Failure and Fault Tolerance in a distributed π-calculus

Imperial College/ Queen Mary, May 2005.

Adrian Francalanza and Matthew Hennessy

Ω{adrianf, matthewh}@sussex.ac.uk.

Overview

- **Motivation:**
  - π-calculus overview
  - A distributed π-calculus
  - Failure in a distributed setting
- Behaviour and Partial Failure: DπF
- Dependability: Fault Tolerance

Starting Point: π-calculus Syntax

P, Q	::=	**stop**	(*inaction*)
	|	$a!\langle v\rangle.P$	(*output*)
	|	$a?(x).P$	(*input*)
	|	$*P$	(*replication*)
	|	if $v = u.P\lceil Q\rceil$	(*name matching*)
	|	$P\|Q$	(*parallel composition*)
	|	$(va)P$	(*hiding*)

π-calculus Reduction

(r-comm)

$$\frac{}{a!\langle b\rangle.P \mid a?(x).Q \longrightarrow P \mid Q\{^{b}/_{x}\}}$$

(r-par)

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{Q \mid P \longrightarrow Q \mid P'}$$

(r-ncond)

$$\frac{}{\text{if } a=b.P\lceil Q\rceil \longrightarrow Q}\ a \neq b$$

(r-cond)

$$\frac{}{\text{if } a=a.P\lceil Q\rceil \longrightarrow P}$$

(r-rest)

$$\frac{P \longrightarrow P'}{(\nu a)P \longrightarrow (\nu a)P'}$$

π-calculus Reduction

(r-comm)

$$\frac{}{a!\langle b\rangle.P \mid a?(x).Q \longrightarrow P \mid Q\{^{b}/_{x}\}}$$

(r-par)

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{Q \mid P \longrightarrow Q \mid P'}$$

(r-ncond)

$$\frac{}{\text{if } a=b.P\lceil Q\rceil \longrightarrow Q}\ a \neq b$$

(r-cond)

$$\frac{}{\text{if } a=a.P\lceil Q\rceil \longrightarrow P}$$

(r-rest)

$$\frac{P \longrightarrow P'}{(\nu a)P \longrightarrow (\nu a)P'}$$

π-calculus Reduction

(r-comm)

$$\frac{}{a!\langle b\rangle.P \mid a?(x).Q \longrightarrow P \mid Q\{^{b}/_{x}\}}$$

(r-par)

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q \qquad Q \mid P \longrightarrow Q \mid P'}$$

(r-ncond)

$$\frac{}{\text{if } a=b.P\lceil Q\rceil \longrightarrow Q}\ a \neq b$$

(r-cond)

$$\frac{}{\text{if } a=a.P\lceil Q\rceil \longrightarrow P}$$

(r-rest)

$$\frac{P \longrightarrow P'}{(\nu a)P \longrightarrow (\nu a)P'}$$

π-calculus Reduction

(r-comm)

$$\frac{}{a!\langle b\rangle.P \mid a?(x).Q \longrightarrow P \mid Q\{^{b}/_{x}\}}$$

(r-par)

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad Q \mid P \longrightarrow Q \mid P'$$

(r-ncond)

$$\frac{}{\text{if } a=b.P\lceil Q\rceil \longrightarrow Q}\ a \neq b$$

(r-cond)

$$\frac{}{\text{if } a=a.P\lceil Q\rceil \longrightarrow P}$$

(r-rest)

$$\frac{P \longrightarrow P'}{(\nu a)P \longrightarrow (\nu a)P'}$$

π-calculus Reduction(2)

(s-par)

$$P \mid Q \equiv Q \mid P$$

(s-inact)

$$P \mid \mathbf{stop} \equiv P$$

(s-repl)

$$*P \equiv *P \mid P$$

(s-rest)

$$P \mid (\nu a)Q \equiv (\nu a)(P \mid Q) \quad a \notin \mathbf{fn}(P)$$

(r-struct)

$$\frac{P \equiv P' \qquad P' \longrightarrow Q' \qquad Q' \equiv Q}{P \longrightarrow Q}$$

π-calculus Reduction(2)

(s-par)

$$P \mid Q \equiv Q \mid P$$

(s-inact)

$$P \mid \mathbf{stop} \equiv P$$

(s-repl)

$$*P \equiv *P \mid P$$

(s-rest)

$$P \mid (\nu a)Q \equiv (\nu a)(P \mid Q) \quad a \notin \mathbf{fn}(P)$$

(r-struct)

$$\frac{P \equiv P' \qquad P' \longrightarrow Q' \qquad Q' \equiv Q}{P \longrightarrow Q}$$

π-calculus Reduction(2)

(s-par)

$$\frac{}{P \mid Q \equiv Q \mid P}$$

(s-inact)

$$\frac{}{P \mid \mathbf{stop} \equiv P}$$

(s-repl)

$$\frac{}{*P \equiv *P \mid P}$$

(s-rest)

$$\frac{}{P \mid (\nu a)Q \equiv (\nu a)(P \mid Q)} \; a \notin \mathbf{fn}(P)$$

(r-struct)

$$\frac{P \equiv P' \qquad P' \longrightarrow Q' \qquad Q' \equiv Q}{P \longrightarrow Q}$$

π-calculus Reduction(2)

(s-par)

$$P \mid Q \equiv Q \mid P$$

(s-inact)

$$P \mid \mathbf{stop} \equiv P$$

(s-repl)

$$*P \equiv *P \mid P$$

(s-rest)

$$P \mid (\nu a)Q \equiv (\nu a)(P \mid Q) \quad a \notin \mathbf{fn}(P)$$

(r-struct)

$$\frac{P \equiv P' \qquad P' \longrightarrow Q' \qquad Q' \equiv Q}{P \longrightarrow Q}$$

π-calculus example: Client Server

$$\text{server} \Leftarrow (\nu\, data)\left(\begin{array}{l} req?(x_1, x_2).data!\langle x_1, x_2\rangle \\ \\ \mid data?(y_1, y_2).y_2!\langle\, \textbf{lookup}(y_1)\rangle \end{array}\right)$$

π-calculus example: Client Server

$$\text{server} \Leftarrow (\nu\, data)\left(\begin{array}{l} req?(x_1, x_2).data!\langle x_1, x_2\rangle \\ \\ \mid data?(y_1, y_2).y_2!\langle\, \textbf{lookup}(y_1)\rangle \end{array}\right)$$

$$\text{client} \Leftarrow req!\langle Adrian, ans\rangle \mid ans?(x)...$$

π-calculus example: Client Server

$$\text{server} \Leftarrow (\nu\, \mathit{data}) \left(\begin{array}{l} \mathit{req}?(x_1, x_2).\mathit{data}!\langle x_1, x_2 \rangle \\ \\ \mid \mathit{data}?(y_1, y_2).y_2!\langle\, \mathbf{lookup}(y_1) \rangle \end{array} \right)$$

$$\text{client} \Leftarrow \mathit{req}!\langle \mathit{Adrian}, \mathit{ans} \rangle \mid \mathit{ans}?(x)...$$

π-calculus example: Client Server

$$\text{server} \Leftarrow (\nu\, \mathit{data}) \left(\begin{array}{l} \mathit{data}!\langle \mathit{Adrian}, \mathit{ans} \rangle \\ \\ \mid \mathit{data}?(y_1, y_2).y_2!\langle\, \mathbf{lookup}(y_1) \rangle \end{array} \right)$$

$$\text{client} \Leftarrow \mathit{ans}?(x)...$$

π-calculus example: Client Server

$$\text{server} \Leftarrow (\nu\, data)\left(\begin{array}{l} data!\langle Adrian, ans\rangle \\ \\ \mid data?(y_1, y_2).y_2!\langle \textbf{lookup}(y_1)\rangle \end{array}\right)$$

$$\text{client} \Leftarrow ans?(x)...$$

π-calculus example: Client Server

$$\text{server} \Leftarrow (\nu\, \mathit{data})\left(\quad \mathit{ans}!\langle\, \textbf{lookup}(\mathit{Adrian})\rangle \quad \right)$$

$$\text{client} \Leftarrow \mathit{ans}?(x)...$$

π-calculus example: Client Server

$$\text{server} \Leftarrow (\nu\, \mathit{data})\left(\quad \mathit{ans}!\langle\, \textbf{lookup}(\mathit{Adrian})\rangle \right)$$

$$\text{client} \Leftarrow \mathit{ans}?(x)...$$

$D\pi$ Sytnax

Processes

$$P, Q \quad ::= \quad \mathbf{stop} \quad a!\langle v\rangle.P \quad | \; a?(x).P \quad | \; *P$$
$$\quad | \quad \text{if } v = u.P\lceil Q\rceil \quad | \; P|Q \quad | \; (\nu a)P$$
$$\quad | \quad \text{go } l.P \quad (\textit{migration})$$

Systems

$$N, M \quad ::= \quad l[\![P]\!] \quad (\textit{located process})$$
$$\quad | \quad (\nu\, n)N \quad (\textit{network scoping})$$
$$\quad | \quad N|M \quad (\textit{parallel systems})$$

$D\pi$: Reduction Rules

(r-comm)

$$l[\![a!\langle n\rangle.P]\!] \mid l[\![a?(x).Q]\!] \longrightarrow l[\![P]\!] \mid l[\![Q\{^{n}/_{x}\}]\!]$$

(r-go)

$$l[\![\text{go } k.P]\!] \longrightarrow k[\![P]\!]$$

$D\pi$: Reduction Rules

(r-comm)

$$l[\![a!\langle n\rangle.P]\!] \mid l[\![a?(x).Q]\!] \longrightarrow l[\![P]\!] \mid l[\![Q\{^{n}/_{x}\}]\!]$$

(r-go)

$$l[\![\text{go } k.P]\!] \longrightarrow k[\![P]\!]$$

$D\pi$: Reduction Rules

(r-comm)

$$l[\![a!\langle n\rangle.P]\!] \mid l[\![a?(x).Q]\!] \longrightarrow l[\![P]\!] \mid l[\![Q\{^{n}/_{x}\}]\!]$$

(r-go)

$$l[\![\text{go } k.P]\!] \longrightarrow k[\![P]\!]$$

$D\pi$ Example: Distributed Client Server

$$\text{server} \Leftarrow (\nu\, \mathit{data}) \left(\begin{array}{l} l[\![\mathit{req}?(x_1, x_2).\mathit{data}!\langle x_1, x_2\rangle]\!] \\ \\ \mid l[\![\mathit{data}?(y_1, y_2).y_2!\langle \mathbf{lookup}(y_1)\rangle]\!] \end{array} \right)$$

$$\text{serverLoc} \Leftarrow (\nu\, \mathit{data}) \left(\begin{array}{l} l[\![\mathit{req}?(x_1, x_2).\text{go}\, k_1.\mathit{data}!\langle x_1, x_2\rangle]\!] \\ \\ \mid k_1[\![\mathit{data}?(y_1, y_2).\text{go}\, l.y_2!\langle \mathbf{lookup}(y_1)\rangle]\!] \end{array} \right)$$

$D\pi$ Example: Distributed Client Server

$$\text{server} \Leftarrow (\nu\, data)\left(\begin{array}{l} l[\![req?(x_1, x_2).data!\langle x_1, x_2\rangle]\!] \\ \\ \mid l[\![data?(y_1, y_2).y_2!\langle \mathbf{lookup}(y_1)\rangle]\!] \end{array}\right)$$

$$\text{serverLoc} \Leftarrow (\nu\, data)\left(\begin{array}{l} l[\![req?(x_1, x_2).\text{go}\, k_1.data!\langle x_1, x_2\rangle]\!] \\ \\ \mid k_1[\![data?(y_1, y_2).\text{go}\, l.y_2!\langle \mathbf{lookup}(y_1)\rangle]\!] \end{array}\right)$$

$D\pi$ Example: Distributed Client Server

$$\text{serverLnk} \Leftarrow (\nu\, data)\left(\begin{array}{l} l\left[\!\left[req?(x_1, x_2).(\nu sync)\left(\begin{array}{l} \text{go } k_1.data!\langle x_1, sync\rangle \\ \mid \text{go } k_2, k_1.data!\langle x_1, sync\rangle \\ \mid sync?(z).x_2!\langle z\rangle \end{array}\right)\right]\!\right] \\ \mid k_1\left[\!\left[data?(y_1, y_2).\left(\begin{array}{l} \text{go } l.y_2!\langle \textbf{lookup}(y_1)\rangle \\ \mid \text{go } k_2, l.y_2!\langle \textbf{lookup}(y_1)\rangle \end{array}\right)\right]\!\right] \end{array}\right)$$

$D\pi$ Example: Distributed Client Server

$$\text{serverLnk} \Leftarrow (\nu\, data)\left(\begin{array}{l} l[\![\, req?(x_1, x_2).(\nu sync)\left(\begin{array}{l} \text{go}\, k_1.data!\langle x_1, sync\rangle \\ \mid \text{go}\, k_2,\ k_1.data!\langle x_1, sync\rangle \\ \mid sync?(z).x_2!\langle z\rangle \end{array}\right)\,]\!] \\ \\ \mid k_1[\![\, data?(y_1, y_2).\left(\begin{array}{l} \text{go}\, l.y_2!\langle \textbf{lookup}(y_1)\rangle \\ \mid \text{go}\, k_2,\ l.y_2!\langle \textbf{lookup}(y_1)\rangle \end{array}\right)\,]\!] \end{array}\right)$$

Reduction Closed Relations

$$M \quad \mathcal{R} \quad N$$

Reduction Closed Relations

$$M \quad \mathcal{R} \quad N$$
$$\downarrow$$
$$M'$$

Reduction Closed Relations

$$\begin{array}{ccc} M & \mathcal{R} & N \\ \downarrow & & \Downarrow \\ M' & \mathcal{R} & N' \end{array}$$

$D\pi$ Equivalence

Barbs

$M \Downarrow_{a@l}$ iff $M \Longrightarrow \equiv (\nu \tilde{n}) M' \mid l[\![a!\langle n \rangle.Q]\!]$ where $l, a \notin \tilde{n}$

$D\pi$ Equivalence

Barbs

$M \Downarrow_{a@l}$ iff $M \Longrightarrow \equiv (\nu \tilde{n})M' \mid l[\![a!\langle n \rangle.Q]\!]$ where $l, a \notin \tilde{n}$

Barb Preserving Relation

$M \mathcal{R} N$ and $M \Downarrow_{a@l}$ implies $N \Downarrow_{a@l}$

Contextual Relation

$M \mathcal{R} N$ implies $\forall O$ we infer $M \mid O \ \mathcal{R} \ N \mid O$

and $O \mid M \ \mathcal{R} \ O \mid N$

$D\pi$ Equivalence

Reduction Barbed Congruence

The *largest* relation over *systems*, denoted by $\cong$, that is:

- reduction closed
- barb preserving
- contextual

$D\pi$ Equivalence: Application

According to this equivalence

$$\text{server} \cong \text{serverLoc} \cong \text{serverLnk}$$

$D\pi$ Equivalence: Application

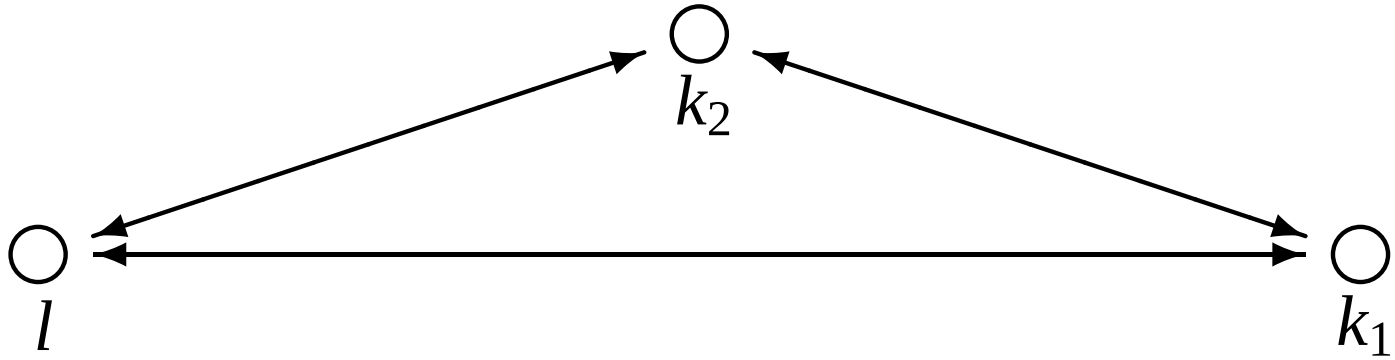
According to this equivalence

$$\text{server} \cong \text{serverLoc} \cong \text{serverLnk}$$

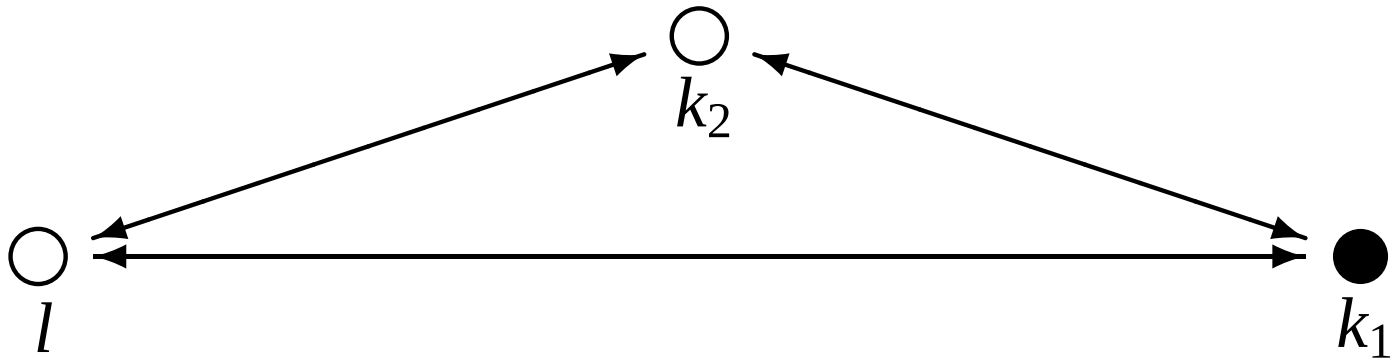
This only holds if :

- no node fails
- every node is interconnected (clique)

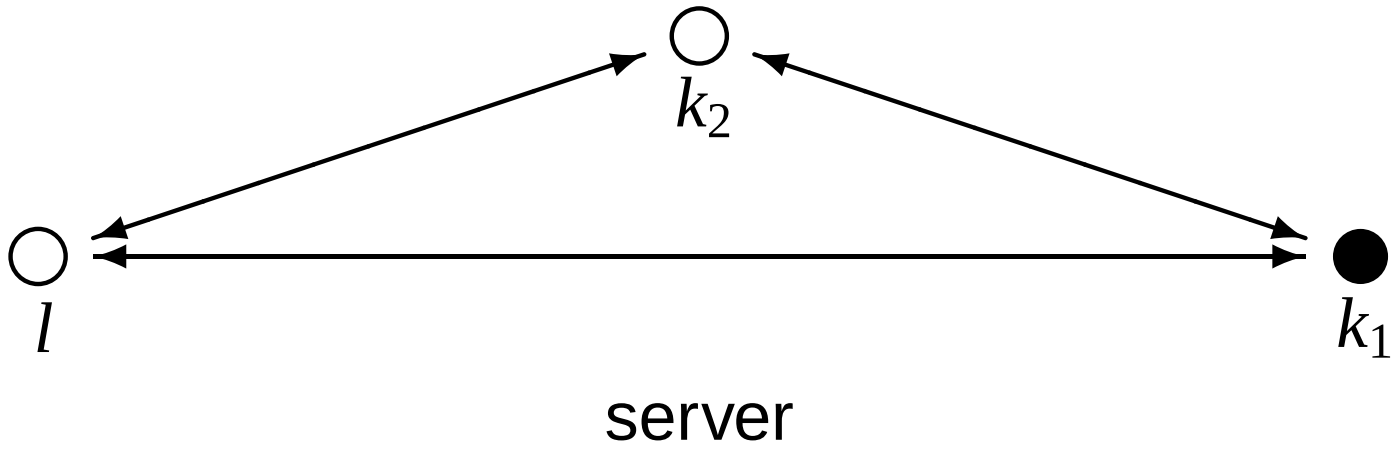
The Underlying Network



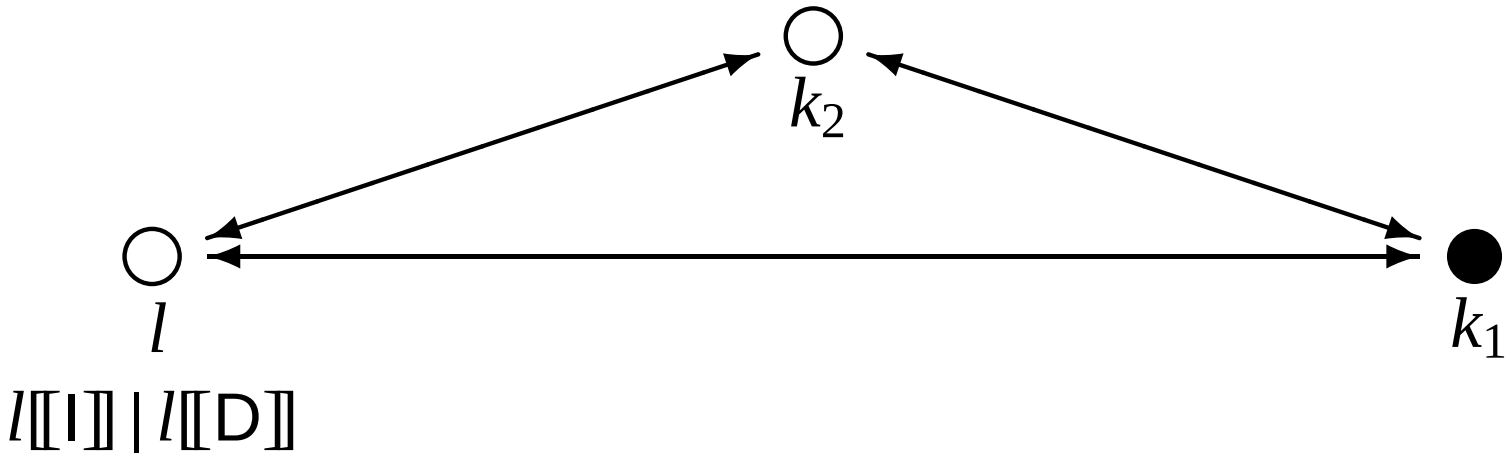
The Underlying Network



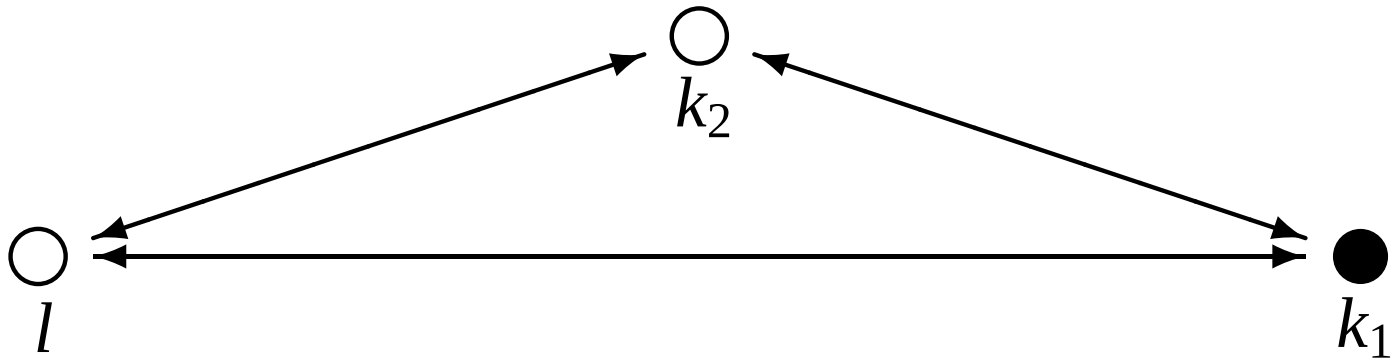
The Underlying Network



The Underlying Network

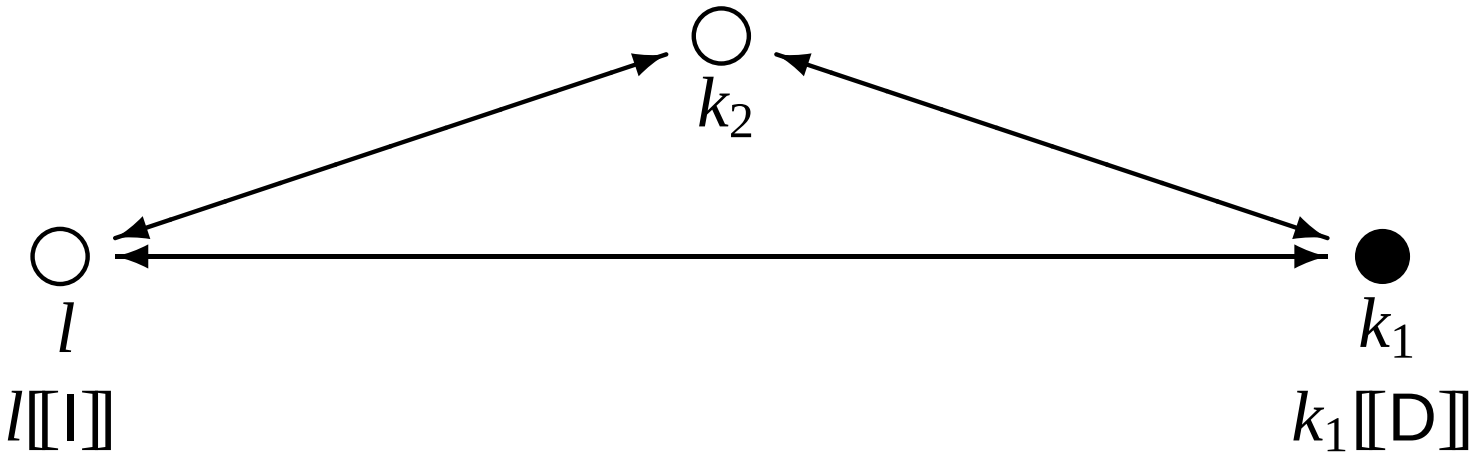


The Underlying Network

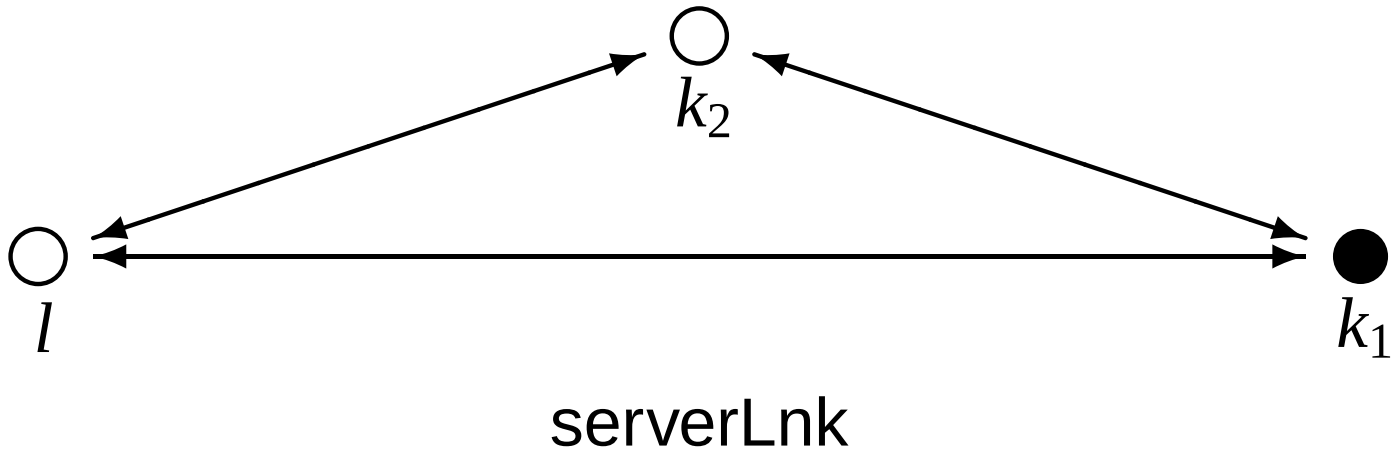


serverLoc

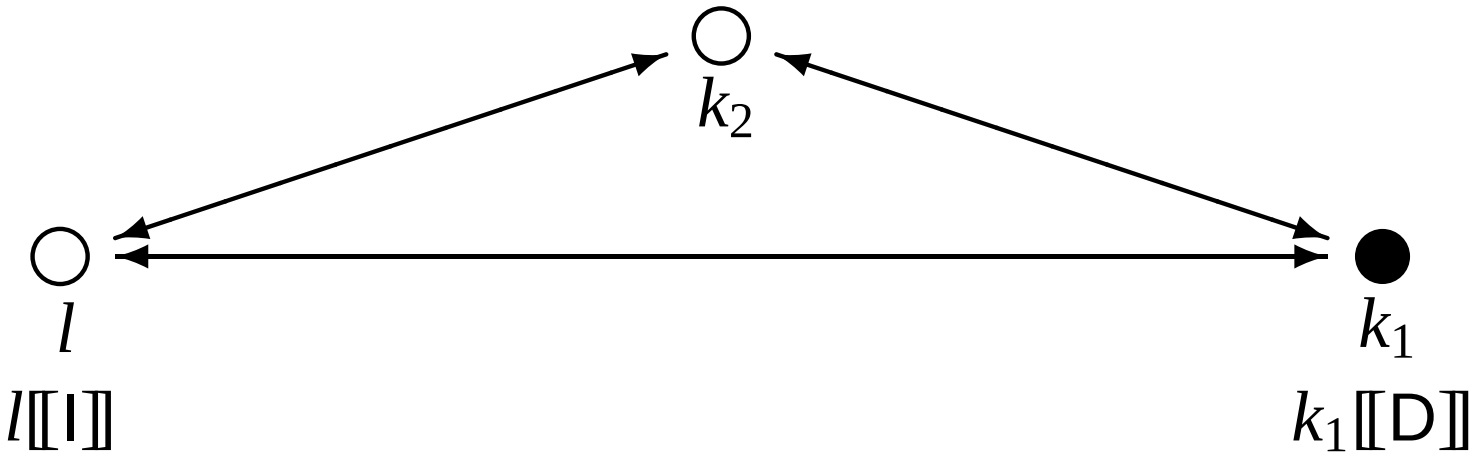
The Underlying Network



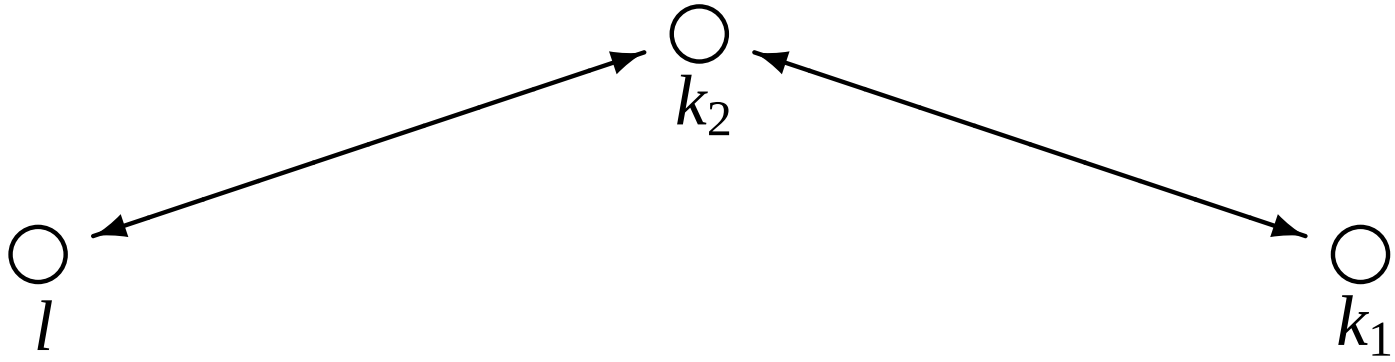
The Underlying Network



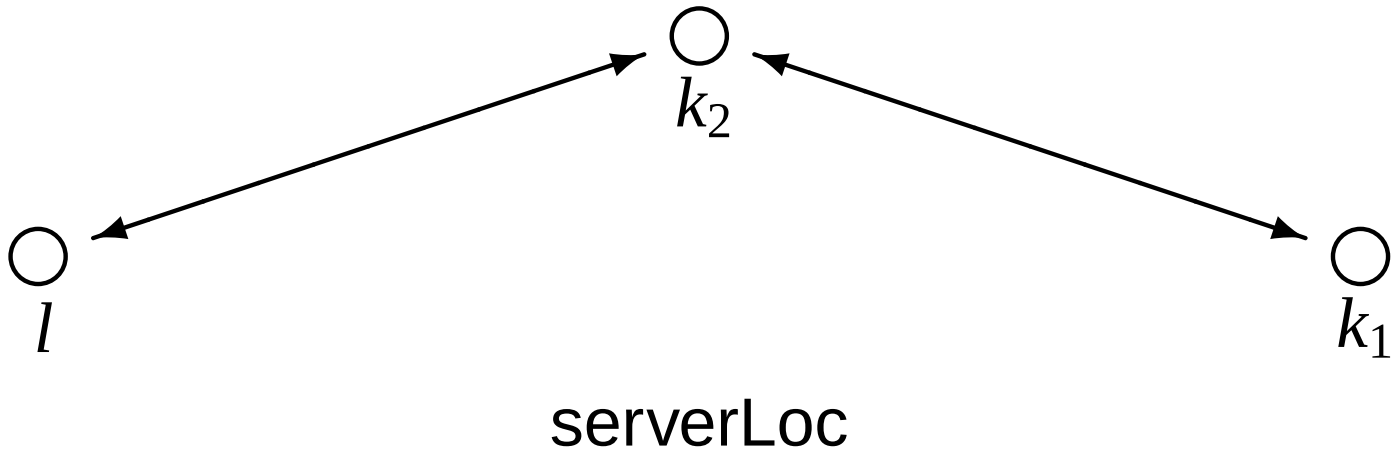
The Underlying Network



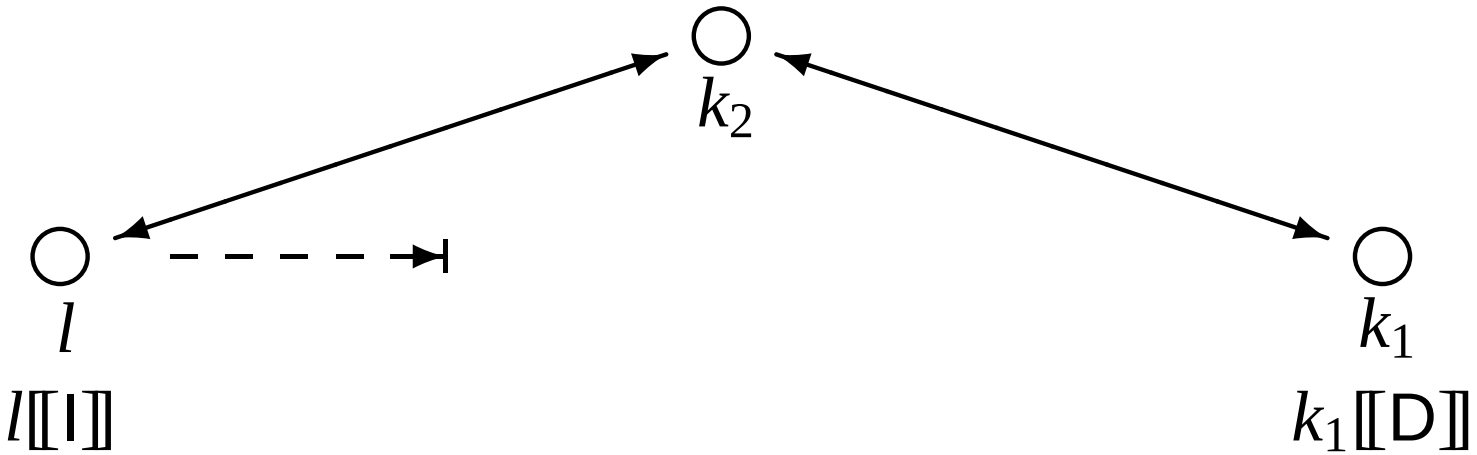
The Underlying Network



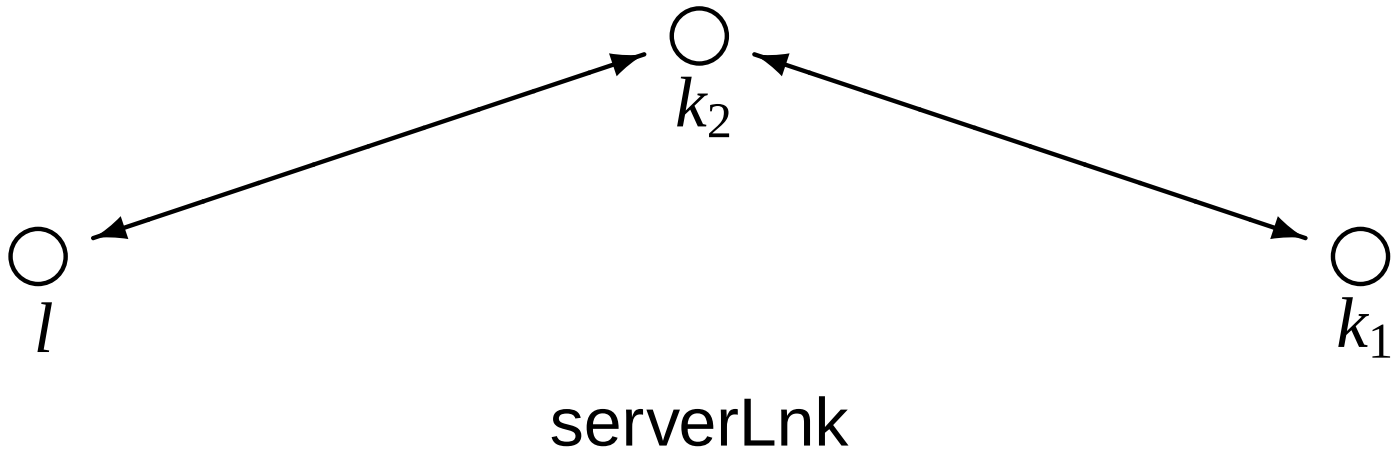
The Underlying Network



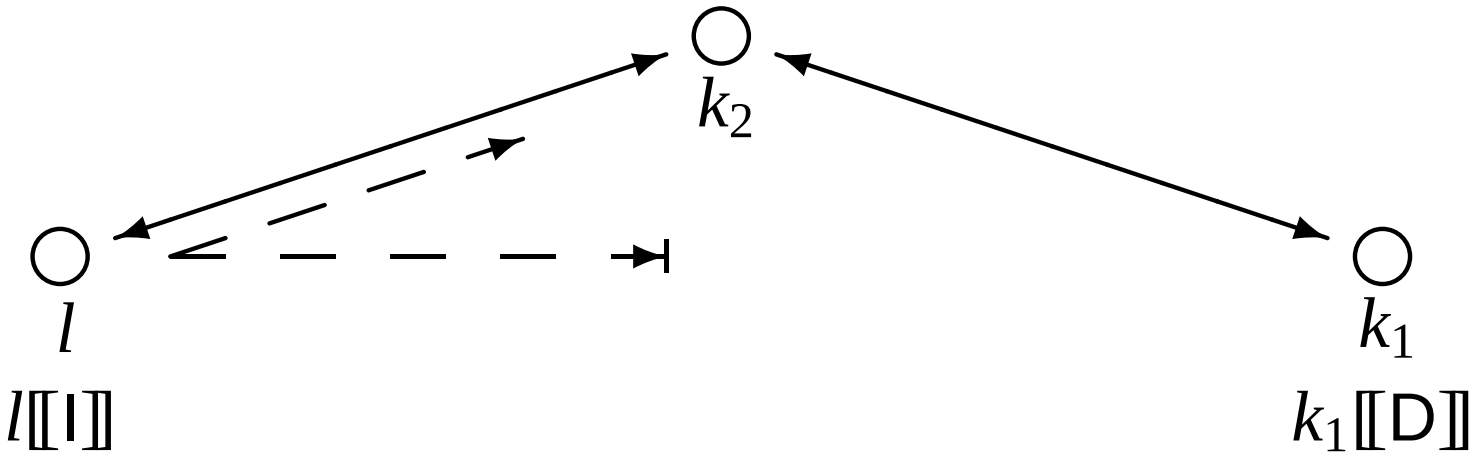
The Underlying Network



The Underlying Network



The Underlying Network



$D\pi$ Scope Extrusion Example

$$\mathsf{M} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![c!\langle\rangle]\!])$$

$D\pi$ Scope Extrusion Example

$$\mathsf{M} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![c!\langle\rangle]\!])$$

$$\mathsf{M} \not\cong \mathsf{N}$$

$D\pi$ Scope Extrusion Example

$$\mathsf{M} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![c!\langle\rangle]\!])$$

$$\mathsf{M} \not\cong \mathsf{N}$$

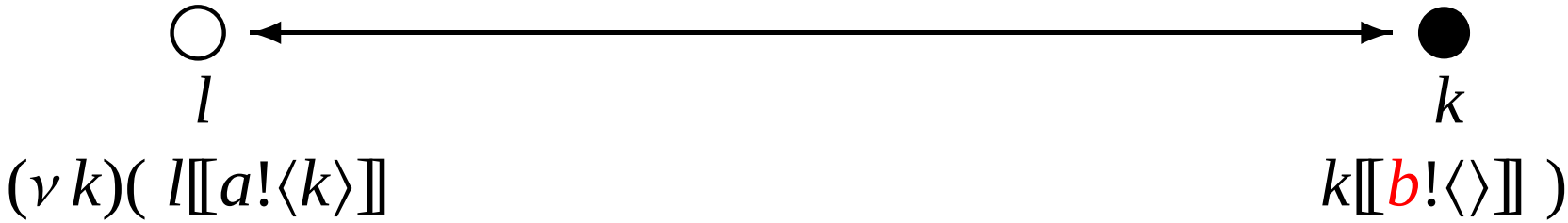
$$\mathsf{O} \Leftarrow l[\![a?(x).\mathsf{go}\ x.b?().\mathsf{go}\ l.\mathsf{ok}!\langle\rangle]\!]$$

$$\mathsf{M}|\mathsf{O} \Downarrow_{\mathsf{ok}@l}$$

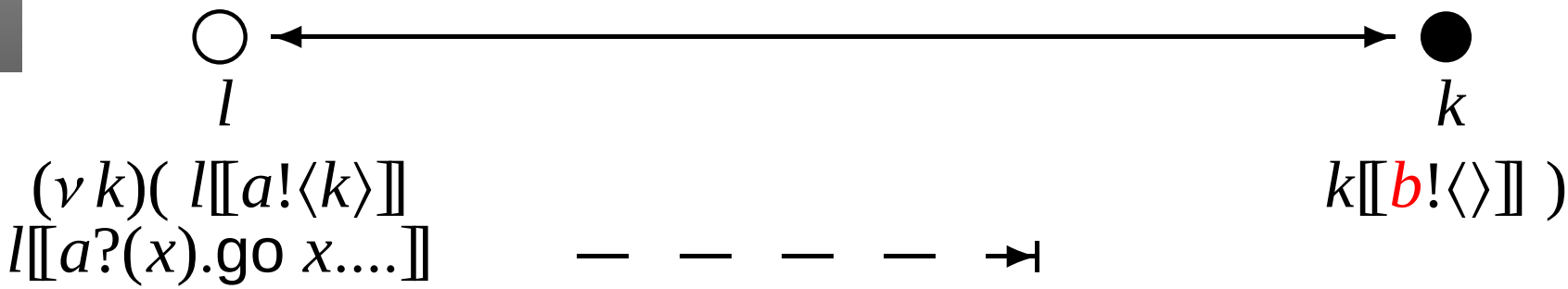
$$\mathsf{N}|\mathsf{O} \not\Downarrow_{\mathsf{ok}@l}$$

$D\pi$ Scope Extrusion Example

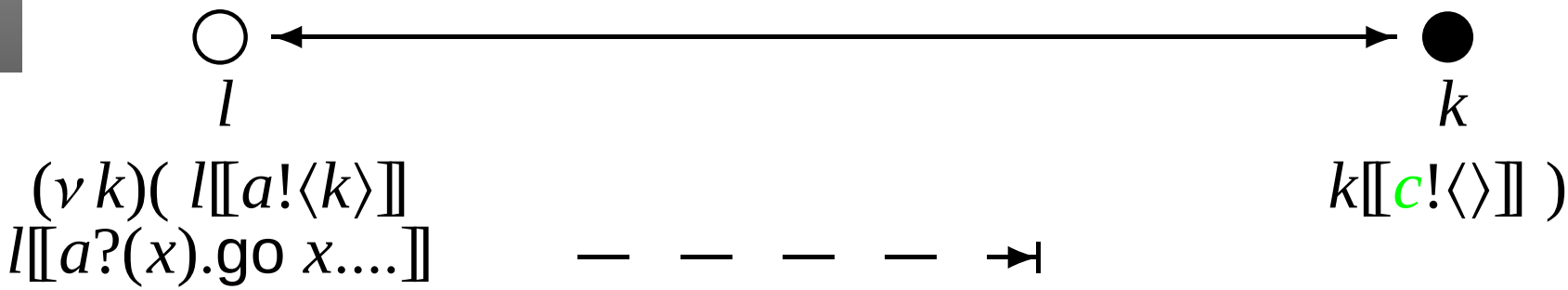
$D\pi$ Scope Extrusion Example



$D\pi$ Scope Extrusion Example

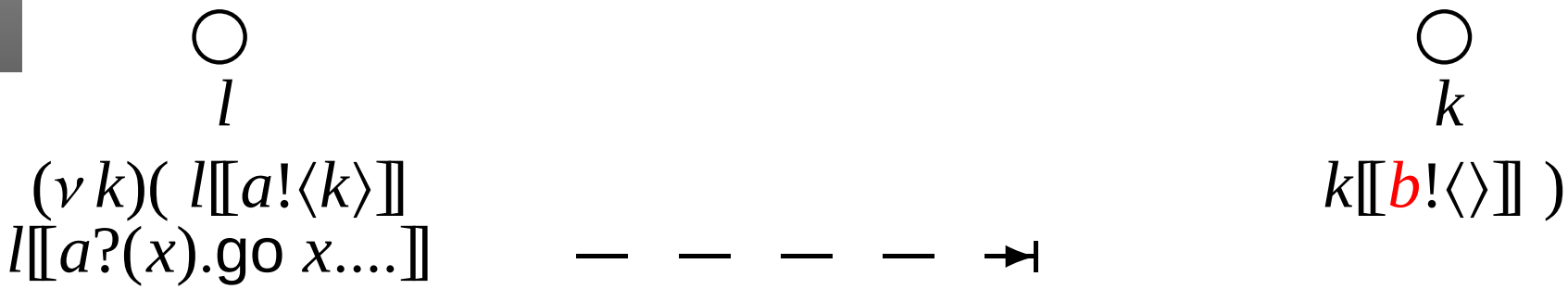


$D\pi$ Scope Extrusion Example



$D\pi$ Scope Extrusion Example

$D\pi$ Scope Extrusion Example



$D\pi$ Scope Extrusion Example

$D\pi$ Scope Extrusion Example

$$\mathsf{M} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k)(l[\![a!\langle k\rangle]\!] \mid k[\![c!\langle\rangle]\!])$$

$$\mathsf{O} \Leftarrow l[\![a?(x).\mathsf{go}\ x.b?().\mathsf{go}\ l.\mathsf{ok}!\langle\rangle]\!]$$

$$\mathsf{M}|\mathsf{O}\Downarrow\!\!\!\!\!\!\not\quad_{\mathsf{ok}@l}$$
$$\mathsf{N}|\mathsf{O}\Downarrow\!\!\!\!\!\!\not\quad_{\mathsf{ok}@l}$$

$\mathsf{M} \cong \mathsf{N}$ for certain networks

Overview

- Motivation:
  - π-calculus overview
  - A distributed π-calculus
  - Failure in a distributed setting
- **Behaviour and Partial Failure: DπF**
- Dependability: Fault Tolerance

$D\pi F$: $D\pi$ with failure

Types

$$\begin{array}{rcll} \mathtt{T}, \mathtt{U} & ::= & \mathtt{loc}[\mathtt{A}, \{l_1, .., l_n\}] & |\,\mathtt{ch} \\ \mathtt{A} & ::= & \mathtt{a} \,|\, \mathtt{d} & \end{array}$$

Processes

$$\begin{array}{rcll} P, Q & ::= & \ldots & \\ & | & (\nu\, n : \mathtt{T})P & (\textit{typed scoping}) \\ & | & \mathsf{kill} & (\textit{kill location}) \\ & | & \mathsf{break}\ l & (\textit{break link}) \\ & | & \mathsf{ping}\ l.P\lceil Q \rceil & (\textit{ping}) \end{array}$$

$D\pi F$: $D\pi$ with failure

Systems

$$N, M \quad ::= \quad \ldots \quad | (\nu\, n : \mathtt{T}) N$$

We work at the level of **Configurations**

$$\Gamma \triangleright M$$

where Γ is a network representation

$D\pi F$: $D\pi$ with failure

(r-comm)

$$\frac{\Gamma \vdash l : \textbf{alive}}{\Gamma \triangleright l[\![a!\langle n\rangle.P]\!] \mid l[\![a?(x).Q]\!] \longrightarrow \Gamma \triangleright l[\![P]\!] \mid l[\![Q\{^{n}/_{x}\}]\!]}$$

(r-go)

$$\frac{\Gamma \vdash l : \textbf{alive} \qquad \Gamma \vdash k : \textbf{alive} \qquad \Gamma \vdash l \leftrightarrow k}{\Gamma \triangleright l[\![\text{go } k.P]\!] \longrightarrow \Gamma \triangleright k[\![P]\!]}$$

$D\pi F$: $D\pi$ with failure

(r-kill)

$$\frac{\Gamma \vdash l : \mathbf{alive}}{\Gamma \triangleright l[\![\mathsf{kill}]\!] \longrightarrow \Gamma - l \triangleright l[\![\mathbf{stop}]\!]}$$

(r-brk)

$$\frac{\Gamma \vdash l : \mathbf{alive} \qquad \Gamma \vdash l \leftrightarrow k}{\Gamma \triangleright l[\![\mathsf{break}\ k]\!] \longrightarrow \Gamma - l \leftrightarrow k \triangleright l[\![\mathbf{stop}]\!]}$$

$D\pi F$: $D\pi$ with failure

(r-ping)

$$\frac{\Gamma \vdash l : \mathbf{alive} \qquad \Gamma \vdash k : \mathbf{alive} \qquad \Gamma \vdash l \leftrightarrow k}{\Gamma \triangleright l[\![\text{ping } k.P\lceil Q\rceil]\!] \longrightarrow \Gamma \triangleright l[\![P]\!]}$$

(r-nping1)

$$\frac{\Gamma \vdash l : \mathbf{alive} \qquad \Gamma \vdash k : \mathbf{dead}}{\Gamma \triangleright l[\![\text{ping } k.P\lceil Q\rceil]\!] \longrightarrow \Gamma \triangleright l[\![Q]\!]}$$

(r-nping2)

$$\frac{\Gamma \vdash l : \mathbf{alive} \qquad \Gamma \vdash l \nleftrightarrow k}{\Gamma \triangleright l[\![\text{ping } k.P\lceil Q\rceil]\!] \longrightarrow \Gamma \triangleright l[\![Q]\!]}$$

DπF Example

$$\mathsf{M} \quad \Leftarrow \quad (\nu\, k : \mathtt{loc[a, \{}l\mathtt{\}]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

If $\Gamma \vdash l$: **alive**

DπF Example

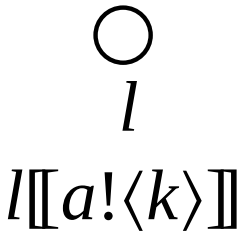
$$\mathsf{M} \quad \Leftarrow \quad (\nu\, k : \mathtt{loc[d, \{}l\mathtt{\}]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

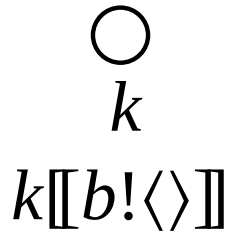
If $\Gamma \vdash l$: **alive**

DπF Example

$$\mathsf{M} \quad \Leftarrow \quad (\nu\, k : \mathtt{loc[a, \emptyset]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

If $\Gamma \vdash l$: **alive**





Barbs

$\Gamma \triangleright M{\Downarrow_{a@l}}$ iff $M \Longrightarrow \equiv (\nu \tilde{n})M' \mid l[\![a!\langle n \rangle.Q]\!]$
where $l, a \notin \tilde{n}$ and $\Gamma \vdash l$: **alive**

Contextual Relation

$\Gamma \models M \ \mathcal{R} \ N$ implies $\forall O$ we infer $\Gamma \models M \mid O \ \mathcal{R} \ N \mid O$

and $\Gamma \models O \mid M \ \mathcal{R} \ O \mid N$

and $\Gamma + n : \mathtt{T} \models M \ \mathcal{R} \ N$

Reduction Barbed Congruence

The *largest* relation over *systems*, running over the *same network* denoted by $\Gamma \models M \cong N$, that is:

- reduction closed
- barb preserving
- contextual

DπF bisimulation

DπF lts over configurations

$$\Gamma \triangleright M \xrightarrow{\mu} \Gamma' \triangleright M'$$

where μ is either:

- τ (internal action)
- $(\tilde{n} : \tilde{\mathrm{T}})l : a!\langle V \rangle$ (bound output)
- $(\tilde{n} : \tilde{\mathrm{T}})l : a?(V)$ (bound input)
- kill : k (external kill)
- $l \leftrightarrow k$ (external break)

DπF bisimulation

(l-out)

$$\frac{\Gamma \vdash l : \mathbf{alive}}{\Gamma \triangleright l[\![a!\langle V\rangle.P]\!] \xrightarrow{l:a!\langle V\rangle} \Gamma \triangleright l[\![P]\!]}$$

(l-open)

$$\frac{\Gamma + n:\mathtt{T} \triangleright M \xrightarrow{(\tilde{}n\,\tilde{\mathtt{T}})l:a!\langle V\rangle} \Gamma' \triangleright M'}{\Gamma \triangleright (\nu\, n:\mathtt{T})M \xrightarrow{(n:\mathtt{T},\tilde{}n\,\tilde{\mathtt{T}})l:a!\langle V\rangle} \Gamma' \triangleright M'}\; l, a \neq n$$

$D\pi F$ bisimulation

(r-go)

$$\frac{\Gamma \vdash l : \textbf{alive} \qquad \Gamma \vdash k : \textbf{alive} \qquad \Gamma \vdash l \leftrightarrow k}{\Gamma \triangleright l[\![\text{go } k.P]\!] \xrightarrow{\tau} \Gamma \triangleright k[\![P]\!]}$$

(r-kill)

$$\frac{\Gamma \vdash l : \textbf{alive}}{\Gamma \triangleright M \xrightarrow{\text{kill}:l} \Gamma - l \triangleright M}$$

$D\pi F$ bisimulation

Weak bisimulation equivalence, denoted as $\approx_{bad}$, is the *largest symmetric* relation over *configurations* such that whenever $\Gamma_1 \triangleright M_1 \approx \Gamma_2 \triangleright M_2$ then:

- $\Gamma_1 \triangleright M_1 \xrightarrow{\mu} \Gamma_1' \triangleright M_1'$ implies $\Gamma_2 \triangleright M_2 \stackrel{\hat{\mu}}{\Longrightarrow} \Gamma_2' \triangleright M_2'$ such that $\Gamma_1' \triangleright M_1' \approx \Gamma_2' \triangleright M_2'$
- $\Gamma_2 \triangleright M_2 \xrightarrow{\mu} \Gamma_2' \triangleright M_2'$ implies $\Gamma_1 \triangleright M_1 \stackrel{\hat{\mu}}{\Longrightarrow} \Gamma_1' \triangleright M_1'$ such that $\Gamma_1' \triangleright M_1' \approx \Gamma_2' \triangleright M_2'$

$D\pi F$ bisimulation

But $\approx_{\text{bad}}$ is **too intensional!**

$$\mathsf{M} \Leftarrow (\nu\, k : \mathtt{loc}[\mathtt{d}, \{l\}])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k : \mathtt{loc}[\mathtt{a}, \emptyset])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$D\pi F$ bisimulation

But $\approx_{\text{bad}}$ is **too intensional!**

$$\mathsf{M} \Leftarrow (\nu\, k : \mathtt{loc}[\mathsf{d}, \{l\}])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

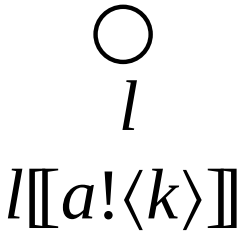
$$\mathsf{N} \Leftarrow (\nu\, k : \mathtt{loc}[\mathsf{a}, \emptyset])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

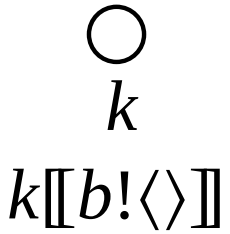
$D\pi F$ bisimulation

But $\approx_{\text{bad}}$ is **too intensional!**

$$\mathsf{M} \Leftarrow (\nu\, k : \mathtt{loc}[\mathsf{d}, \{l\}])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k : \mathtt{loc}[\mathsf{a}, \emptyset])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$





$D\pi F$ bisimulation

But $\approx_{\text{bad}}$ is **too intensional!**

$$\mathsf{M} \quad \Leftarrow \quad (\nu\, k : \mathtt{loc}[\mathtt{d}, \{l\}])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \quad \Leftarrow \quad (\nu\, k : \mathtt{loc}[\mathtt{a}, \emptyset])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

For $\Gamma \vdash l$: **alive**

$$\Gamma \models \mathsf{M} \;\cong\; \mathsf{N}$$

$D\pi F$ bisimulation

But $\approx_{\mathsf{bad}}$ is **too intensional!**

$$\mathsf{M} \Leftarrow (\nu\, k : \mathtt{loc}[\mathtt{d}, \{l\}])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k : \mathtt{loc}[\mathtt{a}, \emptyset])(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

However

$$\Gamma \models \mathsf{M} \not\approx_{\mathsf{bad}} \mathsf{N}$$

$D\pi F$ bisimulation

But $\approx_{\mathsf{bad}}$ is **too intensional!**

$$\mathsf{M} \Leftarrow (\nu\, k : \mathtt{loc[d,\{}l\mathtt{\}]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k : \mathtt{loc[a,}\emptyset\mathtt{]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

However

$$\Gamma \models \mathsf{M} \not\approx_{\mathsf{bad}} \mathsf{N}$$

$$\Gamma \triangleright \mathsf{M} \xrightarrow{(k:\mathtt{loc[d,\{}l\mathtt{\}]})l:a!\langle k\rangle} \ldots$$

$D\pi F$ bisimulation

But $\approx_{\mathsf{bad}}$ is **too intensional!**

$$\mathsf{M} \Leftarrow (\nu\, k : \mathtt{loc[d,\{}l\mathtt{\}]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k : \mathtt{loc[a,}\emptyset\mathtt{]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

However

$$\Gamma \models \mathsf{M} \not\approx_{\mathsf{bad}} \mathsf{N}$$

$$\Gamma \triangleright \mathsf{M} \xrightarrow{(k:\mathtt{loc[d,\{}l\mathtt{\}]})l:a!\langle k\rangle} \ldots$$

$$\Gamma \triangleright \mathsf{N} \xrightarrow{(k:\mathtt{loc[a,}\emptyset\mathtt{]})l:a!\langle k\rangle} \ldots$$

$D\pi F$ bisimulation

Solution:

1. New network representation Σ showing *observer's view of the network*
2. Derived actions: $\Sigma \triangleright M \overset{\alpha}{\longmapsto} \Sigma' \triangleright M'$
3. Filter label: $(\tilde{n} : \tilde{\mathrm{T}})l : a!\langle V \rangle$ to $(\tilde{n} : \tilde{\mathrm{S}})l : a!\langle V \rangle$
4. New bisimulation, $\approx$, defined over new *derived* actions α.

$D\pi F$ bisimulation

$$\mathsf{M} \Leftarrow (\nu\, k : \mathtt{loc[d, \{}l\mathtt{\}]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

$$\mathsf{N} \Leftarrow (\nu\, k : \mathtt{loc[a, \emptyset]})(l[\![a!\langle k\rangle]\!] \mid k[\![b!\langle\rangle]\!])$$

We now have

$$\Gamma \models \mathsf{M} \approx \mathsf{N}$$

$$\Sigma(\Gamma) \triangleright \mathsf{M} \xrightarrow{(k:\mathtt{loc[\emptyset]})l:a!\langle k\rangle} \ldots$$

$$\Sigma(\Gamma) \triangleright \mathsf{N} \xrightarrow{(k:\mathtt{loc[\emptyset]})l:a!\langle k\rangle} \ldots$$

Results

Soundness

$\Sigma(\Gamma) \models M \approx N$ implies $\Gamma \models M \cong N$

Completness

$\Gamma \models M \cong N$ implies $\Sigma(\Gamma) \models M \approx N$

Overview

- Motivation:
  - π-calculus overview
  - A distributed π-calculus
  - Failure in a distributed setting
- Behaviour and Partial Failure: DπF
- **Dependability: Fault Tolerance**

Fault Tolerance Example

$$\text{serverLoc} \Leftarrow (\nu\, data)\left(\begin{array}{l} l[\![req?(x_1, x_2).\text{go}\, k_1.data!\langle x_1, x_2\rangle]\!] \\ \\ \mid k_1[\![data?(y_1, y_2).\text{go}\, l.y_2!\langle \textbf{lookup}(y_1)\rangle]\!] \end{array}\right)$$

Fault Tolerance Example

$$\text{server2} \quad \Leftarrow \quad (\nu\, data)\left(\begin{array}{l} l\left[\!\left[req?(x_1, x_2).(\nu sync)\left(\begin{array}{l} \text{go } k_1.data!\langle x_1, sync\rangle \\ \mid \text{go } k_2.data!\langle x_1, sync\rangle \\ \mid sync?(z).x_2!\langle z\rangle \end{array}\right)\right]\!\right] \\ \\ \mid k_1[\![data?(y_1, y_2).\text{go } l.y_2!\langle \textbf{lookup}(y_1)\rangle]\!] \\ \mid k_2[\![data?(y_1, y_2).\text{go } l.y_2!\langle \textbf{lookup}(y_1)\rangle]\!] \end{array}\right)$$

Fault Tolerance Example

$$\text{srvPng} \Leftarrow (\nu\, data)\left(\begin{array}{l} l[\![\; serv?(x,y).\text{ping}\, k_1.\text{go}\, k_1.data!\langle x,y\rangle \\ \qquad\qquad \lceil \text{go}\, k_2.data!\langle x,y\rangle \rceil \;]\!] \\ \mid k_1[\![data?(x,y).\text{go}\, l\, .y!\langle \textbf{lookup}(x)\rangle]\!] \\ \mid k_2[\![data?(x,y).\text{go}\, l\, .y!\langle \textbf{lookup}(x)\rangle]\!] \end{array}\right)$$

Fault Tolerance Intuition

$server_2$ is *more fault tolerant than* serverLoc because

"it preserves its (entire) observable behaviour up to a limit n of faults injected"

Fault Tolerance Intuition

$server_2$ is *more fault tolerant than* serverLoc because

"it preserves its (entire) observable behaviour up to a limit n of faults injected"

- How and where to inject faults?
- What is observable behaviour?
- How to limit observation?

Internal and External View

Partition locations in Γ into two:

- Public: $\Gamma \vdash l : \mathtt{p}$
- Confined: $\Gamma \vdash l : \mathtt{c}$

Internal and External View

Partial View

- limit observer to *public* locations

$$\Gamma \vdash_{\text{obs}} O \quad \text{if } \mathbf{fn}(O) \in \text{Pub}(\Gamma)$$

- *never communicate confined* locations to the observer: guaranteed through a type system

Internal and External View

Partial View

- limit observer to *public* locations

$$\Gamma \vdash_{\mathsf{obs}} O \quad \text{if } \mathbf{fn}(O) \in \mathrm{Pub}(\Gamma)$$

- never communicate confined locations to the observer: guaranteed through aType System

Example

$$\text{If } \Gamma \vdash l : \mathrm{p},\ k : \mathrm{c}$$

$$\Gamma \vdash_{\mathsf{obs}} l[\![a!\langle b\rangle]\!]$$
$$\Gamma \not\vdash l[\![a!\langle k\rangle]\!]$$

Internal and External View

Partial View Barbs

$$\Gamma \triangleright M{\Downarrow_{a@l}} \text{ iff } M \Longrightarrow \equiv (\nu \tilde{n})M' \mid l[\![a!\langle n\rangle.Q]\!]$$
where $l, a \notin \tilde{n}$ and $\Gamma \vdash l$: **alive**, p

Partial View Contextual Relation

$$\Gamma \models M \ \mathcal{R} \ N \quad \text{if} \quad \forall O.\, \Gamma \vdash_{\text{obs}} O \text{ then } \Gamma \models M \mid O \ \mathcal{R} \ N \mid O \text{ and } \Gamma \models O \mid M \ \mathcal{R} \ O \mid N \ \ldots$$

Internal and External View

Partial View Reduction Barbed Congruence

The *largest* relation over *systems*, running over the *same network* denoted by $\Gamma \models M \cong_{\text{obs}} N$, that is:

- reduction closed
- partial view barb preserving
- partial view contextual

Partial View Bisimulation: $\approx_{\text{obs}}$

- $\approx_{\text{obs}} \subseteq \cong_{\text{obs}}$
- $\cong_{\text{obs}} \subseteq \approx_{\text{obs}}$

Internal and External View

Induce Failure

- Use the process $l[\![\mathsf{kill}]\!]$
- $F^n \Leftarrow \underbrace{l_1[\![\mathsf{kill}]\!] \mid \ldots \mid l_n[\![\mathsf{kill}]\!]}_{n}$
- $\Gamma \vdash F^n$ if $\mathbf{fn}(F^n) \in \mathsf{conf}(\Gamma)$

Internal and External View

Induce Failure

- Use the process $l[\![\mathsf{kill}]\!]$
- $F^n \Leftarrow \underbrace{l_1[\![\mathsf{kill}]\!] \mid \ldots \mid l_n[\![\mathsf{kill}]\!]}_{n}$
- $\Gamma \vdash F^n$ if $\mathbf{fn}(F^n) \in \mathsf{conf}(\Gamma)$

Example

$$\text{If } \Gamma \vdash l : \mathsf{p},\ k : \mathsf{c}$$

$$\Gamma \vdash F^1 \Leftarrow k[\![\mathsf{kill}]\!]$$
$$\Gamma \not\vdash F^2 \Leftarrow l[\![\mathsf{kill}]\!] | k[\![\mathsf{kill}]\!]$$

Fault Tolerance Definition

The configuration $\Gamma \triangleright N$ is fault tolerant up to n-faults iff:

$$\forall F^n \text{ such that } \Gamma \vdash F^n \quad \text{then}$$

$$\Gamma \models N \cong_{\text{obs}} N|F^n$$

Example

If $\Gamma \vdash l : \mathrm{p},\ k_1 : \mathrm{c},\ k_2 : \mathrm{c}$

Recall

$$\mathsf{serverLoc} \Leftarrow (\nu\, \mathit{data}) \left(\begin{array}{l} l[\![\mathit{req}?(x_1, x_2).\mathrm{go}\, k_1.\mathit{data}!\langle x_1, x_2\rangle]\!] \\ \mid k_1[\![\mathit{data}?(y_1, y_2).\mathrm{go}\, l.y_2!\langle \mathbf{lookup}(y_1)\rangle]\!] \end{array} \right)$$

Then $\Gamma \triangleright \mathsf{serverLoc}$ is *not* fault tolerant up to 1-fault because

$$\Gamma \models \mathsf{serverLoc} \not\approx_{\mathsf{obs}} \mathsf{serverLoc} \mid k_1[\![\mathsf{kill}]\!]$$

Example

If $\Gamma \vdash l : \mathrm{p},\ k_1 : \mathrm{c},\ k_2 : \mathrm{c}$

Recall

$$\text{server2} \quad \Leftarrow \quad (\nu\, \mathit{data})\left(\begin{array}{l} l\left[\!\left[\mathit{req}?(x_1, x_2).(\nu \mathit{sync})\left(\begin{array}{l} \text{go}\, k_1.\mathit{data}!\langle x_1, \mathit{sync}\rangle \\ \mid \text{go}\, k_2.\mathit{data}!\langle x_1, \mathit{sync}\rangle \\ \mid \mathit{sync}?(z).x_2!\langle z\rangle \end{array}\right)\right]\!\right] \\ \\ \mid k_1[\![\mathit{data}?(y_1, y_2).\text{go}\, l.y_2!\langle \mathbf{lookup}(y_1)\rangle]\!] \\ \mid k_2[\![\mathit{data}?(y_1, y_2).\text{go}\, l.y_2!\langle \mathbf{lookup}(y_1)\rangle]\!] \end{array}\right)$$

Example

If $\Gamma \vdash l : \mathrm{p},\ k_1 : \mathrm{c},\ k_2 : \mathrm{c}$

$\Gamma \triangleright \mathrm{server}_2$ is fault tolerant up to 1-fault because

$$\Gamma \models \mathrm{server}_2 \cong_{\mathrm{obs}} \mathrm{server}_2 \mid k_1[\![\mathrm{kill}]\!]$$
$$\Gamma \models \mathrm{server}_2 \cong_{\mathrm{obs}} \mathrm{server}_2 \mid k_2[\![\mathrm{kill}]\!]$$

Example

If $\Gamma \vdash l : \mathsf{p},\ k_1 : \mathsf{c},\ k_2 : \mathsf{c}$

Recall

$$\mathsf{srvPng} \quad \Leftarrow \quad (\nu\, data)\left(\begin{array}{l} l[\![\ serv?(x,y).\mathsf{ping}\ k_1.\mathsf{go}\ k_1.data!\langle x,y\rangle \\ \qquad\qquad \lceil \mathsf{go}\ k_2.data!\langle x,y\rangle \rceil\ ]\!] \\ \mid k_1[\![data?(x,y).\mathsf{go}\ l\ .y!\langle \mathbf{lookup}(x)\rangle]\!] \\ \mid k_2[\![data?(x,y).\mathsf{go}\ l\ .y!\langle \mathbf{lookup}(x)\rangle]\!] \end{array}\right)$$

Example

If $\Gamma \vdash l : \mathrm{p},\ k_1 : \mathrm{c},\ k_2 : \mathrm{c}$

$\Gamma \triangleright$ srvPng is *not* fault tolerant up to 1-fault because

$$\Gamma \models \mathrm{srvPng} \not\approx_{\mathrm{obs}} \mathrm{srvPng} \mid k_1[\![\mathrm{kill}]\!]$$

Conclusions

- Distributed Behaviour and Partial Failure: DπF
  - Reduction Semantics
  - Sound and Complete Bisimulation
- Dependability: Fault Tolerance
  - Formal Definition
  - Bisimulation Techniques

References

- **Location and Link Failure in a Distributed π-calculus** ,
  Adrian Francalanza, Matthew Hennessy, University of Sussex Technical Report, 2005:01, January 2005.
- **Failure and Fault Tolerance in a Distributed π-calculus** ,
  Adrian Francalanza, University of Sussex D.Phil. Thesis,*(expected May 2005).*