Mobile Ambients

Luca Cardelli

Digital Equipment Corporation, Systems Research Center

Andrew D. Gordon

University of Cambridge, Computer Laboratory

Presented by

Michael Hicks

CIS 640

Spring 1998

Mobility

- *Mobile Computing*
  - Computing devices are **mobile environments**
- *Mobile Computation*
  - Computations which move among environments are **mobile agents**

Administrative Domains

- Network level
  - **Firewall** partitioning of Intranet from Internet
  - **Address** partitioning of subnet from LAN
- Host level
  - Access to remote **resources** (disk, CPU, etc.)

Mobility and access require authorization

Outline

- Overview of approach and related work
- Mobility Calculus
  - Primitives, Semantics, and Examples
- Complete Ambient Calculus
  - Communication Primitives
  - Examples and Encoding of async π-calculus
- Criticisms and Conclusions

Ambients

- *Bounded* location for computation
  - a web page, an address space, a filesystem, a data object, a laptop, …
  - *not* a thread, collections of objects, …

Each ambient has a **name**, and may contain

- a collection of *local agents*
- a collection of *sub-ambients*

Names

- May be
  - created,
  - passed around, and
  - used to name new ambients
- May be used to derive *capabilities*

Related Work

- Obliq
- Telescript
- Java
- Linda

- π-calculus
- spi-calculus
- Chemical Abstract Machine
- join-calculus
- LLinda
- distributed calculi

Mobility Primitives

n		names	M ::=	capabilities
P,Q ::=		processes	$in\ n$	can enter n
	$(\nu n)P$	restriction	$out\ n$	can leave n
	$\mathbf{0}$	inactivity	$open\ n$	can open n
	$P \mid Q$	composition		
	$!P$	replication		
	$n[P]$	ambient		
	$M.P$	action		

Restriction

$(\nu n)P$

- creates a new (unique) name n within a scope of P
- may be used to name ambients and operate on ambients by name
- is transparent to reduction:

  $P \rightarrow Q \quad \Rightarrow \quad (\nu n)P \rightarrow (\nu n)Q$

Inaction

0

- does nothing

Composition

$$P \mid Q$$

- denotes process P executing in parallel with process Q
- is commutative and associative
- obeys the rule:

$$P \rightarrow Q \quad \Rightarrow \quad P \mid R \rightarrow Q \mid R$$

Replication

$!P$

- creates as many parallel replicas of P as needed
- may be used to express iteration and recursion
- to be reduced, it is first expanded to $P \mid !P$

Ambients

$n[P]$

- an ambient with name n within which P is executing:

  $P \rightarrow Q \quad \Rightarrow \quad n[P] \rightarrow n[Q]$

- may contain nested sub-ambients as well as processes running in parallel:

  $n[P_1 \mid \ldots \mid P_p \mid m_1[\ldots] \mid \ldots \mid m_q[\ldots]]$

Entry capability

in *n*. *P*

- instructs the surrounding ambient to enter a sibling ambient *n*
- If *n* doesn't exist, it blocks. If more than one exists, any one may be chosen
- Reduction rule:

$n[in\ m.\ P \mid Q] \mid m[R] \quad \rightarrow \quad m[n[P \mid Q] \mid R]$

Exit capability

out *n*. *P*

- instructs the surrounding ambient to exit its parent ambient *n*
- If *n* doesn't exist, it blocks.
- Reduction rule:

$m[n[\mathit{out}\ m.\ P \mid Q] \mid R] \quad \rightarrow \quad n[P \mid Q] \mid m[R]$

Open capability

open *n*. *P*

- dissolves the ambient *n* at the same level as the surrounding ambient
- If *n* doesn't exist, it blocks. If more than one exists, any one may be chosen
- Reduction rule:

  open n. *P* | *n*[*Q*] $\rightarrow$ *P* | *Q*

Example: Locks

acquire n. P ▷ *open n. P*

release n. P ▷ *n*[] | *P*

- *handshake:*

acquire n. release m. P | *release n. acquire m. Q*

Objective Moves

- Allows a computation to move into an ambient. Only possible if the ambient allows it

$$mv\ in\ n.\ P \mid n^{\uparrow\downarrow}[Q] \rightarrow^* n^{\uparrow\downarrow}[P \mid Q]$$
$$n^{\uparrow\downarrow}[mv\ out\ n.\ P \mid Q] \rightarrow^* P \mid n^{\uparrow\downarrow}[Q]$$

Objective Moves

allow n	▷	!*open n*
mv in n. P	▷	(ν*k*) *k*[*in n*. *in*[*out k*. *P*]]
mv out n. P	▷	(ν*k*) *k*[*out n*. *out*[*out k*. *P*]]
$n^{\downarrow}[P]$	▷	*n*[*P* | *allow in*]
$n^{\uparrow}[P]$	▷	*n*[*P*] | *allow out*
$n^{\uparrow\downarrow}[P]$	▷	*n*[*P* | *allow in*] | *allow out*

Synchronization on Named Channels

- Channel *n* is defined as $n^{\uparrow\downarrow}[]$

$n?.P \triangleright$ *mv in n. acquire rd. release wr. mv out n. P*

$n!.P \triangleright$ *mv in n. release rd. acquire wr. mv out n. P*

Mobility and Communication Primitives

P,*Q* ::=	processes
(*vn*)*P*	restriction
0	inactivity
P | *Q*	composition
!*P*	replication
M[*P*]	ambient
M.*P*	action
(x).*P*	input action
<*M*>	async output action

M ::=	capabilities
x	variable
n	name
in M	can enter *M*
out M	can leave *M*
open M	can open *M*
ε	null
M.M'	path

Communicable Values

- *Names*, *capabilities*, and ε may be exchanged
- Multiple capabilities may be combined into *paths* (such as for transmitting a route)

Ambient I/O

(*x*). *P*

<M>

- <*M*> releases a capability into the local ambient
- (*x*).*P* captures the result and binds it lexically
- Reduction rule:

(*x*). *P* | <*M*> $\rightarrow$ *P* {*x* $\leftarrow$ *M*}

Examples: Cells

- Allows for storage and retrieval of values at a named location

$cell\ c\ v \quad \triangleright \quad c^{\uparrow\downarrow}[\langle v \rangle \mid !(x).\langle x \rangle]$

$get\ c\ (x).\ P \quad \triangleright \quad mv\ in\ c.\ (x).\ (\langle x \rangle \mid mv\ out\ c.\ P)$

$set\ c\ (v).\ P \quad \triangleright \quad mv\ in\ c.\ (x).\ (\langle v \rangle \mid mv\ out\ c.\ P)$

Routable Packets

- A *packet* carries a computation
- May be *routed* to an ambient via path *M*
- An ambient may *forward* a packet via a path

packet pkt ▷ *pkt*[!(*x*).*x* | !*open route*]
route pkt with P to M ▷ *route*[in *pkt*. <*M*> | *P*]
forward pkt to *M* ▷ *route pkt with* 0 *to M*

Ether I/O

- Both parent and child ambients must be enabled for I/O. Children may then input and output using parent's *Ether*

$n[\leftrightarrow P]$	▷ a parent *n*[P] enabling Ether I/O
$n\leftrightarrow[P]$	▷ a child *n*[P] enabling Ether I/O
$n\leftrightarrow(x).P$	▷ receive a value from the Ether
$n\leftrightarrow\langle M\rangle$	▷ send a value into the Ether

Ether I/O

$n[\leftrightarrow P]$ $\triangleright$ $n[e^{\uparrow\downarrow}[] \mid P]$

$n^{\leftrightarrow}[P]$ $\triangleright$ $n^{\uparrow\downarrow}[P]$

$n^{\leftrightarrow}(x).P$ $\triangleright$ *mv out n. mv in e.* (x)*. mv out e. mv in n. P*

$n^{\leftrightarrow}\langle M\rangle$ $\triangleright$ *mv out n. mv in e.* $\langle M\rangle$

Encoding the π-calculus: channels

$ch\ n$	$\triangleright$ a channel
$(ch\ n)P$	$\triangleright$ a new channel
$n(x).P$	$\triangleright$ channel input
$n\langle M\rangle$	$\triangleright$ async channel output

Should satisfy the reduction

$$n(x).P \mid n\langle M\rangle \quad \rightarrow^* \quad P\{x \leftarrow M\}$$

Encoding the π-calculus: channels

$ch\ n \quad \triangleright\ n[!open\ io]$

$(ch\ n)P \quad \triangleright\ (\nu n)\ (ch\ n \mid P)$

$n(x).P \quad \triangleright\ (\nu p)\ (io[in\ n.\ (x).\ p[out\ n.\ P]] \mid open\ p)$

$n\langle M\rangle \quad \triangleright\ io[in\ n.\langle M\rangle]$

Channel Reduction

$ch\ n \mid n(x).P \mid n\langle M\rangle$

$\equiv (\nu p)\ (n[!open\ io] \mid io[in\ n.\ (x).\ p[out\ n.\ P]] \mid open\ p \mid io[in\ n.\langle M\rangle])$

$\rightarrow^* (\nu p)\ (n[!open\ io \mid io[(x).\ p[out\ n.\ P]] \mid io[\langle M\rangle]] \mid open\ p)$

$\rightarrow^* (\nu p)\ (n[!open\ io \mid (x).\ p[out\ n.\ P] \mid \langle M\rangle] \mid open\ p)$

$\rightarrow (\nu p)\ (n[!open\ io \mid p[out\ n.\ P\{x \leftarrow M\}]] \mid open\ p)$

$\rightarrow (\nu p)\ (n[!open\ io] \mid p[P\{x \leftarrow M\}] \mid open\ p)$

$\rightarrow (\nu p)\ (n[!open\ io] \mid P\{x \leftarrow M\})$

$\equiv ch\ n \mid P\{x \leftarrow M\}$

Encoding

$\langle\!\langle (\mathrm{v}n)P \rangle\!\rangle \quad \triangleright\ (\mathrm{v}n)\ (n[!open\ io] \mid \langle\!\langle P \rangle\!\rangle)$

$\langle\!\langle n(x).P \rangle\!\rangle \quad \triangleright\ (\mathrm{v}p)\ (io[in\ n.\ (x).\ p[out\ n.\ \langle\!\langle P \rangle\!\rangle]] \mid open\ p)$

$\langle\!\langle n\langle m\rangle \rangle\!\rangle \quad \triangleright\ io[in\ n.\langle m\rangle]$

$\langle\!\langle P \mid Q \rangle\!\rangle \quad \triangleright\ \langle\!\langle P \rangle\!\rangle \mid \langle\!\langle Q \rangle\!\rangle$

$\langle\!\langle !P \rangle\!\rangle \quad \triangleright\ !\langle\!\langle P \rangle\!\rangle$

Issues

- Interference
  - name clashes with “temporary” locations during evaluation with concurrent processes
- No type system (yet)
  - some legal programs are meaningless because of ‘type errors’ resulting from communication
- Notions of security are too simple

Conclusions

- Introduced notion of mobile ambients
- Presented a simple, yet powerful calculus
  - mobility
  - security
- Other document (the “Annex”) formally defines notions of *observational equivalence*