Reflecting joint work with Luís Caires, Andrew D. Gordon.
Isn’t $\pi$-calculus good/(bad) enough?

- Most process calculi use a powerful *channel* abstraction.
- This is “too abstract” for global communication: failure modes get increasingly harder to ignore.
- Channels abstract *wires*.
  
  What kind of wires do we actually need to model?

Two “Paradoxes” of global communication:

- Wires are very, very complicated.
  
  Most of Computer Science is about modeling or implementing wires.
- Even when nothing goes wrong, still things don’t work.
  
  Global Murphy’s Law.

Ditch channels, but keep $\pi$-names.
LAN Wires

- must handle partial failures
- audit all actions
- must apply access control
- must be authenticated
- relocate objects for load balancing
- linearize data
- keep client resource use to minimum
- optimize for 1000s of clients
- keep client resource use to minimum
- optimize for 1-shot access
- must handle partial failures
- audit all actions
- must apply access control
- must be authenticated
- relocate objects for load balancing
- linearize data
WAN Wires

- Often Unplugged
- Often Overloaded

- must handle net delays
- do not keep client status
- cache
- Support multiple architectures
- must survive QoS attacks
- UDP cannot cross firewalls
- optimize for 10000s of clients
- must trust/verify mobile code
- must encrypt
- Firewall
- in some countries, use weak crypto
- bandwidth costs money
- very, very long wires
- must encrypt
- proxy
- proxy
- must survive DOS attacks
- sync

2003-03-18 12:52
Mobile ("Wireless") Wires

- Handover protocols
- Unpredictable connectivity
- Roaming forwarding
- Determine closest cell
- Allocate bandwidth
- Tolerate noise

Mobile obstacles
Tunnel Effect

Mobile devices going around obstacles

Or, why $\pi$-calculus is not the whole story.
Tunnel Effect

Mobile devices going around obstacles

Or, why $\pi$-calculus is not the whole story.
Tunnel Effect

Mobile devices going around obstacles

Or, why $\pi$-calculus is not the whole story.
Tunnels vs Reliable Communication

Reliable communication = continuous unbreakable wires

Reliable communication + Tunnels
= wires get tangled (and untangling them is hard)
= eventually one can no longer move (or the wire breaks).
About the Tunnel Effect

In hardwired communication:

• *Capable* = *Able*.
• Unless, of course, something is broken.

In the tunnel effect:

• *Capable* but *unable* to communicate.
• Moreover, nothing is broken:
  • The client is working.
  • The server is working.
  • The tunnel tunnels.
  • The ether ethers.
  • All goes back to normal without need to *fix* anything.

Just one of a variety of phenomena where…
Sudden Inability to Communicate

No longer to be regarded as a failure

It is a state of affairs, due to many causes:

- Congestion (“The server could not be reached.”)
- Obstructions (“Infrared device out of sight.”)
- Geography (“No Cellnet service in Kinloch Rannoch.”)
- Security (“No Internet access on secure computer.”)
- Safety (“No electronic devices during takeoff and landing.”)
- Policy (“No mobile phones at Harrod’s.”)
- Privacy (“I am busy, go away.”)
- Psyche (“I left my wireless PDA in my other pants.”)
- Crime (“My laptop was stolen at Charles De Gaulle’s.”)
- Physics (“Please wait 8 minutes for answer from Mars.”)

Nothing is broken

- “broken” $\triangleq$ “somebody can be found to fix the problem”.
- In the cases above, nothing is “broken”. Yet, things don’t work.
- The failure model is not “it crashed” but…
Connectivity Depends on Location

1) Proximity:

![Diagram of proximity](image)

Ok. Fast (bounded delay), reliable, secure.

2) Physical distance: (possibly with virtual distance = 0)

![Diagram of physical distance](image)

No such thing as remote real-time control. No unbreakable links. Observationally different from (1).

3) Virtual distance: (possibly with physical distance = 0)

![Diagram of virtual distance](image)

No such thing as implicitly secure remote links. Observationally different from (1).
Global Computation

How do we embed the features and restrictions of global communication in a computational model?

Must abandon function call/handshake.
- We cannot afford to have every function call over the network to block waiting for an answer. (π vs. async-π.)

Must abandon symmetric multi-party (even async) communication.
- We cannot afford to solve consensus problems all the time. (async-π vs. join.)

Must abandon pointers/references.
- We cannot afford references of any kind that are always connected to their target, and we must be able to reconnect them. (π vs. ambients.)

Must abandon familiar failure models.
- We cannot assume that every failure leads to an exception.
- We cannot assume we are even allowed to know that a failure ever happened.
Ambients Approach

We want to capture in an abstract way, notions of locality, of mobility, and of ability to cross barriers.

An *ambient* is a place, delimited by a boundary, where computation happens.

Ambients have a name, a collection of local processes, and a collection of subambients.

Ambients can move in and out of other ambients, subject to capabilities that are associated with ambient names.

Ambient names are unforgeable (as in $\pi$ and spi).
The Ambient Calculus

The \textit{Ambient Calculus}: a computational model for:

- Behaviors that are \textit{capable} but sometimes \textit{unable} to communicate.
- Communication that is neither \textit{broken} nor \textit{not broken}.

Spatial structures (agents, networks, etc.) are represented by nested locations:

\begin{itemize}
  \item \textbf{Processes}
  \item \textbf{Tree Representation}
\end{itemize}

\begin{align*}
0 & \quad \text{(void)} \\
N[P] & \quad \text{(location)} \\
P \parallel Q & \quad \text{(composition)}
\end{align*}

\begin{tikzpicture}
  \node at (0,0) {\(n\)};
  \node at (-1,-1) {\(P\)};
  \node at (1,-1) {\(Q\)};
  \draw (-1,-1) -- (0,0) -- (1,-1);
\end{tikzpicture}
Mobility

*Mobility* is change of spatial structures over time.

\[
\begin{align*}
\text{a} & \mid \text{c[\text{out a. in b. P}]}, \quad \text{b} \mid \text{b[R]}
\end{align*}
\]
Mobility

*Mobility* is change of spatial structures over time.

\[
\begin{array}{ccc}
  a & | & b \\
  \\ & | & \\
  c & | & \\
\end{array}
\]

\[
\begin{array}{ccc}
  a & \uparrow & b \\
  \downarrow & & \\
  c & \downarrow & \\
  & & \begin{array}{ccc}
  & | & \\
  & | & \\
  & | & \\
\end{array}
\end{array}
\]

\[
a[Q] \quad | \quad c[\text{in } b. \ P] \quad | \quad b[R]
\]
Mobility

Mobility is change of spatial structures over time.
Communication

Communication is strictly local, within a given location.

Remote communication must be simulated by sending around mobile packets (which may get lost).
Security

Security issues are reduced to the ability to create, destroy, enter and exit locations.

- \( \pi \)-calculus restriction accounts for private capabilities.
- As for communication, capabilities can be exercised only in the right places.

enabled

<table>
<thead>
<tr>
<th>b</th>
<th>enter a</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>ok</td>
</tr>
</tbody>
</table>

blocked

<table>
<thead>
<tr>
<th>b</th>
<th>enter a</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>?!?</td>
</tr>
</tbody>
</table>
# The Ambient Calculus

<table>
<thead>
<tr>
<th>$P \in \Pi ::=$</th>
<th>Processes</th>
<th>$M ::= $</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\forall n)P$</td>
<td>restriction</td>
<td>$n$</td>
<td>name</td>
</tr>
<tr>
<td>0</td>
<td>inactivity</td>
<td>$\text{in } M$</td>
<td>entry capability</td>
</tr>
<tr>
<td>$P \parallel P'$</td>
<td>parallel</td>
<td>$\text{out } M$</td>
<td>exit capability</td>
</tr>
<tr>
<td>$M[P]$</td>
<td>ambient</td>
<td>$\text{open } M$</td>
<td>open capability</td>
</tr>
<tr>
<td>$!P$</td>
<td>replication</td>
<td>$\epsilon$</td>
<td>empty path</td>
</tr>
<tr>
<td>$M.P$</td>
<td>exercise a capability</td>
<td>$M.M'$</td>
<td>composite path</td>
</tr>
<tr>
<td>$(n).P$</td>
<td>input locally, bind to $n$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\langle M \rangle$</td>
<td>output locally (async)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Location Trees
- Spatial

### Actions
- Temporal

$n[] \equiv n[0]$  
$M \equiv M.0$  
(where appropriate)
Reduction Semantics

A structural congruence relation \( P \equiv Q \):

- On spatial expressions, \( P \equiv Q \) iff \( P \) and \( Q \) denote the same tree. So, the syntax modulo \( \equiv \) is a notation for spatial trees.
- On full ambient expressions, \( P \equiv Q \) if in addition the respective threads are “trivially equivalent”.
- Prominent in the definition of the logic.

A reduction relation \( P \rightarrow^* Q \):

- Defining the meaning of mobility and communication actions.
- Closed up to structural congruence:
  \[
  P \equiv P', \ P' \rightarrow^* Q', \ Q' \equiv Q \ \Rightarrow \ P \rightarrow^* Q
  \]
Reduction

\[ n\{\text{in } m.\ P \parallel Q\} \mid m[R] \rightarrow m\{n[P \parallel Q]\} \mid R \]  (Red In)

\[ m\{n[\text{out } m.\ P \parallel Q\} \mid R] \rightarrow n\{P \parallel Q\} \mid m[R] \]  (Red Out)

\[ \text{open } m.\ P \mid m[Q] \rightarrow P \parallel Q \]  (Red Open)

\[ (n).P \mid \langle M\rangle \rightarrow P\{n\leftarrow M\} \]  (Red Comm)

\[ P \rightarrow Q \Rightarrow (\forall n)P \rightarrow (\forall n)Q \]  (Red Res)

\[ P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q] \]  (Red Amb)

\[ P \rightarrow Q \Rightarrow P \parallel R \rightarrow Q \parallel R \]  (Red Par)

\[ P' \equiv P,\ P \rightarrow Q,\ Q \equiv Q' \Rightarrow P' \rightarrow Q' \]  (Red \(\equiv\))

\[ \rightarrow^* \text{ is the reflexive-transitive closure of } \rightarrow \]
Structural Congruence

\[ P \equiv P \]
\[ P \equiv Q \Rightarrow Q \equiv P \] (Struct Refl)
\[ P \equiv Q, Q \equiv R \Rightarrow P \equiv R \] (Struct Symm)
\[ P \equiv Q \Rightarrow (\forall n)P \equiv (\forall n)Q \] (Struct Trans)
\[ P \equiv Q \Rightarrow P \parallel R \equiv Q \parallel R \] (Struct Res)
\[ P \equiv Q \Rightarrow !P \equiv !Q \] (Struct Par)
\[ P \equiv Q \Rightarrow M[P] \equiv M[Q] \] (Struct Repl)
\[ P \equiv Q \Rightarrow M.P \equiv M.Q \] (Struct Amb)
\[ P \equiv Q \Rightarrow (n).P \equiv (n).Q \] (Struct Action)
\[ \varepsilon.P \equiv P \] (Struct Input)
\[ (M.M').P \equiv M.M'.P \] (Struct Output)
\((\forall n)0 \equiv 0\)                    \(\text{(Struct Res Zero)}\)
\((\forall n)(\forall m)P \equiv (\forall m)(\forall n)P\) \(\text{(Struct Res Res)}\)
\((\forall n)(P \mid Q) \equiv P \mid (\forall n)Q\)  \(\text{if } n \notin \text{fn}(P)\) \(\text{(Struct Res Par)}\)
\((\forall n)(m[P]) \equiv m[(\forall n)P]\) \(\text{if } n \neq m\) \(\text{(Struct Res Amb)}\)

\(P \mid Q \equiv Q \mid P\) \(\text{(Struct Par Comm)}\)
\((P \mid Q) \mid R \equiv P \mid (Q \mid R)\) \(\text{(Struct Par Assoc)}\)
\(P \mid 0 \equiv P\) \(\text{(Struct Par Zero)}\)

\(! (P \mid Q) \equiv !P \mid !Q\) \(\text{(Struct Repl Par)}\)
\(!0 \equiv 0\) \(\text{(Struct Repl Zero)}\)
\(!P \equiv P \mid !P\) \(\text{(Struct Repl Copy)}\)
\(!P \equiv !!P\) \(\text{(Struct Repl Repl)}\)

These axioms (particularly the ones for !) are sound and complete with respect to equality of spatial trees: edge-labeled finite-depth unordered trees, with infinite-branching but finitely many distinct labels under each node.
Ambient Calculus: Example

The packet $msg$ moves from $a$ to $b$, mediated by the capabilities $out a$ (to exit $a$), $in b$ (to enter $b$), and $open msg$ (to open the $msg$ envelope).

- **location a**
  - $a[msg[\langle M \rangle | out a. in b]]$
  - (exit) $\rightarrow a[]$
  - (enter) $\rightarrow a[]$
  - (open) $\rightarrow a[]$
  - (read) $\rightarrow a[]$

- **location b**
  - $b[open msg. (n). P]$
  - $\rightarrow b[open msg. (n). P]$
  - $\rightarrow b[msg[\langle M \rangle | in b]]$
  - $\rightarrow b[msg[\langle M \rangle | open msg. (n). P]]$
  - $\rightarrow b[(M) | (n). P]$
  - $\rightarrow b[P\{n \leftarrow M\}]$
Noticeable Inequivalences

Replication creates new names:

\((\forall n)!P \not\equiv (\forall n)!P\)

Multiple \(n\) ambients have separate identity:

\(n[P] \upharpoonright n[Q] \not\equiv n[P \upharpoonright Q]\)
An ambient can be graphically represented as a folder:

- Consisting of a folder name $n$,
- And active contents $P$, including:
  - Hierarchical data, and computations ("gremlins").
  - Primitives for mobility and communication.
Example: Message from $a$ to $b$
Example: Message from \( a \) to \( b \)
Example: Message from $a$ to $b$
Example: Message from $a$ to $b$
Example: Message from $a$ to $b$

$P\{a\}$
Example: Agent Authentication

(home)

`n`

`open n`

`n`

`g`

`out home. in home`

`x`

`P`

`out g. open g`
Example: Agent Authentication
Example: Agent Authentication
Example: Agent Authentication

home

n
open n
n

in home

out g.
open g

P
Example: Agent Authentication
Example: Agent Authentication
Example: Agent Authentication
Calculi for Communication

One basic notion

• Communication channels (a.k.a. wires).

One billion variations

• Value passing / name passing / process passing
• Synchronous / asynchronous / broadcast
• Internal choice / external choice / mixed choice / no choice
• Linearity / fresh output
• …
Calculi for Mobility

One basic notion

- Dynamic topology

One million variations

- Name mobility, process mobility
- Synchronous / asynchronous / datagram
- Actions / coactions / intermediaries
- Talk to local ether / talk to parent / talk to children
- …
Safe Ambients [Levi, Sangiorgi]

“Each action has an equal and opposite coaction.”

In Ambient Calculus it is difficult to count reliably the number of visitors to an ambient. The fix:

\[
\begin{align*}
n[\text{in } m. \ P \mid Q] \mid m[\text{in } m. \ R \mid S] & \rightarrow m[n[P \mid Q] \mid R \mid S] \quad \text{(In)} \\
m[\text{out } m. \ P \mid Q] \mid \text{out } m. \ R \mid S & \rightarrow n[P \mid Q] \mid m[R \mid S] \quad \text{(Out)} \\
\text{open } n. \ P \mid n[\text{open } n. Q \mid R] & \rightarrow P \mid Q \mid R \quad \text{(Open)} \\
(m).P \mid \langle M \rangle .Q & \rightarrow P\{m\leftarrow M\} \mid Q \quad \text{(Comm)}
\end{align*}
\]

The Ambient Calculus is recovered by sprinkling \texttt{!in} \ n, \texttt{!out} \ n, \texttt{!open} \ n appropriately.
Channeled Ambients [Pericas-Geertsen]

Each ambient contains a list of channels $c$ that are used for named communication within the ambient. They are restricted as usual.

\[
\begin{align*}
n[D, c; c(M).P \mid c(m).Q \mid R] & \quad \text{(Send)} \\
& \rightarrow n[D, c; P \mid Q\{m\leftarrow M\} \mid R] \\

n[D; \text{in m. } P \mid Q] \mid m[E; R] & \quad \rightarrow m[E; n[D; P \mid Q] \mid R] \quad \text{(In)} \\
m[E; n[D; \text{out m. } P \mid Q] \mid R] & \quad \rightarrow n[D; P \mid Q] \mid m[E; R] \quad \text{(Out)} \\
m[D; \text{open n. } P \mid n[E; Q] \mid R] & \quad \rightarrow m[D; P \mid Q \mid R] \quad \text{(Open)}
\end{align*}
\]
Boxed Ambients [Bugliesi, Castagna, Crafa]

I/O to parents/children is tricky to encode reliably in Ambient Calculus, but is a very natural basic primitive.

Boxed Ambients provide it directly (simplifying Seal):

\[
\begin{align*}
n[in \; m. \; P \mid Q] \mid m[R] & \rightarrow m[n[P \mid Q] \mid R] \quad (\text{In}) \\
m[n[out \; m. \; P \mid Q] \mid R] & \rightarrow n[P \mid Q] \mid m[R] \quad (\text{Out}) \\
(m).P \mid \langle M \rangle.Q & \rightarrow P\{m\leftarrow M\} \mid Q \quad (\text{Local}) \\
(m)^n.P \mid n[\langle M \rangle.Q \mid R] & \rightarrow P\{m\leftarrow M\} \mid n[Q \mid R] \quad (\text{Input } n) \\
\langle M \rangle^n.P \mid n[(m).Q \mid R] & \rightarrow P \mid n[Q\{m\leftarrow M\} \mid R] \quad (\text{Output } n) \\
\langle M \rangle.P \mid n[(m)^\uparrow.Q \mid R] & \rightarrow P \mid n[Q\{m\leftarrow M\} \mid R] \quad (\text{Input } \uparrow) \\
(m).P \mid n[\langle M \rangle^\uparrow.Q \mid R] & \rightarrow P\{m\leftarrow M\} \mid n[Q \mid R] \quad (\text{Output } \uparrow)
\end{align*}
\]
[CG] Ambient Calculus + [AC] Object Calculus =

\[
n.a(M).P \mid n[D; a(m).Q; R] \quad \xrightarrow{(Send)} \quad P \mid Q\{m\leftarrow M, \ self\leftarrow n\} \mid n[D; a(m).Q; R]
\]

\[
n[D; \ in\ m. \ P \mid Q] \mid m[E; R] \quad \xrightarrow{(In)} \quad m[E; n[D; P \mid Q] \mid R]
\]

\[
m[E; n[D; \ out\ m. \ P \mid Q] \mid R] \quad \xrightarrow{(Out)} \quad n[D; P \mid Q] \mid m[E; R]
\]

\[
m[E; \ open\ n. \ P \mid n[D; Q] \mid R] \quad \xrightarrow{(Open)} \quad m[E; D; P \mid Q \mid R]
\]
Joinbients [Anonymous]

Ambient Calculus + Join Calculus =

??? n[D; P]

n[D; in m. P | Q] | m[E; R] → m[E; n[D; P | Q] | R] (Join)

m[E; n[D; out m. P | Q] | R] → n[D; P | Q] | m[E; R] (In)

m[E; open n. P | n[D; Q]] → m[E; D; P | Q] (Out)

m[E; open n. P | n[D; Q]] → m[E; D; P | Q] (Open)
BioAmbients [Shapiro, Cardelli, et. al.]

Nameless membranes

\[
\begin{align*}
\text{[in n. } P & \mid Q \] \mid \text{[in n. } R & \mid S \] } & \rightarrow \text{[[P \mid Q] \mid R \mid S]} \quad \text{(In)} \\
\text{[[out n. } P & \mid Q \] \mid \text{out n. } R & \mid S \] } & \rightarrow \text{[P \mid Q] \mid [R \mid S]} \quad \text{(Out)} \\
\text{[merge n. } P & \mid Q \] \mid \text{[merge n. } R & \mid S \] } & \rightarrow \text{[P \mid Q \mid R \mid S]} \quad \text{(Merge)} \\
\ldots & \quad \text{(Comm)}
\end{align*}
\]
## Daring Classification

<table>
<thead>
<tr>
<th></th>
<th>Will work fine on a:</th>
<th>Will work fine on a:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>LAN</strong></td>
<td><strong>WAN</strong></td>
</tr>
<tr>
<td></td>
<td>(bounded-delay, integrated management, uniform access)</td>
<td>(unbounded-delay, federated management, restricted access)</td>
</tr>
<tr>
<td><strong>F-</strong></td>
<td><strong>Caluli</strong> (synch/asynch-)<strong>π, d-π</strong></td>
<td><strong>Caluli</strong> <strong>π-i, join</strong></td>
</tr>
<tr>
<td>(fixed</td>
<td><strong>DOOP</strong></td>
<td><strong>Infrastructure</strong> <strong>SOAP, B2C, B2B, P2P</strong></td>
</tr>
<tr>
<td>processes</td>
<td><strong>File Servers</strong></td>
<td><strong>Apps</strong> <strong>Email, Web, Kazaa</strong></td>
</tr>
<tr>
<td>and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>locations)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>M-</strong></td>
<td><strong>Caluli</strong> <strong>d-join</strong></td>
<td><strong>Caluli</strong> <strong>ambients, ..., seals</strong></td>
</tr>
<tr>
<td>(mobile</td>
<td><strong>AGLETS</strong></td>
<td><strong>Mobile Threads</strong></td>
</tr>
<tr>
<td>processes</td>
<td><strong>Trusted Applets</strong></td>
<td><strong>Worms, Agents?</strong></td>
</tr>
<tr>
<td>or</td>
<td><strong>Wireless Ethernet</strong></td>
<td><strong>Wireless Telephony</strong></td>
</tr>
<tr>
<td>locations)</td>
<td><strong>Work during meetings</strong></td>
<td><strong>Mobile B2C, B2B</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

Studied many encodings, type systems, and equivalences.

- Often building on π-calculus technology.

“Strong mobility” is still a dream, in practice.

- Although many interesting techniques have been proposed, typically in Java.

Ambients suggest new security models.

- Location-based; perhaps more intuitive.
- Analysis of security boundaries.
- But new security issues are also raised.

Ambients are “more” than π.

- Still don’t know how to encode ambients in π (vice versa is easy).
- For a generalization of both Ambients and π, see Milner’s BiGraphs.