Logic and Semantics seminar
University of Cambridge Computer Laboratory
25 January 2002

Local Areas in the π-Calculus

Ian Stark and Tom Chothia

Laboratory for Foundations of Computer Science
Division of Informatics, University of Edinburgh

Open distributed systems

Many successful “distributed systems” are really loosely coupled collections of programs and machines, linked by agreements on how widely known names should refer to certain local resources. For example:

- well-known IP ports (finger, daytime, http, ftp);
- library calls (Java API, libc).

We can model these systems using the π-calculus, but problems arise from the fact that there is no way to have a private interaction on a channel with a well-known name.

What to do

We propose a *local area π-calculus* in which names may be known everywhere yet still refer to local resources.

This is enough to give a (very) high-level description of:

- TCP/IP, sockets;
- Internet routing, Network address translation (NAT);
- FTP and Napster.

For example, we can demonstrate the success and failure of FTP operating through a NAT firewall.

The π-calculus

Describes concurrent systems with dynamically created mobile names. These names serve (at least) two roles:

- Information — who knows what;
- Communication — channels for messages.

In the standard π-calculus these are closely interlocked; if you know the name of a channel, you can always exchange messages with everyone else who knows it. This is not always what is wanted...

Example — π-calculus internet daemon

An *internet daemon* routes incoming service requests to appropriate handlers. Here is a π-calculus model of this in the classic style. It works, but leaks a little.

Client:	$\nu c \,.\, \overline{server}\langle finger, c\rangle$	$\leftarrow$ ask server
	$\mid c(x).\overline{print}\langle x\rangle$	$\leftarrow$ print response
Server:	$server(service, reply).\overline{service}\langle reply\rangle$	$\leftarrow$ inet daemon
	$\mid finger(reply).\overline{reply}\langle users\rangle$	$\leftarrow$ finger daemon
	$\mid time(reply).\overline{reply}\langle now\rangle$	$\leftarrow$ time daemon

Local areas for the internet daemon

To plug the leaks, we assign levels to each channel that limit their reach.

$$c, \mathrm{server} \mathbin{@} \mathrm{net} \qquad \mathrm{print}, \mathrm{finger}, \mathrm{time} \mathbin{@} \mathrm{host}$$

We then mark out the boundary of each host and application as a *local area*.

$$\begin{aligned}
\text{Client:} \quad & \mathrm{host}\,[\ \mathrm{app}\,[\ \nu c{:}\mathrm{string@net}\ .\ \overline{\mathrm{server}}\langle \mathrm{finger}, c\rangle \\
& \qquad\qquad\qquad\qquad\qquad\qquad \mid c(x).\overline{\mathrm{print}}\langle x\rangle\ ]\ ] \\
\text{Server:} \quad & \mathrm{host}\,[\ \mathrm{app}\,[\ \mathrm{server}(\mathrm{service}, \mathrm{reply}).\overline{\mathrm{service}}\langle \mathrm{reply}\rangle\ ] \\
& \qquad\qquad \mid \mathrm{app}\,[\ \mathrm{finger}(\mathrm{reply}).\overline{\mathrm{reply}}\langle \mathrm{users}\rangle\ ] \\
& \qquad\qquad \mid \mathrm{app}\,[\ \mathrm{time}(\mathrm{reply}).\overline{\mathrm{reply}}\langle \mathrm{now}\rangle\ ]
\end{aligned}$$

Syntax for a local area π-calculus

We take a plain π-calculus

$$\mathbf{0} \qquad \overline{a}\langle\vec{b}\rangle \qquad a(\vec{b}).P \qquad !a(\vec{b}).P \qquad P \mid Q$$

choose some total order of levels,

$$\text{app} < \text{host} < \text{net}$$

and add terms for new areas and channels:

$\ell\,[P]$	local area at level ℓ
$\nu a{:}\sigma.P$	fresh channel a of type σ

$$\sigma ::= (\sigma_1 \ldots \sigma_n)@\ell \ .$$

Areas, like processes, are anonymous.

Dynamic scope vs. local areas

We have separated the two roles of names, distinguishing between the *scope* of a name (who knows it) from its *reach* (what it does).

Every name may have within its scope several disjoint *local areas* of communication.

Scope is flexible: it varies as processes pass on their knowledge of a name.

Local areas are rigid: replication can create new ones, but they do not change, and must nest correctly.

Scope and areas interact via *levels*: wherever a name travels, these determine its communication reach at any point.

Semantics for $l a \pi$

We can give for the $l a \pi$-calculus an inductive definition of well-typed terms $\mathbf{\Gamma} \vdash_{\ell} \mathrm{P}$ and a structural operational semantics $\mathbf{\Gamma} \vdash_{\ell} \mathrm{P} \xrightarrow{\alpha} \mathrm{Q}$.

The types constrain the use of names within the appropriate local areas and give us a safety result:

$$\text{If} \quad \mathbf{\Gamma} \vdash_{\ell} \mathrm{P} \xRightarrow{\bar{\mathrm{a}}\langle \vec{\mathrm{b}} \rangle} \mathrm{Q} \quad \text{or} \quad \mathbf{\Gamma} \vdash_{\ell} \mathrm{P} \xRightarrow{\mathrm{a}(\vec{\mathrm{b}})} \mathrm{Q} \quad \text{then} \quad \mathrm{level}_{\mathbf{\Gamma}}(\mathrm{a}) \geq \ell.$$

This confirms that no communication ever goes beyond its local area.

Encoding $la\pi$ in the π-calculus

There is a compositional encoding from $la\pi$ terms to the π-calculus; every local area becomes a process with a local *ether*.

All $la\pi$ names become data, sent over ethers.

There is an *operational correspondence*, and for observable actions behaviour matches quite well:

$$\mathrm{P} \xrightarrow{\overline{\mathrm{a}}\langle \mathrm{b} \rangle} \mathrm{P}' \quad \text{if and only if} \quad [\![\mathrm{P}]\!] \xrightarrow{\overline{\mathrm{e}}\langle \mathrm{a},\mathrm{b} \rangle} [\![\mathrm{P}']\!] \ .$$

But the translation introduces speculative input and divergence almost everywhere. It also exposes some communications to “packet snooping”.

Future directions

- Multiple FTP/NAT sessions; model-checker?
- Notions of observation: from the inside looking out.
- Bisimulation: when do two environments look the same to their components?
- Mobile areas.
- Using areas to organise denotational semantics for processes.

Conclusions

Examples of contexts in which names need to be widely known, while always referring to local resources:

- Standard libraries
- Applets, plugins
- Mobile agents
- Remote procedure call
- Service discovery
- Communicating structured data

The local area π-calculus can offer a way to describe these systems, and reason about how they behave and interact.

“Think globally, act locally”

Related work

Several extensions to the π-calculus look at *locations*, typically labelled with identifiers, and investigate issues like failure or causality.

Many systems using agents or mobile ambients allow only short-range communication. In others a channel may only be used where it is created, or may only receive messages at a single point.

Simple CCS *blocking* restricts communication on a particular channel to a given area, but requires concrete labelling of every restriction.

Types for the $l\alpha\pi$ Inetd

$$\begin{aligned} \text{c} &: \text{string@net} \\ \text{print} &: \text{string@host} \end{aligned} \qquad \begin{aligned} \text{server} &: (\text{service,response})@\text{net} \\ \text{finger} &: \text{service} \\ \text{time} &: \text{service} \end{aligned}$$

$$\begin{aligned} \text{service} &= \text{response@host} \\ \text{response} &= \text{string@net} \end{aligned}$$

Encoding $l\alpha\pi$ into π (details, simplified)

A parameter Δ maps levels to ethers. The interesting clauses are these:

$$[\![\ell[\mathsf{P}]]\!]_{\Delta} = \nu \mathsf{e}.[\![\mathsf{P}]\!]_{\Delta,\ell\mapsto\mathsf{e}} \qquad [\![\nu \mathsf{a}.\mathsf{P}]\!]_{\Delta} = \nu \mathsf{a}.[\![\mathsf{P}]\!]_{\Delta}$$

$$[\![\overline{\mathsf{a}}\langle \mathsf{b}\rangle]\!]_{\Delta} = \overline{\mathsf{e}}\langle \mathsf{a}, \mathsf{b}\rangle \qquad \text{where } \mathsf{e} = \Delta(\mathsf{level}(\mathsf{a}))$$

$$[\![\mathsf{a}(\mathsf{b}).\mathsf{P}]\!]_{\Delta} = \mu \mathsf{X}.\mathsf{e}(\mathsf{x}, \mathsf{b}).\text{if } \mathsf{x} = \mathsf{a} \text{ then } [\![\mathsf{P}]\!]_{\Delta} \text{ else } (\overline{\mathsf{e}}\langle \mathsf{x}, \mathsf{b}\rangle \mid \mathsf{X})$$