The Internals of Distributed TyCO

Luís Lopes
Departamento de Ciência de Computadores
Faculdade de Ciências, Universidade do Porto

joint work with

Departamento de Informática
Faculdade de Ciências, Universidade de Lisboa
Talk Outline

- The DiTyCO Calculus
- The Programming Language
- The Computational Model
- Current and Future Work
The TyCO Calculus • Syntax

\[
x \quad \text{channels}
\]
\[
X \quad \text{procedures}
\]
\[
l \quad \text{method labels}
\]
\[
P ::=\begin{array}{l}
0 \\
\mid P \mid P \\
\mid \textbf{new} x P \\
\mid x! l[\tilde{v}] \\
\mid x?\{l_1(\tilde{x}_1) = P_1, \ldots, l_n(\tilde{x}_n) = P_n\} \\
\mid \textbf{def} \; X_1(\tilde{x}_1) = P_1 \ldots X_n(\tilde{x}_n) = P_n \; \textbf{in} \; P \\
\mid X[\tilde{v}] 
\end{array}
\quad \text{terminated process}
\quad \text{concurrent composition}
\quad \text{new local variable}
\quad \text{asynchronous message}
\quad \text{object}
\quad \text{definition}
\quad \text{instantiation}
\]
The DiTyCO Calculus • Syntax

\[ s \]

\[ P ::= \]
\[ \begin{align*}
0 & \quad \text{terminated process} \\
| P & \quad \text{concurrent composition} \\
| \text{new } x P & \quad \text{new local variable} \\
| x!l[\bar{v}] & \quad \text{asynchronous message} \\
| x?\{l_1(\hat{x}_1) = P_1, \ldots, l_n(\hat{x}_n) = P_n\} & \quad \text{object} \\
| \text{def } X_1(\hat{x}_1) = P_1 \ldots X_n(\hat{x}_n) = P_n \text{ in } P & \quad \text{definition} \\
| X[\bar{v}] & \quad \text{instantiation}
\end{align*} \]

\[ N ::= \]
\[ \begin{align*}
0 & \quad \text{terminated network} \\
| N & \quad \text{concurrent composition} \\
| \text{new } x@s N & \quad \text{new local variable} \\
| \text{def } D@s \text{ in } N & \quad \text{definition} \\
| s[P] & \quad \text{site running process}
\end{align*} \]

(cf. talks on LSD\(\pi\) by Francisco Martins and António Ravara)
The Base Calculus • Operational Semantics

- reduction (always local, within sites):

  Communication  \( x?\{\ldots, l(\bar{x}) = P, \ldots \} | x!l[\bar{v}] \rightarrow \{\bar{v}/\bar{x}\}P \)

  Instantiation  \( \text{def } X(\bar{x}) = P \text{ in } X[\bar{v}] | Q \rightarrow \text{def } X(\bar{x}) = P \text{ in } \{\bar{v}/\bar{x}\}P | Q \)

- weak migration (site-to-site, triggered by lexical scope):

  Message Migration  \( r[x@s!l[\bar{v}]] \rightarrow s[x!l[\bar{v}\sigma_{rs}]] \)

  Object Migration  \( r[x@s?M] \rightarrow s[x?M\sigma_{rs}] \)

  Remote Instantiation  \( \text{def } D@s \text{ in } r[X@s[\bar{v}]] \rightarrow \text{def } D@s \text{ in } s[X[\bar{v}\sigma_{rs}]] \)
The Programming Language

Abstractions:

- straightforward extension of TyCO;
- sites are the programming modules;
- interface channels are explicitly located at sites;
- `export` defines the external site interface;
- `import` establishes bindings with remote resources.

Semantics:

- channels and procedures are the distributed resources;
- migration triggered by lexical scope;
- computation is always local.
The Programming Language

---

### Server

```python
def AppletS (self) =
    self ? {
        applet_1(p) =
        p?(x) = \$P_1\$ | AppletS[self],
        ...
        applet_k(p) =
        p?(x) = \$P_k\$ | AppletS[self]
    }

in export applets
in AppletS[applets]
```

### Client

```python
import applets from Server

in new p applets!applet_j[p] | p![v]
```

---

Department of Computer Science, University of Porto – Luís Lopes

Workshop on Global Computing, Rovereto, February 2003
The Computational Model

- flat network topology;
- peer-to-peer architecture;

from a logical point of view we have just:
  - sites

from an implementation point of view we have a three layered model:
  - network (provides a name-space);
  - nodes (proxies for sites);
  - sites.

network and nodes do not compute;
all computation is local to sites.
The Computational Model • Software Organization

The Network

- **Interface NameSpace** – the name-space functionality;
- **Class Network implements NameSpace** – the network provides a name-space;

The Nodes

- **Interface Proxy** – the node manager for Distributed TyCO;
- **Class Node implements Proxy** – a Distributed TyCO node is a proxy for sites;
The Sites

- **Interface InstructionSet** – the core TyCO instruction set;
- **Class VirtualMachine implements InstructionSet** – an implementation;
- **Class Externals** – operations external to the core TyCO definition;

- **Class Site extends VirtualMachine**
  - I/O queues;
  - translation table;
  - operations supporting distribution and mobility implemented under the package Externals.
network wide service that provides a name-space for computations;

- register a Site@IP;
- unregister a Site;
- register a resource Channel@Site;
- resolve a resource name Channel@Site.
The Computational Model • The Network Layer

- register Site@IP
- unregister Site
- export Channel@Site
- import Channel@Site

Name Service

IP.Site.Identifier

IP.Site.Identifier
The Computational Model • The Network Layer

Inside Site
Translation Table
Network

memory reference

IP Site Identifier
The Computational Model • The Node Layer

- one-to-one correspondence with physical/logical IP nodes;
- a proxy for sites that handles:
  - communication;
  - migration;
  - name-resolution;
- every IP node requires a Distributed TyCO proxy to run sites;
- the proxy shares an address space with the local sites;
- uses shared-memory for local communication.
The Computational Model • The Node Layer
The Computational Model • The Site Layer

TyCO Virtual Machine

- register based virtual machine:
  - program area with byte code;
  - heap with activation records and channels;
  - general purpose registers;
  - run-queue for run-time generated tasks.

- single/multi-threaded.

Extended with:

- queues for incoming/outgoing code;
- translation table for converting network references into local references;
- module of instructions for processing remote interactions.
The Computational Model • The Site Layer

- Translation Table
- Heap
- Registers
- Byte Code
- I/O Queues
- Run Queue
- Site
- TyCOvm
Computational Model • Putting it all Together

R[ x@S ? M ] →

(local channels translated to network addresses)

Site R

Translation Table

Heap

Registers

Run Queue

Byte

Code

OutGoing

Book-Keeping

Task Wrapper

to Proxy

Department of Computer Science, University of Porto – Luís Lopes
Workshop on Global Computing, Rovereto, February 2003
Computational Model • Putting it all Together

Site S

Translation Table
Heap
Registers
Byte Code
Run Queue
Book-Keeping
Task Wrapper

S[ x ? Mσ ]

(from Proxy)

InComming

(channels y@:s translated to local addresses)
Current Work and Future Developments

- Preparation of next release of TyCO
  - major revisions of compiler and virtual machine;
  - infra-structure for distributed development;
  - system documentation;
  - open system.
- Implementation of Distributed TyCO in this environment
- Multithreaded Virtual Machine
- A Debugger for (Distributed) TyCO

URL ⇒ http://www.ncc.up.pt/~lblopes/tyco/