A Distributed Calculus with Local Areas of Communication

Think Globally Act Locally

Tom Chothia

Abstract: This poster introduces a process calculus designed to capture the phenomenon of names which are known universally but always refer to local information. Our system extends the π-calculus so that a channel name can have within its scope several disjoint *local areas*. Such a channel name may be used for communication within an area, it may be sent between areas, but it cannot itself be used to transmit information from one area to another. Areas are arranged in a hierarchy of *levels*, distinguishing for example between a single application, a machine, or a whole network.

Introduction

Most computer programs make assumptions about the environment in which they operate: the facilities available, and how to use them. Often this involves using a globally-known name to access local resources. This calculus begins an investigation of the interaction between the scope over which a name is known and the local areas in which it operates.

Agents and Levels

This system is based around the π-calculus which provides an established framework for reasoning about names and communication. The π-calculus is extended with *local areas* or agents and each channel and agent are assigned a *level of operation*.

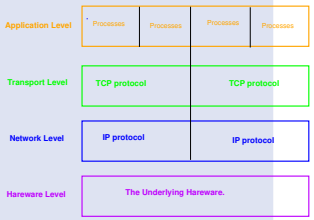
Levels

Levels represent the different strata in which a communication may take place. A system to model simple network interaction could have the following levels:

$$application < host < network$$

The lowest level is for channels that are local to individual processes. The next level is for channels that are local to each computer, such as a channel to print data to a screen. The final level is for channels that can be used across an entire network such as IP address.

The TCP/IP Reference Model, models network communication by assigns processes to one of four levels or layers:



The TCP/IP Reference Model

Agents

An *Agent* is a distinct process running at a fixed level. $l\ [\ P\ ]$ means an agent which is running process P at level l.

$$\begin{array}{rll} Process \text{P, Q} ::= & 0 & \text{inactive process} \\ | & P \mid Q & \text{parallel composition} \\ | & \bar{a} < \vec{b} > & \text{output tuple} \\ | & a(\vec{b}).P & \text{input} \\ | & !a(\vec{b}).P & \text{replicated input} \\ | & l\ [\ P\ ] & \text{local area/agent} \\ | & \nu\ a@l.P & \text{fresh channel } a \text{ at level } l \end{array}$$

- Communication on a channel assign to a level is restricted to stay inside any agent of that level.

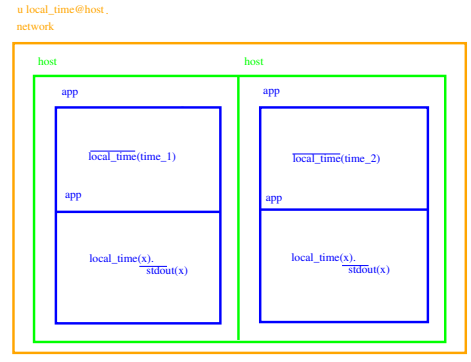
Example: local time on a computer

As a simple example, consider a network made up of two computers, where each computer has one application that boardcasts the local time and another that listens for it:

$$\begin{array}{c} network[\ Comp_1 \mid Comp_2\ ] \\ Comp_i \equiv host\ [Time_agent_i \mid Listener_Agent_i\ ] \\ Listener_Agent_i \equiv app\ [\overline{local_time}(x).\overline{stdout}(x)\ ] \\ Time_Agent_i \equiv app\ [!\ \overline{local_time}(time_i)\ ] \end{array}$$

As we wish the local time to be restricted to the computer it is on, the channel *local_time* is assigned the to the level *host*.See figure: *A local time system*.

The orange box is the network agent. The two green boxes are the computers and the blue boxes are the application. As *local_time* is assign the host level the communications on this channel are restricted to inside the green host boxes therefore the listener agent can only receive the time local to its host.



A local time system

The Semantics and the Type System

The calculus has a late-binding, small-step operational semantics. Local areas are enforced by using a *level context* to keep track of the levels assigned to each name and then making run-time checks on each input and output and also on all communications leaving an agent.

A type system can be used to remove the run time checks on the input and output actions by giving each channels a type made up of the level the channel is assigned to and the type of any channel it may send and receive. A single static check can then be made to see if a system respects its local areas.

Conclusion and Further work

The local area calculus provides a reasonable setting to explore the use of names that are known globally but act locally. Further work that is underway includes:

- Adding primitives for mobile agents.
- An encoding of this local area calculus into the π-calculus.
- An Interpreter for the calculus in the language MLj.
- A bisimulation relation that can be tuned to observe the communication at a single level.

Division of informatics