility

\[ l_1 \rightarrow k_1 \rightarrow n_1 \leftarrow k_2 \rightarrow l_2 \rightarrow k_2 \rightarrow k_3 \rightarrow l_3 \rightarrow n_3 \]
Process mobility

- $l_1[\text{goto } n_1.P]$ ? active path
- $k_2[\text{goto } n_3.P]$ ? any path
- $l_2[\text{goto } l_3.P]$ ? dead receiver
- $n_2[\text{goto } n_3.P]$ ? dead sender
- $k_2[\text{goto } k_3.P]$ ? active link
Process mobility

- \( l_1[\text{goto } n_1.P] \) ? active path
- \( k_2[\text{goto } n_3.P] \) ? any path
- \( l_2[\text{goto } l_3.P] \) ? dead receiver
- \( n_2[\text{goto } n_3.P] \) ? dead sender
- \( k_2[\text{goto } k_3.P] \) ? active link ∗
Network programming

How do processes take failures into account?
How do processes take failures into account?

- \( l_1[[\text{pathping } n_1.P]] \) ?
- \( l_1[[\text{pathping } n_3.P \text{ else } Q]] \) ?
- \( l_2[[\text{go } n_3.P \text{ elselocal } Q]] \) ?
- \( l_2[[\text{go } n_3.\text{(P andlocal R)elselocal } Q]] \) ?
- \( l_1[[\text{ping } l_2.P]] \) ?
- \( l_1[[\text{ping } l_2.P \text{ else } Q]] \) ?
Network programming

How do processes take failures into account?

- $l_1[\text{pathping } n_1.P]$ ?
- $l_1[\text{pathping } n_3.P \text{ else } Q]$ ?
- $l_2[\text{go } n_3.P \text{ else } Q]$ ?
- $l_2[\text{go } n_3.(P \text{ andlocal } R) \text{ else } Q]$ ?
- $l_1[\text{ping } l_2.P]$ ?
- $l_1[\text{ping } l_2.P \text{ else } Q]$ ? *
Network generation

Generating new nodes: $k_2[\nu m P]

What locations is the new $m$ connected with?

- only $k_2$?
- some declared set
Generating new nodes: \( k_2[\nu m \cdot P] \)

What locations is the new \( m \) connected with?

- only \( k_2 \) ?
- some declared set \( \star \)

\( k_2[\nu m \cdot \text{loc}_a[S] \cdot P] \)

connects the new \( m \) to all nodes in \( S \) accessible from \( k_2 \)
Network generation

\[ k_2 \left[ (\nu m: \text{loc}_a[\{n_1, n_3\}]) P \right] \text{ leads to } \]
Network generation

\[ k_2 \left[ (\nu m : \text{loc}_a[\{n_1, n_3\}]) P \right] \text{ leads to } \]

Dagstuhl, February 2005 – p.8/26
Syntax of DPIF

- **Configurations:** $\Delta \triangleright M$
- **Systems:** $M ::= l[P] | N | M | (\nu n : T)M$
- **Types:** $T ::= \text{ch} | \text{loc}_a[S] | \text{loc}_d[S]$
  
  $S$ a set of location names
- **Processes:**

  $$P ::= u!(V).P | u?(X).P | \ldots$$

  $$\text{goto } u.P | \text{ping } u.P \text{ else } Q | \text{kill} | \text{break } l$$
Use of kill and break

Modelling fragile nodes and links:

\[ l[[ (\nu k_1 : \text{loc}[\{l, n\}]) (\nu k_2 : \text{loc}[\{k_1, l\}])] \]

\[ \text{goto } k_1.P_1 \mid \text{goto } k_2.P_2 \]

\[ \mid \text{goto } k_1.\text{break } k_2 \]

Here link between $k_1$ and $k_2$ is subject to failure
Use of kill and break 
l

• Modelling fragile nodes and links:

\[ l [] (\nu k_1 : \text{loc}[\{l, n\}]) (\nu k_2 : \text{loc}[\{k_1, l\}]) \]

\[ \text{goto } k_1.P_1 \mid \text{goto } k_2.P_2 \]

\[ \mid \text{goto } k_1.\text{break } k_2 ] \]

Here link between \( k_1 \) and \( k_2 \) is subject to failure

\[ M \approx N \] will mean:

\( M \) and \( N \) are behaviourally equivalent in the presence of failures

– provided \( \approx \) is contextual
Reduction semantics of DPIF

\[\text{comm}\]
\[
\Delta \vdash l : \text{alive}
\]
\[
\Delta \triangleright l[a!(V) . P] | l[a? (X) . Q] \rightarrow \Delta \triangleright l[P] | \ldots
\]

\[\text{ping}\]
\[
\Delta \vdash l \leftrightarrow k
\]
\[
\Delta \triangleright l[\text{ping } k . P \text{ else } Q] \rightarrow \Delta \triangleright l[P]
\]

\[\text{not-ping}\]
\[
\Delta \nvdash l \leftrightarrow k
\]
\[
\Delta \triangleright l[\text{ping } k . P \text{ else } Q] \rightarrow \Delta \triangleright l[Q]
\]

\[\text{brk}\]
\[
\Delta \vdash l \leftrightarrow k
\]
\[
\Delta \triangleright l[\text{break } k] \rightarrow (\Delta - (l \leftrightarrow k)) \triangleright l[0]
\]
Reduction semantics - more

\[\text{newl} \]
\[\Delta \vdash l : alive \]
\[\Delta \triangleright l[[ (\nu k)\text{loc}_a[S] \ P ]] \longrightarrow \Delta \triangleright (\nu k: \text{loc}_a[D]) l[P] \]

where \( D \) is set of locations in \( S \) accessible from \( l \)

\[\text{go} \]
\[\Delta \vdash l \leftrightarrow k \]
\[\Delta \triangleright l[[ \text{goto } k.\ P ]] \longrightarrow \Delta \triangleright k[P] \]

\[\text{no-go} \]
\[\Delta \vdash l \not\leftrightarrow k \]
\[\Delta \triangleright l[[ \text{goto } k.\ P ]] \longrightarrow \Delta \triangleright k[0] \]

\[\ldots\]
Reduction barbed congruence

\[ \Delta \models M \cong N \]

means: \( M \) and \( N \) can not be distinguished by any observer

- interacting with \( M \) and \( N \)

- running on any network extension of \( \Delta \)
Reduction barbed congruence

\[ \Delta \models M \cong N \]

means: \( M \) and \( N \) can not be distinguished by any observer
- interacting with \( M \) and \( N \)
- running on any network extension of \( \Delta \)

Problem: Find *bisimulation equivalence* which captures exactly

\[ \Delta \models M \cong N \]
Reduction barbed congruence - example

\[ \Delta_l \models N_i \cong N_j \text{ where } \Delta_l \text{ - network with one live location } l \]

\[
\begin{align*}
N_1 & \iff (\nu k : \text{loc}_d[l])l[a!\langle k\rangle] \\
N_2 & \iff (\nu k : \text{loc}_d[\{}])l[a!\langle k\rangle] \\
N_3 & \iff (\nu k : \text{loc}_a[\{}])l[a!\langle k\rangle]
\end{align*}
\]
Reduction barbed congruence - example

\[ \Delta_l \models N_i \cong N_j \text{ where } \Delta_l - \text{network with one live location } l \]

\[
\begin{align*}
N_1 & \iff (\nu k : \text{loc}_d[\{l\}])l[a!\langle k \rangle] \\
N_2 & \iff (\nu k : \text{loc}_d[\{\}])l[a!\langle k \rangle] \\
N_3 & \iff (\nu k : \text{loc}_a[\{\}])l[a!\langle k \rangle]
\end{align*}
\]

Effective networks of running code \( l[a!\langle k \rangle] \):

\[
\begin{align*}
\Delta_1 &= \Delta_l + k : \text{loc}_d[\{l\}] \qquad = \quad \begin{array}{c} l \\ \rightarrow \end{array} k \\
\Delta_2 &= \Delta_l + k : \text{loc}_d[\{\}] \qquad = \quad \begin{array}{c} l \\ \end{array} k \\
\Delta_3 &= \Delta_l + k : \text{loc}_a[\{\}] \qquad = \quad \begin{array}{c} l \\ \end{array} k
\end{align*}
\]
Another example

\[ M_1 \iff (\nu k_1 : \{ l \}) (\nu k_2 : \{ k_1 \}) (\nu k_3 : \{ k_1, k_2 \}) l[a!\langle k_2, k_3 \rangle.P] \]
\[ M_2 \iff (\nu k_1 : \{ l \})(\nu k_2 : \{ k_1 \})(\nu k_3 : \{ k_1 \})l[a!\langle k_2, k_3 \rangle.P] \]

Is \( \Delta_l \models M_1 \cong M_2 \)?
Another example

\[ M_1 \leftarrow (\nu k_1 : \{l\}) (\nu k_2 : \{k_1\}) (\nu k_3 : \{k_1, k_2\}) l[a!\langle k_2, k_3 \rangle.P] \]
\[ M_2 \leftarrow (\nu k_1 : \{l\})(\nu k_2 : \{k_1\})(\nu k_3 : \{k_1\})l[a!\langle k_2, k_3 \rangle.P] \]

Is \( \Delta_l \models M_1 \cong M_2 \)?

Effective networks of running code \( l[a!\langle k_2, k_3 \rangle.P] \):

\[ M_1 : \]

\[ M_2 : \]
Resist temptation

\[ \Delta \triangleright M \xrightarrow{\mu} \Delta' \triangleright M' \] is not subtle enough
Resist temptation

\[ \Delta \triangleright M \xrightarrow{\mu} \Delta' \triangleright M' \] is not subtle enough

- Observers learn about new nodes, by extrusion
- Observers can discover connections between new nodes by using ping
- But must be able to access new nodes in order to ping
Resist temptation

\[ \Delta \triangleright M \xrightarrow{\mu} \Delta' \triangleright M' \] is not subtle enough

- Observers learn about new nodes, by extrusion
- Observers can discover connections between new nodes by using ping
- But must be able to access new nodes in order to ping

Result:

- Observers may only be aware of part of underlying network
- \( \Delta \) must represent both actual network, and observers knowledge of it
Extended network representations \( \Gamma \)

\[ \Gamma = \langle \mathcal{N}, \mathcal{O}, \mathcal{S} \rangle \]

where

- \( \mathcal{N} \) is a set of names - known to the observer
- \( \mathcal{O} \) a linkset of live links known to the observer
- \( \mathcal{S} \) a linkset of live links unknown to the observer
Extended network representations \( \Gamma \)

\[
\Gamma = \langle \mathcal{N}, \mathcal{O}, \overline{\mathcal{S}} \rangle
\]

where

- \( \mathcal{N} \) is a set of names - known to the observer
- \( \mathcal{O} \) a linkset of live links known to the observer
- \( \overline{\mathcal{S}} \) a linkset of live links unknown to the observer

Note:

- Dead nodes are known \textit{indirectly}
- Links to dead nodes are not recorded
Extended network representations $\Gamma$

$$\Gamma = \langle N, O, \overline{S} \rangle$$

where

- $N$ is a set of names - known to the observer
- $O$ a linkset of live links known to the observer
- $\overline{S}$ a linkset of live links unknown to the observer

Note:

- Dead nodes are known \textit{indirectly}
- Links to dead nodes are not recorded

Required actions: $\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright M'$
Lts for DPIF

\[ \Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright M' \]

where \( \mu \) can be

- \( \tau \)
- \( (\tilde{n} : \tilde{T}) l : a?(V) \)
  \( (\tilde{n}) \) are fresh names introduced by observer
  \( (\tilde{T}) \) are their state information
- \( (\tilde{n} : \tilde{T}) l : a!(V) \)
  \( (\tilde{n}) \) are fresh names exported to the observer
  \( (\tilde{T}) \) are their state information
- \( \text{kill}(l) \)
  external location kill
- \( l \leftrightarrow k \)
  external link break
Example rules

\[ \text{fail} \]
\[ \begin{array}{c}
\Gamma \vdash l : \text{alive} \\
\Pi \triangleright N \xrightarrow{\text{kill}(l)} (\Pi - l) \triangleright N
\end{array} \]

\[ \text{l–weak–in} \]
\[ \begin{array}{c}
(\Gamma + n : T) \triangleright N \xrightarrow{\alpha_{\text{in}}} \Gamma' \triangleright N' \\
\Gamma \triangleright N \xrightarrow{(n : T)\alpha_{\text{in}}} \Gamma' \triangleright N'
\end{array} \]

\[ \text{l–rest–typ} \]
\[ \begin{array}{c}
\Gamma + k : T \triangleright N \xrightarrow{(\tilde{n} : \tilde{T})l : a \langle V \rangle} (\Gamma + \tilde{n} : \tilde{U}) + k : U \triangleright N' \\
\Gamma \triangleright (\nu k : T)N \xrightarrow{(\tilde{n} : \tilde{U})l : a \langle V \rangle} (\Gamma + \tilde{n} : \tilde{U}) \triangleright (\nu k : U)N'
\end{array} \]
Soundness

Let $\Gamma \models M \approx N$ mean that $\Gamma \triangleright M$ and $\Gamma \triangleright N$ are weakly bisimilar.

Theorem: $\Gamma \models M \approx N$ implies $\Gamma \models M \simeq N$

Proof: Essentially $\approx$ is contextual
Soundness

Let $\Gamma \models M \approx N$ mean that $\Gamma \triangleright M$ and $\Gamma \triangleright N$ are weakly bisimilar.

Theorem: $\Gamma \models M \approx N$ implies $\Gamma \models M \sim N$
Proof: Essentially $\approx$ is contextual

But $\Gamma \models M \sim N$ does not imply $\Gamma \models M \approx N$

State information on actions is too detailed.
Counter-example

\[ N_1 \leftarrow (\nu k : \text{loc}_d[\{l\}])l[a!\langle k \rangle] \quad \text{with} \quad l \rightarrow k \]

\[ N_2 \leftarrow (\nu k : \text{loc}_d[\{\}])l[a!\langle k \rangle] \quad \text{with} \quad l \rightarrow k \]
Counter-example

\[ N_1 \leftarrow (\nu k : \text{loc}_d[\{l\}])l[a!\langle k \rangle] \quad \quad l \quad \quad k \]

\[ N_2 \leftarrow (\nu k : \text{loc}_d[\{\}])l[a!\langle k \rangle] \quad \quad l \quad \quad k \]

\[ \Gamma_l \triangleright N_1 \xrightarrow{\alpha_1} \ldots \triangleright \]

\[ \Gamma_l \triangleright N_2 \xrightarrow{\alpha_2} \ldots \triangleright \]

where

\[ \alpha_1 = (k : \text{loc}_d[\{l\}])l : a!\langle k \rangle \]

\[ \alpha_2 = (k : \text{loc}_d[\{\}])l : a!\langle k \rangle \]
Derived actions for DPIF

\[ \Gamma \triangleright N \xrightarrow{\mu} \Gamma' \triangleright N' \]

where

\[
\begin{align*}
\sigma & \\
(\tilde{n} : \tilde{L}) & l : a?\langle V \rangle \\
(\tilde{n} : \tilde{L}) & l : a!\langle V \rangle K \\
\text{kill}(l) & \\
l & \not\leftrightarrow k
\end{align*}
\]

Here \( \tilde{L} \) are live link sets: the live connections made visible by the appearance of the new \( (\tilde{n}) \).
Example

\[ N_1 \leftarrow (\nu k : \text{loc}_d[l])l[a!(k)] \quad l \rightarrow k \]

\[ N_2 \leftarrow (\nu k : \text{loc}_d[])l[a!(k)] \quad l \rightarrow k \]

\[ \Gamma_l \triangleright N_1 \xrightarrow{\alpha_1} \ldots \triangleright \ldots, \quad \alpha_1 = (k : \text{loc}_d[l])l : a!(k) \]

\[ \Gamma_l \triangleright N_2 \xrightarrow{\alpha_2} \ldots \triangleright \ldots, \quad \alpha_2 = (k : \text{loc}_d[{}])l : a!(k) \]
Example

\[ N_1 \leftrightarrow (\nu k : \text{loc}_d[\{l\}])l[a!(k)] \]

\[ N_2 \leftrightarrow (\nu k : \text{loc}_d[\{\}])l[a!(k)] \]

\[ \Gamma_l \triangleright N_1 \xrightarrow{\alpha_1} \ldots \triangleright \ldots, \quad \alpha_1 = (k : \text{loc}_d[\{l\}])l : a!(k) \]

\[ \Gamma_l \triangleright N_2 \xrightarrow{\alpha_2} \ldots \triangleright \ldots, \quad \alpha_2 = (k : \text{loc}_d[\{\}])l : a!(k) \]

Derived actions:

\[ \Gamma_l \triangleright N_1 \xleftarrow{\alpha} \ldots \triangleright \ldots \quad \alpha = (k : \{\})l : a!(k) \]

\[ \Gamma_l \triangleright N_2 \xleftarrow{\alpha} \ldots \triangleright \ldots \]
Example revisited

\[ \mathsf{ls}\; \Gamma_l \models M_1 \approx M_2 \]

\[
M_1 \iff (\nu k_1 : \{l\}) (\nu k_2 : \{k_1\}) (\nu k_3 : \{k_1, k_2\}) l[a!(k_2, k_3).P]
\]

\[
M_2 \iff (\nu k_1 : \{l\})(\nu k_2 : \{k_1\})(\nu k_3 : \{k_1\})l[a!(k_2, k_3).P]
\]

Effective networks of running code:

\[
M_1 : \\
\]

\[
M_2 : \\
\]

Dagstuhl, February 2005 – p.24/26
Full-abstraction

Theorem: In derived lts

\[ \Gamma \vdash M \approx N \text{ if and only if } \Gamma \vdash M \simeq N \]
Full-abstraction

Theorem: In derived lts

\[ \Gamma \models M \approx N \text{ if and only if } \Gamma \models M \sim N \]

Proof requires:

- contextuality of \( \approx \) in the derived lts
- definability of every derived action
- a formal definition of \( \sim \)

We need to be able to compare configurations running on different networks
Further work

- Application to examples
- Relativisation:
  - to a maximum number of failures
  - to certain permanent nodes, connections
- Fault tolerance
- other connectivity models
- other migration models