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

Mobile Systems: Dpi

Anonymous agents migrating between sites
in a distributed network

$$\textsc{serv}[\![d?(x@z)\, T]\!] \quad | \quad \textsc{cl}[\![(\text{new } a)\ \text{goto } \textsc{serv}.d!\langle a@k\rangle \ \ | \ \ P]\!]$$

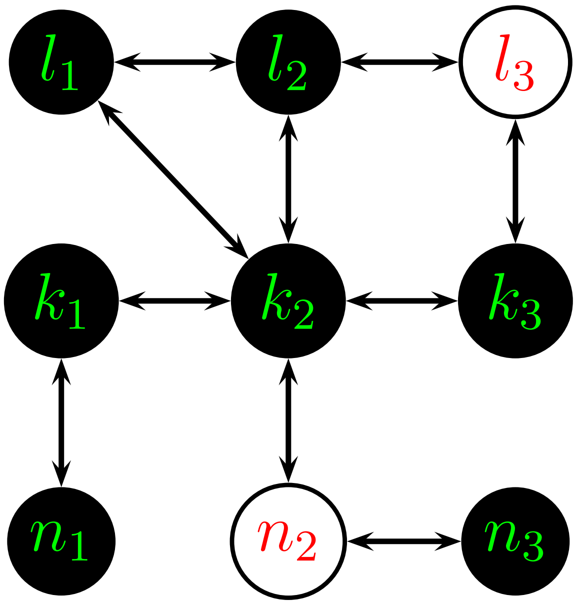
Mobile Systems: Dpi

Anonymous agents migrating between sites
in a distributed network

$$\textsc{serv}[\![d?(x@z)\,T]\!] \;\;|\;\; \textsc{cl}[\![(\text{new}\,a)\,\text{goto}\;\textsc{serv}.d!\langle a@k\rangle \;|\; P]\!]$$

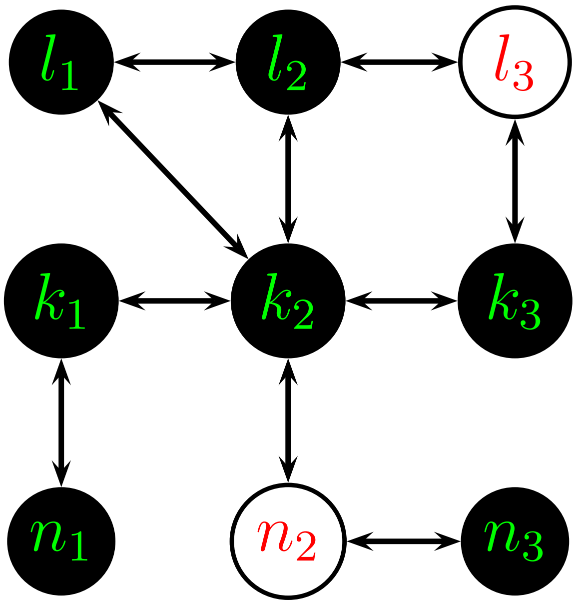
- Mobility between CL and SERV depends on state of underlying network
- Calculus requires explicit representation of network

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 - Δ

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

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

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

Network representations - Δ

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

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

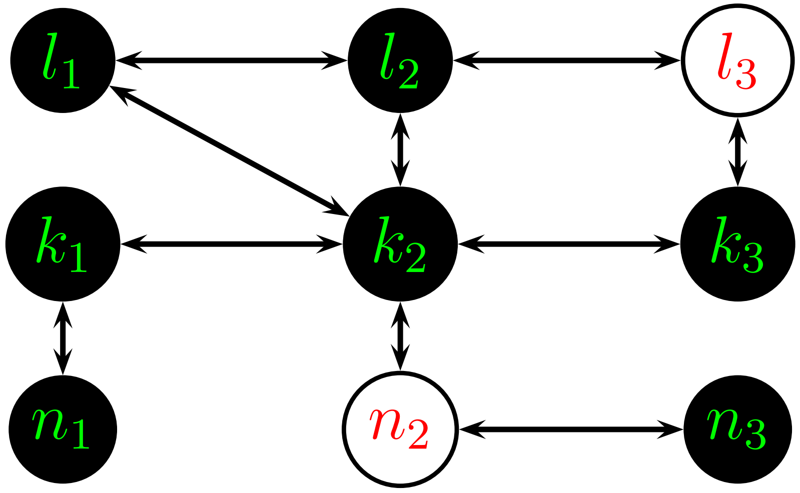
  $\mathcal{L}$ is reflexive symmetric - a *link set*

Look up functions:

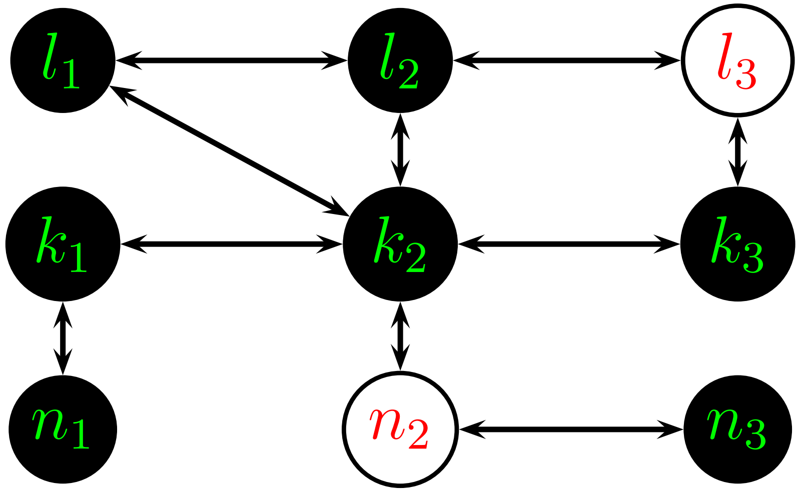
- $\Delta \vdash l : \mathbf{alive}$
- $\Delta \vdash l \leftrightarrow k$ - a link
- $\Delta \vdash l \leftrightsquigarrow k$ - active link

  if $\Delta \vdash l, k : \mathbf{alive}, \quad l \leftrightarrow k$

Process mobility

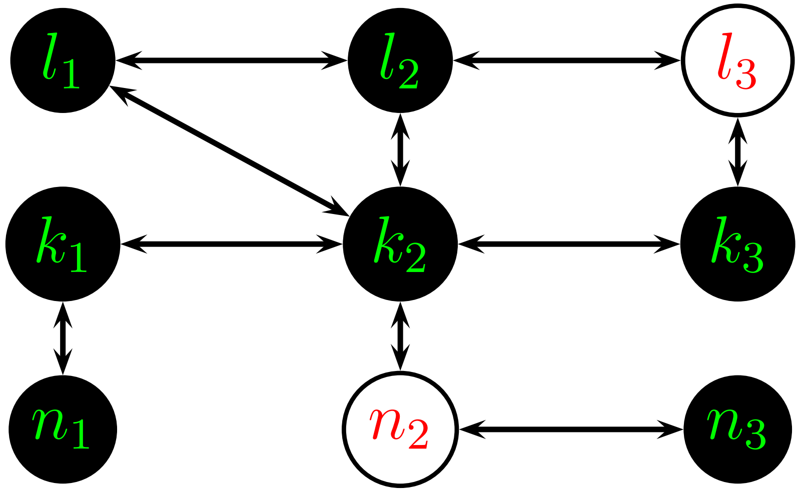


Process mobility



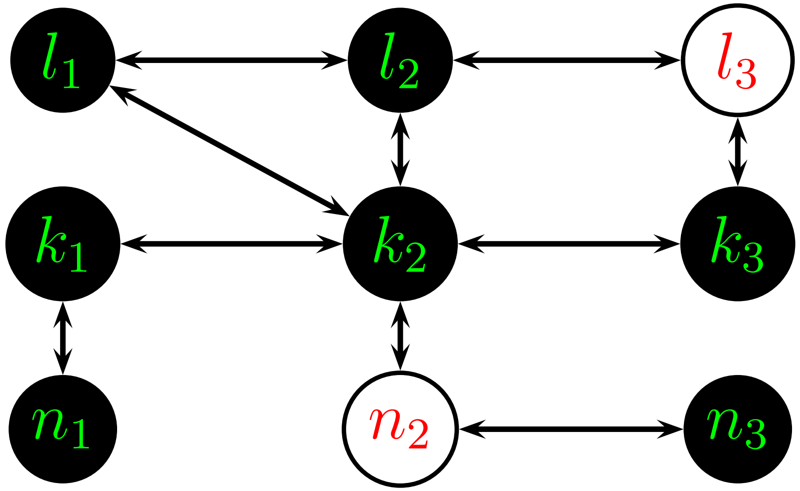
- $l_1[\![\text{goto } n_1.P]\!]$? active path
- $k_2[\![\text{goto } n_3.P]\!]$? any path
- $l_2[\![\text{goto } l_3.P]\!]$? dead receiver
- $n_2[\![\text{goto } n_3.P]\!]$? dead sender
- $k_2[\![\text{goto } k_3.P]\!]$? active link

Process mobility



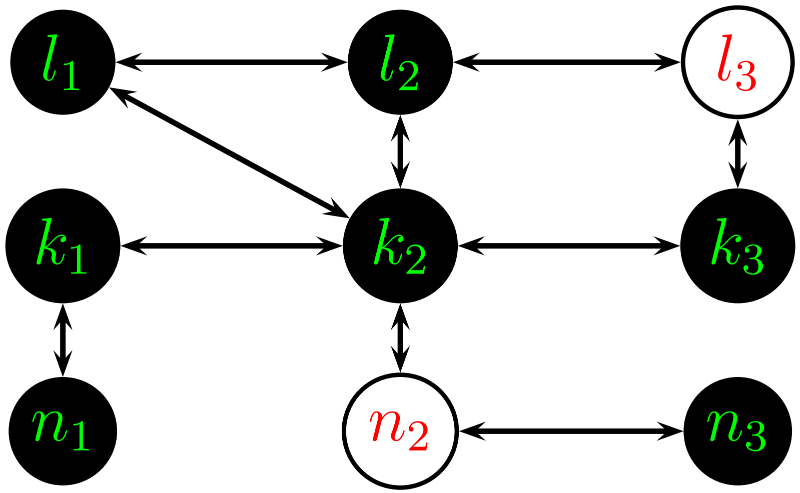
- $l_1[\![\text{goto } n_1.P]\!]$? active path
- $k_2[\![\text{goto } n_3.P]\!]$? any path
- $l_2[\![\text{goto } l_3.P]\!]$? dead receiver
- $n_2[\![\text{goto } n_3.P]\!]$? dead sender
- $k_2[\![\text{goto } k_3.P]\!]$? active link $\star$

Network programming



How do processes take failures into account?

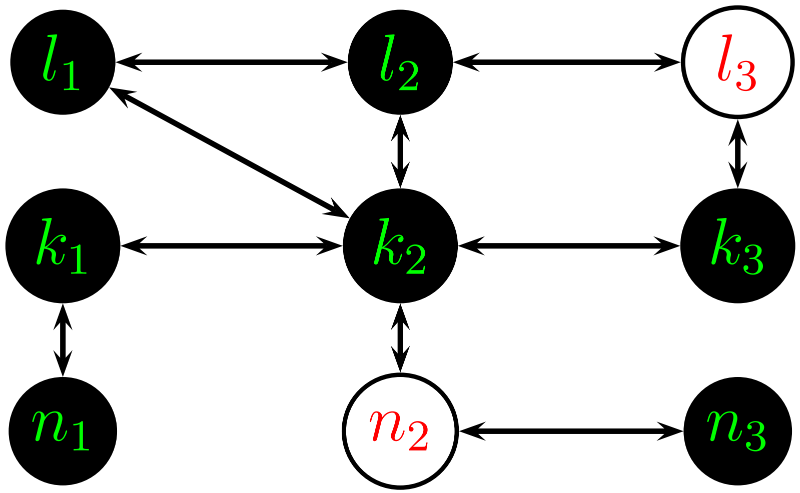
Network programming



How do processes take failures into account?

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

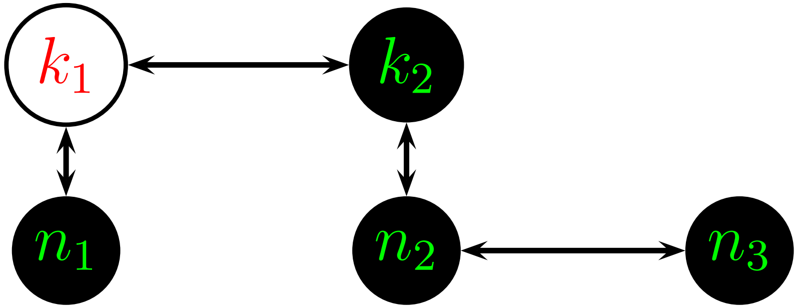
Network programming



How do processes take failures into account?

- $l_1[\![$pathping $n_1.P]\!]$?
- $l_1[\![$pathping $n_3.P$ else $Q]\!]$?
- $l_2[\![$go $n_3.P$ elselocal $Q]\!]$?
- $l_2[\![$go $n_3.(P$ andlocal $R)$elselocal $Q]\!]$?
- $l_1[\![$ping $l_2.P]\!]$?
- $l_1[\![$ping $l_2.P$ else $Q]\!]$? $\star$

Network generation

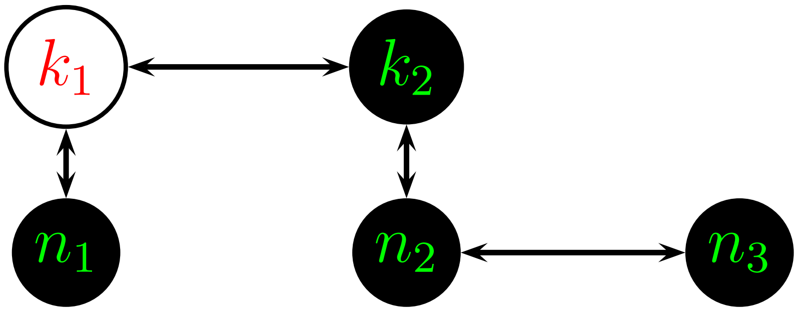


Generating new nodes: $k_2[\![(\nu\, m)P]\!]$

What locations is the new m connected with?

- only k_2 ?
- some declared set

Network generation



Generating new nodes: $k_2[\![(\nu\, m)P]\!]$

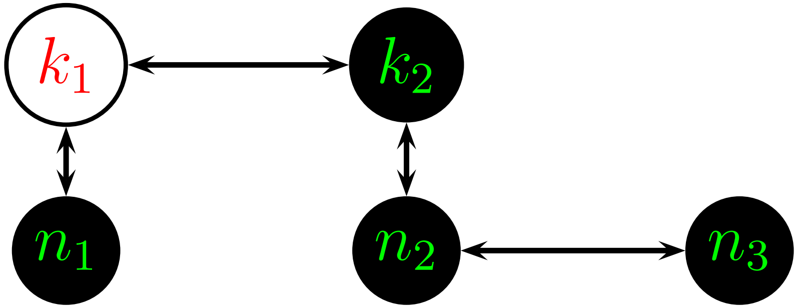
What locations is the new m connected with?

- only k_2 ?
- some declared set $\star$

$k_2[\![(\nu\, m : \mathtt{loc_a}[\mathtt{S}])P]\!]$

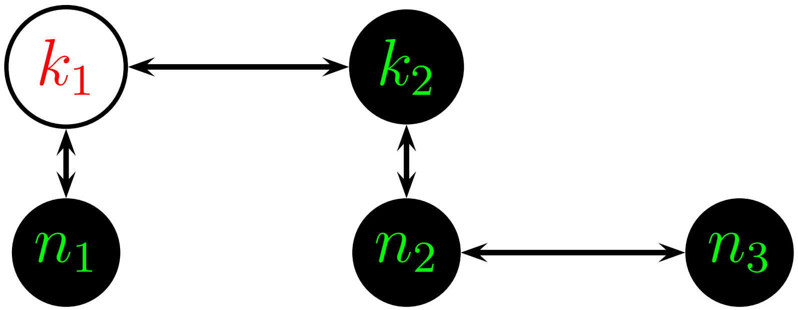
connects the new m to all nodes in $\mathtt{S}$ *accessible* from k_2

Network generation

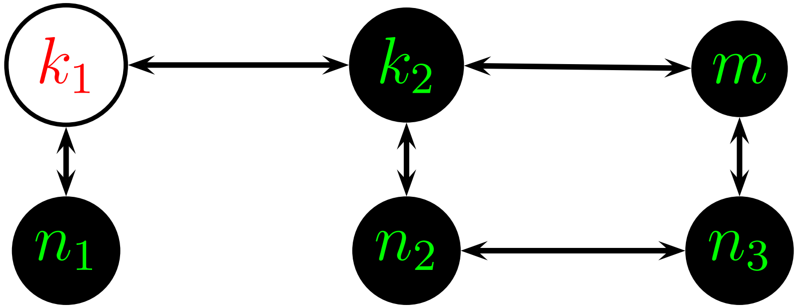


$k_2[\![(\nu\, m\!:\!\mathtt{loc}_{\mathtt{a}}[\{n_1, n_3\}])P]\!]$ leads to

Network generation



$k_2[\![(\nu\, m : \mathtt{loc_a}[\{n_1, n_3\}])P]\!]$ leads to



Syntax of DPIF

- **Configurations:** $\Delta \triangleright M$
- **Systems:** $M ::= l[\![P]\!] \mid N|M \mid (\nu\, n : \mathtt{T})M$
- **Types:** $\mathtt{T} ::= \mathtt{ch} \mid \mathtt{loc}_{\mathbf{a}}[\mathtt{S}] \mid \mathtt{loc}_{\mathbf{d}}[\mathtt{S}]$

  $\mathtt{S}$ a set of location names
- **Processes:**

$$P \quad ::= \quad u!\langle V\rangle.P \mid u?(X).P \mid \ldots$$

$$\text{goto } u.P \mid \text{ping } u.P \text{ else } Q \mid \text{kill} \mid \text{break } l$$

Use of kill and break l

- Modelling fragile nodes and links:

$$l[\![\ (\nu\, k_1 : \mathtt{loc}[\{l, n\}])(\nu\, k_2 : \mathtt{loc}[\{k_1, l\}])$$
$$\text{goto } k_1.P_1 \ \mid\ \text{goto } k_2.P_2$$
$$\mid\ \text{goto } k_1.\text{break } k_2\ ]\!]$$

Here link between k_1 and k_2 is subject to failure

Use of kill and break l

- Modelling fragile nodes and links:

$$l[\![\ (\nu\, k_1 : \mathtt{loc}[\{l, n\}])(\nu\, k_2 : \mathtt{loc}[\{k_1, l\}])$$

$$\text{goto } k_1.P_1 \mid \text{goto } k_2.P_2$$

$$\mid \text{goto } k_1.\text{break } k_2\ ]\!]$$

  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

- *comm*
$$\frac{\Delta \vdash l : \mathbf{alive}}{\Delta \triangleright l[\![a!\langle V\rangle.P]\!] \mid l[\![a?(X).Q]\!] \;\longrightarrow\; \Delta \triangleright l[\![P]\!] \mid \ldots}$$

- *ping*
$$\frac{\Delta \vdash l \leftrightsquigarrow k}{\Delta \triangleright l[\![\text{ping } k.P \text{ else } Q]\!] \longrightarrow \Delta \triangleright l[\![P]\!]}$$

- *not−ping*
$$\frac{\Delta \not\vdash l \leftrightsquigarrow k}{\Delta \triangleright l[\![\text{ping } k.P \text{ else } Q]\!] \longrightarrow \Delta \triangleright l[\![Q]\!]}$$

- *brk*
$$\frac{\Delta \vdash l \leftrightsquigarrow k}{\Delta \triangleright l[\![\text{break } k]\!] \longrightarrow (\Delta - (l \leftrightarrow k)) \triangleright l[\![\mathbf{0}]\!]}$$

Reduction semantics - more

- $$\frac{\Delta \vdash l : alive}{\Delta \triangleright l[\![(\nu k)\texttt{loc}_{\mathbf{a}}[\texttt{S}]\ P]\!] \longrightarrow \Delta \triangleright (\nu\, k : \texttt{loc}_{\mathbf{a}}[\texttt{D}])\ l[\![P]\!]}\ newl$$

  where D is set of locations in S accessible from l

- $$\frac{\Delta \vdash l \leftrightsquigarrow k}{\Delta \triangleright l[\![\text{goto } k.P]\!] \longrightarrow \Delta \triangleright k[\![P]\!]}\ go$$

- $$\frac{\Delta \vdash l \not\leftrightsquigarrow k}{\Delta \triangleright l[\![\text{goto } k.P]\!] \longrightarrow \Delta \triangleright k[\![\mathbf{0}]\!]}\ no{-}go$$

- ...

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 Δ

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 Δ

Problem: Find *bisimulation equivalence* which captures exactly

$$\Delta \models M \cong N$$

.

Reduction barbed congruence - example

$\Delta_l \models N_i \cong N_j$ where Δ_l - network with one live location l

$$
\begin{aligned}
N_1 &\Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{l\}])l[\![a!\langle k\rangle]\!] \\
N_2 &\Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{\}])l[\![a!\langle k\rangle]\!] \\
N_3 &\Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{a}}[\{\}])l[\![a!\langle k\rangle]\!]
\end{aligned}
$$

Reduction barbed congruence - example

$\Delta_l \models N_i \cong N_j$ where Δ_l - network with one live location l

$$N_1 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{l\}])l[\![a!\langle k\rangle]\!]$$
$$N_2 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{\}])l[\![a!\langle k\rangle]\!]$$
$$N_3 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{a}}[\{\}])l[\![a!\langle k\rangle]\!]$$

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

$\Delta_1 = \Delta_l + k : \mathtt{loc}_{\mathbf{d}}[\{l\}] =$

$\Delta_2 = \Delta_l + k : \mathtt{loc}_{\mathbf{d}}[\{\}] =$

$\Delta_3 = \Delta_l + k : \mathtt{loc}_{\mathbf{a}}[\{\}] =$

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 $\Delta_l \models M_1 \cong M_2$?

Another example

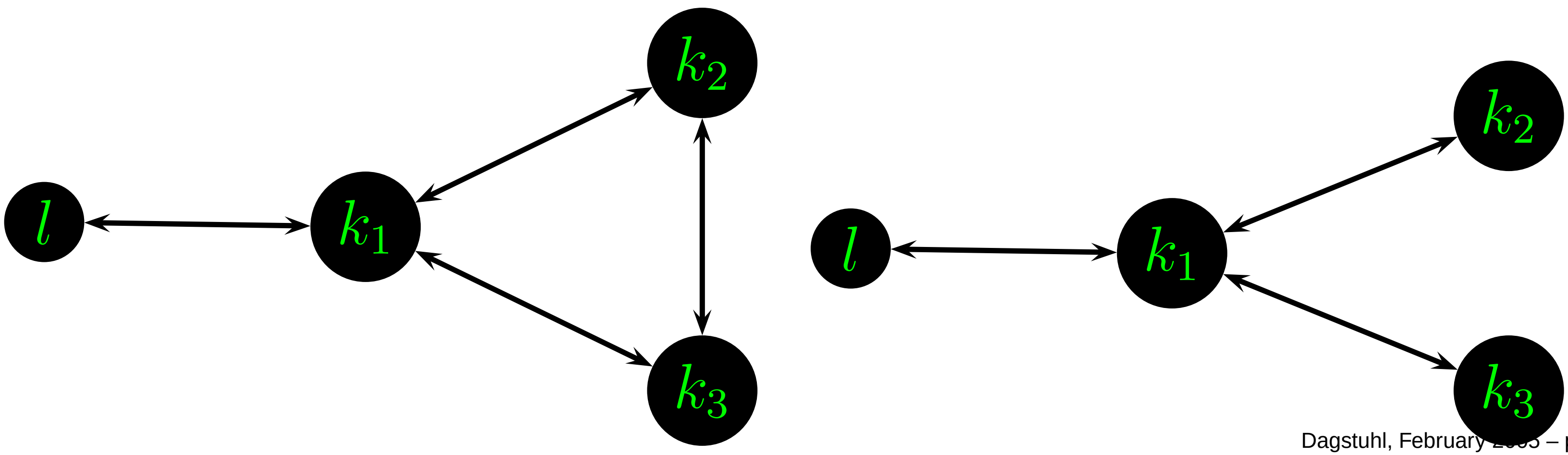
$$M_1 \;\Leftarrow\; (\nu\, k_1 : \{l\})\,(\nu\, k_2 : \{k_1\})\,(\nu\, k_3 : \{k_1, k_2\})\; l[\![a!\langle k_2, k_3\rangle.P]\!]$$

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

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

$$\Delta \triangleright M \xrightarrow{\mu} \Delta' \triangleright M' \qquad \text{is not subtle enough}$$

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

$$\Delta \triangleright M \xrightarrow{\mu} \Delta' \triangleright M' \qquad \text{is not subtle enough}$$

- 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
- Δ must represent *both* actual network, and *observers* knowledge of it

Extended network representations Γ

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

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

Extended network representations Γ

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

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 *indirectly*
- Links to dead nodes are not recorded

Extended network representations Γ

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

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 *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 μ can be

- τ
- $(\tilde{n}:\tilde{\mathtt{T}})l : a?(V)$
  $(\tilde{n})$ are fresh names introduced by observer
  $(\tilde{\mathtt{T}})$ are their state information
- $(\tilde{n}:\tilde{\mathtt{T}})l : a!\langle V\rangle$
  $(\tilde{n})$ are fresh names exported to the observer
  $(\tilde{\mathtt{T}})$ are their state information
- kill(l)
  external location kill
- $l \not\leftrightarrow k$
  external link break

Example rules

- $fail$
$$\frac{\Gamma \vdash l : \mathbf{alive}}{\Pi \triangleright N \xrightarrow{\mathsf{kill}(l)} (\Pi - l) \triangleright N}$$

- $l{-}weak{-}in$
$$\frac{(\Gamma + n : \mathtt{T}) \triangleright N \xrightarrow{\alpha_{in}} \Gamma' \triangleright N'}{\Gamma \triangleright N \xrightarrow{(n:\mathtt{T})\alpha_{in}} \Gamma' \triangleright N'}$$

- $l{-}rest{-}typ$
$$\frac{\Gamma + k : \mathtt{T} \triangleright N \xrightarrow{(\tilde{n}:\tilde{\mathtt{T}})l:a!\langle V\rangle} (\Gamma + \tilde{n} : \tilde{\mathtt{U}}) + k : \mathtt{U} \triangleright N'}{\Gamma \triangleright (\nu\, k : \mathtt{T})N \xrightarrow{(\widetilde{n}:\widetilde{\mathtt{U}})l:a!\langle V\rangle} (\Gamma + \widetilde{n} : \widetilde{\mathtt{U}}) \triangleright (\nu\, k : \mathtt{U})N'} \quad k \text{ in } (\tilde{T})$$

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 $\Gamma \models M \cong N$

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 $\Gamma \models M \cong N$
Proof: Essentially $\approx$ is contextual

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

State information on actions is too detailed.

Counter-example

$$N_1 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{l\}])l[\![a!\langle k\rangle]\!]$$

$$N_2 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{\}])l[\![a!\langle k\rangle]\!]$$

Counter-example

$$N_1 \Leftarrow (\nu\, k : \mathtt{loc_d}[\{l\}])l[\![a!\langle k\rangle]\!]$$

$$N_2 \Leftarrow (\nu\, k : \mathtt{loc_d}[\{\}])l[\![a!\langle k\rangle]\!]$$

$$\Gamma_l \triangleright N_1 \xrightarrow{\alpha_1} \ldots \triangleright$$

$$\Gamma_l \triangleright N_2 \xrightarrow{\alpha_2} \ldots \triangleright$$

where

$$\alpha_1 = (k : \mathtt{loc_d}[\{l\}])l : a!\langle k\rangle$$

$$\alpha_2 = (k : \mathtt{loc_d}[\{\}])l : a!\langle k\rangle$$

Derived actions for DPIF

$$\Gamma \triangleright N \stackrel{\mu}{\longmapsto} \Gamma' \triangleright N'$$

where

- τ
- $(\tilde{n} : \tilde{\mathtt{L}})l : a?(V)$
- $(\tilde{n} : \tilde{\mathtt{L}})l : a!\langle V \rangle K$
- kill(l)
- $l \not\leftrightarrow k$

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

Example

$$N_1 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{l\}])l[\![a!\langle k\rangle]\!]$$

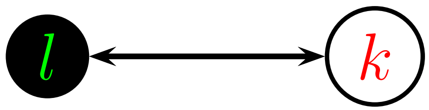
$$N_2 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{\}])l[\![a!\langle k\rangle]\!]$$

$$\Gamma_l \triangleright N_1 \xrightarrow{\alpha_1} \ldots \triangleright \ldots, \qquad \alpha_1 = (k : \mathtt{loc}_{\mathbf{d}}[\{l\}])l : a!\langle k\rangle$$

$$\Gamma_l \triangleright N_2 \xrightarrow{\alpha_2} \ldots \triangleright \ldots, \qquad \alpha_2 = (k : \mathtt{loc}_{\mathbf{d}}[\{\}])l : a!\langle k\rangle$$

Example

$$N_1 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{l\}])l[\![a!\langle k\rangle]\!]$$



$$N_2 \Leftarrow (\nu\, k : \mathtt{loc}_{\mathbf{d}}[\{\}])l[\![a!\langle k\rangle]\!]$$

$$\Gamma_l \triangleright N_1 \xrightarrow{\alpha_1} \ldots \triangleright \ldots, \qquad \alpha_1 = (k : \mathtt{loc}_{\mathbf{d}}[\{l\}])l : a!\langle k\rangle$$

$$\Gamma_l \triangleright N_2 \xrightarrow{\alpha_2} \ldots \triangleright \ldots, \qquad \alpha_2 = (k : \mathtt{loc}_{\mathbf{d}}[\{\}])l : a!\langle k\rangle$$

Derived actions:

$$\Gamma_l \triangleright N_1 \stackrel{\alpha}{\longmapsto} \ldots \triangleright \ldots \qquad \alpha = (k : \{\})l : a!\langle k\rangle$$

$$\Gamma_l \triangleright N_2 \stackrel{\alpha}{\longmapsto} \ldots \triangleright \ldots$$

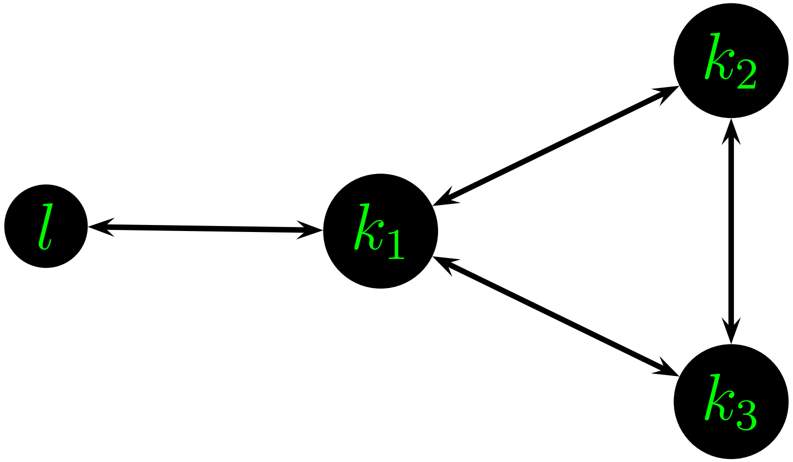
Example revisitied

Is $\Gamma_l \models M_1 \approx M_2$

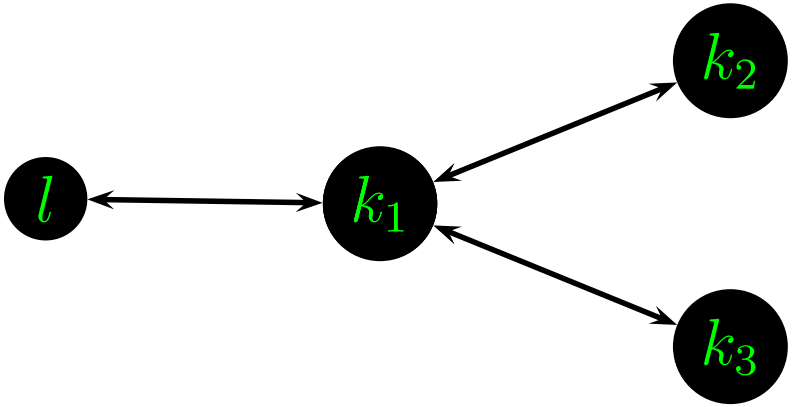
$$M_1 \Leftarrow (\nu\, k_1 : \{l\})\,(\nu\, k_2 : \{k_1\})\,(\nu\, k_3 : \{k_1, k_2\})\, l[\![a!\langle k_2, k_3\rangle.P]\!]$$
$$M_2 \Leftarrow (\nu\, k_1 : \{l\})(\nu\, k_2 : \{k_1\})(\nu\, k_3 : \{k_1\})l[\![a!\langle k_2, k_3\rangle.P]\!]$$

Effective networks of running code:

M_1 :



M_2 :



Full-abstraction

Theorem: In derived lts

$$\Gamma \models M \approx N \text{ if and only if } \Gamma \models M \cong N$$

Full-abstraction

Theorem: In derived lts

$$\Gamma \models M \approx N \text{ if and only if } \Gamma \models M \cong N$$

Proof requires:

- contextuality of $\approx$ in the derived lts
- definability of every derived action
- a formal definition of $\cong$
  We need to be able to compare configuraions 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