Reflecting joint work with Luís Caires, Giorgio Ghelli, Andrew D. Gordon.
Introduction

• We are building infrastructure that allows us to be connected “everywhere all the time”.
  • Global wired and wireless speech and data networks.
  • *Local / reactive / synchronous / connected.*

• At the same time, we are building infrastructure that allows us to be isolated and protected from intrusion.
  • Answering machines, crypto, Great FireWall of China.
  • *Remote / deferred / asynchronous / blocked.*

• We cannot have it both ways. We will have to describe what we want to be *local* or *accessible* and we will have to adapt to what must necessarily be *remote* or *inaccessible*.

• All this applies on a very small scale (ad hoc networks), but global networks tend to stretch the imagination.
Outline

• Global Communication
  • Why it is different from, e.g., send/receive.

• Global Computation
  • Why it is different from, e.g., method invocation.

• Global Data
  • Why it is different from, e.g., arrays and records.
1. Global Communication

• Three “Paradoxes”:
  • Wires are very, very complicated. Most of Computer Science is about implementing wires.
  • Even when nothing breaks, still, things don’t work.
  • Having the capability to communicate does not mean being able to communicate.
In-Memory Wires

- relocation
- reference counting
- memory protection
- garbage collection
- garbage collection
- reference counting
- memory protection
- relocation

reachability analysis
shape analysis
aliasing analysis
memory protection
garbage collection
reference counting
relocation

C

S
LAN Wires

must handle partial failures
must apply access control
must be authenticated

proxy

linearize data
relocate objects
for load balancing

keep client resource use to minimum
optimize for 1-shot access
optimize for 1000s of clients

must handle partial failures
audit all actions

proxy

must apply access control
must be authenticated

proxy
WAN Wires

- Often Unplugged
  - must handle mobile code
  - net delays
  - do not keep client status

- Often Overloaded
  - UDP cannot cross firewalls
  - support multiple architectures
  - must survive DDoS attacks

- Often Costly
  - bandwidth costs money
  - very, very long wires

in some countries, use weak crypto

in some architectures, asynchronous
Mobile ("Wireless") Wires

- Handover protocols
- Unpredictable connectivity
- Roaming forwarding
- Tolerate noise
- Allocate bandwidth
- Determine closest cell
- Mobile obstacles
Tunnel Effect

Mobile devices going around obstacles
Tunnel Effect

Mobile devices going around obstacles
Tunnel Effect

Mobile devices going around obstacles
**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:
  • Whoever is *capable* of communication (holds one end of the wire) is always *able* to communicate (send/receive on the wire).
  • Unless, of course, something is broken.

• In the tunnel effect:
  • The client is *capable* of communication (holds one end of the “wire”) but is still *unable* to communicate in some cases.
  • Moreover, nothing is broken:
    • The client is working. The server is working.
    • The tunnel tunnels.
    • The ether works like physics says it should.
    • 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 range.")
    • Geography ("No Cellnet service in Kinloch Rannoch.")
    • Security ("No UPS pickup in Area 51.")
    • Policy ("No mobile phones allowed at Harrod’s.")
    • Privacy ("Don’t bother me now.")
    • 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” ≡ “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 “it’s in the wrong place”. 
Connectivity Depends on Location

- **Proximity:**
  - Ok. Fast (bounded delay), reliable, secure.

- **Physical distance:**
  - No such thing as remote real-time control. No unbreakable links.

- **Virtual distance:**
  - No such thing as implicitly secure remote links.
Summary: Global Communication

• Mobility is about:
  • Not only mobility of wire endpoints in simple topology
    \((\pi\)-calculus, distributed object systems)\)
  • But also mobility of wire endpoints in complex topology
    (Ambient Calculus, agent systems).
  • In complex topology, wires endpoints cannot be continuously
    connected.

• To model global (wide-area, mobile) communication:
  • We need to model \textit{locations} where communication is attempted.
  • We need to make the \textit{capability to communicate} independent
    from the \textit{ability to communicate}.
  • Capability without ability: security by location access control.
  • Ability without capability: security by resource access control.
2. Global Computation

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

- We must abandon the familiar notion of function call/handshake.
  - We cannot afford to have every function call over the network to block waiting for an answer. ($\pi$ vs. async-$\pi$.)

- We must even abandon the familiar notion of symmetric multi-party (even async) channel communication.
  - We cannot afford to solve consensus problems all the time. (async-$\pi$ vs. join.)

- We must abandon the familiar notion of pointers/references.
  - We cannot afford references of any kind that are always connected to their target, and we must be able to reconnect them later. ($\pi$ vs. ambients.)

- We 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.
The Ambient Calculus

- The *Ambient Calculus*: a computational model for:
  - Behaviors that are *capable* but sometimes *unable* to communicate.
  - Communication that is neither *broken* nor *not broken*.
- To this end, spatial structures (agents, networks, etc.) are represented by nested locations:

**Processes**

0 (void)

n[P] (location)

P | Q (composition)

**Tree Representation**

- n
- P
- Q
Mobility

- **Mobility** is change of spatial structures over time.

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

- *Mobility* is change of spatial structures over time.
Mobility

- Mobility is change of spatial structures over time.

\[ a[Q] \quad | \quad b[R \mid c[P]] \]
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 capability 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.
Properties of Global Computation

- In addition to describing global computations, we want to specify their properties.
- These often have the form:
  - Right now, we have a spatial configuration, and later, we have another spatial configuration.
  - E.g.: Right now, the agent is outside the firewall, …
Properties of Global Computation

• In addition to describing global computations, we want to specify their properties.

• These often have the form:
  • Right now, we have a spatial configuration, and later, we have another spatial configuration.
  • E.g.: Right now, the agent is outside the firewall, and later (after running an authentication protocol), the agent is inside the firewall.
A Modal Specification Logic

- In a modal logic, the truth of a formula is relative to a state (called a *world*).
  - Temporal logic: current time.
  - Program logic: current store contents.
  - Epistemic logic: current knowledge. Etc.

- In our case, the truth of a *space-time modal formula* is relative to the *here and now* of a process.
  - The formula $n[0]$ is read:
    
    *there is here and now an empty location called n*
  - The operator $n[Δ]$ is a single step in space (akin to the temporal next), so we can talk about that place one step down into $n$.
  - Other modal operators talk about undetermined times (in the future) and undetermined places (in the location tree).
Logical Formulas

\[ \text{Formulas} \quad (\eta \text{ is a name } n \text{ or a variable } x) \]

\[ T \quad \text{true} \]
\[ \neg A \quad \text{negation} \]
\[ A \lor A' \quad \text{disjunction} \]
\[ 0 \quad \text{void} \]
\[ \eta[A] \quad \text{location} \quad A@\eta \quad \text{location adjunct} \]
\[ A|A' \quad \text{composition} \quad A\triangleright A' \quad \text{composition adjunct} \]
\[ \eta\otimes A \quad \text{revelation} \quad A\otimes \eta \quad \text{revelation adjunct} \]
\[ \Diamond A \quad \text{somewhere modality} \]
\[ \Box A \quad \text{sometime modality} \]
\[ \forall x.A \quad \text{universal quantification over names} \]
Satisfaction for Basic Operators

- $\models 0$

- $\models n[A]$ if $P \models A$

- $\models A \cup B$ if $P \models A$ and $Q \models B$

- $\models A@n$ if $P \models A$

- $\models A \triangleright B$ if for all $Q \models A$ we have $P \triangleright Q \models B$
N.B.: instead of $\Diamond \mathcal{A}$ and $\mathcal{A}$ we can use a “temporal next” operator $\sigma \mathcal{A}$, along with the existing “spatial next” operator $n[\mathcal{A}]$, together with $\mu$-calculus style recursive formulas.
Satisfaction for Hidden and Public Names

\[ P \{ m \leftarrow n \} \]

Etc.

\[ \models \text{Hx.}\mathcal{A} \quad \text{if} \quad \exists m \notin fn(P,\mathcal{A}) \]

\[ P\{n \leftarrow m\} \models \mathcal{A}\{x \leftarrow m\} \]

\[ P \models \mathcal{O}n \quad \text{if} \quad n \in fn(P) \]

(Technically, \text{Hx.}\mathcal{A} and \mathcal{O}n are defined from \( n \mathcal{O}\mathcal{A} \) and a Gabbay-Pitts axiom.)
Example: “Shared Secret” Postcondition

• Consider a situation where “a hidden name $x$ is shared by two locations $n$ and $m$, and is not known outside those locations”.

\[ Hx.(n[\mathcal{C}x] \mid m[\mathcal{C}x]) \]

• $P \models Hx.(n[\mathcal{C}x] \mid m[\mathcal{C}x])$

\[ \iff \exists r \in \Lambda. r \not\in \text{fn}(P) \cup \{n,m\} \land \exists R',R'' \in \Pi. P \equiv (\forall r)(n[R'] \mid m[R'']) \land r \in \text{fn}(R') \land r \in \text{fn}(R'') \]

• E.g.: take $P = (\forall p) \ (n[p][] \mid m[p[][]])$. 
Possible Applications

• Verifying security+mobility protocols.

• Modelchecking security+mobility assertions:
  • If $P$ is $!$-free and $\mathcal{A}$ is $\triangleright$-free, then $P \models \mathcal{A}$ is decidable. (PSPACE-complete [Cheratonik et al. ’01].)
  • This provides a way of mechanically checking (certain) assertions about (certain) mobile processes.

• Expressing mobility/security policies of host sites.
  • Conferring more flexibility than just sandboxing the agent.

• Just-in-time verification of code containing mobility instructions
  • By either modelchecking or proof-carrying code.
3. Global Data

- Semistructured Data (a.k.a. XML)

  (Abiteboul, Buneman, Suciu: “Data on the Web” Morgan Kaufman’00.)

```
C  A  3
```

```
G  D  K
```

Diagram:

```
   Articles
  /     \
Paper  Paper
|      |
Author Title Year
   |
Author
   |
Author
   |
   Title
```
Unusual Data

• Not really arrays/lists:
  • Many children with the same label, instead of indexed children.
  • Mixture of repeated and non repeated labels under a node.

• Not really records:
  • Many children with the same label.
  • Missing/additional fields with no tagging information.

• Not really variants:
  • Labeled but untagged unions.

• New “flexible” type theories are required.
  • Based on the “effects” of processes over trees (Ambient Types).
  • Based on tree automata (Xduce).

• Unusual data.
  • Yet, it aims to be the new universal standard for interoperability of programming languages, databases, e-commerce...
Analogies

• An accidental(?) similarity between two areas:

• Semistructured Data is the way it is because:
  • “Cannot rely on uniform structure” of data. Abandon schemas based on records and disjoint unions.
  • Adopt “self-describing” data structures: Edge-labeled trees (or graphs).

• Mobile Computation is the way it is because:
  • “Cannot rely on static structure” of networks. Abandon type systems based on records and disjoint unions.
  • Adopt “self-describing” network structures: Edge-labeled trees (or graphs) of locations and agents.

• Both arose out of the Web, because things there are just too dynamic for traditional notions of data and computation.
Implications

• Immediate implication: a new, uniform, model of data and computation on the Web, with opportunities for cross-fertilization:
  • Programming technology can be used to typecheck, navigate, and transform both dynamic network structures and the semistructured data they contain. Uniformly.
  • Database technology can be used to search through both dynamic network structures (“resource discovery”), and the semistructured data they contain. Uniformly.

• This is still a dream, but it did motivate us to apply a particular technology developed for mobile computation to semistructured data:
  • Specification Logic ➔ Query Logic
A Query Language for Semistructured Data

- **Information trees** $I \in \mathcal{FT}$ (semistructured data)
- **Information terms** $F$ (denoting information trees)
- **Formulas** $\mathcal{A}$ (denoting sets of information trees)
- A semantics of terms $[F] \in \mathcal{FT}$
- A semantics of formulas $[\mathcal{A}] \subseteq \mathcal{FT}$
- A satisfaction (i.e. matching) relation $F \models \mathcal{A}$ (i.e. $[F] \in [\mathcal{A}]$)
- A query language $Q$ (including from $F \models \mathcal{A}$ select $Q'$)
- A (naïve/reference) query semantics $[Q] \in \mathcal{FT}$
- A **table algebra** for matching evaluation (i.e. for $F \models \mathcal{A}$)
- A (refined) query semantics / query evaluation procedure for $Q$, based on the table algebra. Correct w.r.t. $[Q]$. 
## The Query Logic

\[ \mathcal{A}, \mathcal{B} \in \Phi ::= \text{Formulas} \quad (\eta \text{ is a name } n \text{ or a variable } x) \]

- \( \mathcal{T} \) \quad true
- \( \neg \mathcal{A} \) \quad negation
- \( \mathcal{A} \land \mathcal{B} \) \quad conjunction
- \( \exists x.\mathcal{A} \) \quad existential quantification over label variables
- \( \eta \sim \eta' \) \quad label comparison
- \( \mathcal{0} \) \quad root
- \( \eta[\mathcal{A}] \) \quad edge
- \( \mathcal{A} \mid \mathcal{B} \) \quad composition
- \( X \) \quad tree variable
- \( \exists X.\mathcal{A} \) \quad existential quantification over tree variables
- \( \xi \) \quad recursion variable
- \( \mu \xi.\mathcal{A} \) \quad recursive formula (least fixpoint)

\( \xi \) may occur only \textit{positively} in \( \mathcal{A} \)
**Example: Schemas**

- A logic is a “very rich type system”. Hence we can comfortably represent various kinds of schemas.
  - However, extensions (or unpleasant encodings) are required for ordered data: $\mathcal{A} \mid \mathcal{B}$ vs. $\mathcal{A} ; \mathcal{B}$.

- **Ex.:** Xduce-like schemas:

  - $\mathbf{0}$: the empty tree
  - $\mathcal{A} \mid \mathcal{B}$: an $\mathcal{A}$ next to a $\mathcal{B}$
  - $\mathcal{A} \lor \mathcal{B}$: either an $\mathcal{A}$ or a $\mathcal{B}$
  - $n[\mathcal{A}]$: an edge $n$ leading to an $\mathcal{A}$
  - $\mathcal{A}^*$: $\mu \xi. \mathbf{0} \lor (\mathcal{A} \mid \xi)$, the merge of zero or more $\mathcal{A}$s
  - $\mathcal{A}^+$: $\mathcal{A} \mid \mathcal{A}^*$, the merge of one or more $\mathcal{A}$s
  - $\mathcal{A}?$: $\mathbf{0} \lor \mathcal{A}$, zero or one $\mathcal{A}$
Example: Search

- Search:
  - “Find one of my articles (ignore non-articles); bind to $X$ all info under the *article* label”:
    \[
    S = \exists X. \text{article}[(\text{author}[\text{Cardelli}[0]] \mid T) \land X] \mid T
    \]
  - Can use recursive formulas to search deeper:
    \[
    \mu \xi. S \lor \exists x. (x[\xi] \mid T)
    \]

- Not a query language yet.
  - It searches for one instance, not all instances.
  - Some *collecting* primitive must be added. This is going to be based on the logical notion of *satisfaction*. 
The Query Language

\[ Q ::= \]

\[ \textbf{from } Q \vdash \mathcal{A} \textbf{ select } Q' \]

\[ X \]

\[ 0 \]

\[ \eta[Q] \]

\[ Q | Q' \]

\[ f(Q) \]

\begin{itemize}
  \item \textbf{from } Q \vdash \mathcal{A} \textbf{ select } Q'
  
  All the matches of \( Q \) with \( \mathcal{A} \) are computed, producing bindings for the \( x \) and \( X \) variables that are free in \( \mathcal{A} \). The result expression \( Q' \) is evaluated for each (distinct!) such binding, and all the results are merged by \( | \).

  \item N.B.: This general approach to building a query language \( Q \) for a logic \( \mathcal{A} \), is fairly independent from the details of the logic.
\end{itemize}
Query Examples

- **Joins**
  Merge info about persons from two db’s:

  \[
  \begin{align*}
  &\text{from } db1 \equiv .\text{person}[\text{name}[X^\lambda] | Y^\lambda] \text{ select} \\
  &\text{from } db2 \equiv .\text{person}[\text{name}[X] | Z^\lambda] \text{ select} \\
  &\text{person}[\text{name}[X] | Y | Z]
  \end{align*}
  \]

- **Restructuring**
  Rearrange publications from by-article to by-year, for each distinct year (i.e., for each distinct binding of \(X\)):

  \[
  \begin{align*}
  &\text{from } db \equiv .\text{article}[\text{year}[X^\lambda]] \text{ select} \\
  &\text{publications-by-year[}
  \begin{align*}
  &\text{year}[X] | \\
  &\text{from } db \equiv .\text{article}[\text{year}[X] | Z^\lambda] \text{ select } \text{article}[Z]
  \end{align*}
  \end{align*}
  \]

  \(Z\) binds all fields except \(\text{year}\); this is rather unusual in QL’s
4. Summary

• Global Communication
  • Broadens communication mechanisms.
  • But also restricts the ways in which we can communicate.
    “Connected anytime anywhere to anything.” NOT!

• Global Data
  • Relaxes the traditional structure of data.
  • But also restricts what we can assume about it.
    “It’s just XML.” NOT!

• Global Computation
  • Extends and connects all computational resources.
  • But must deal with new notions of data and communication.
    “I’ll just write a script to manage my virtual program committee meeting.” NOT!
  • New opportunities: data structures and network structures “look the same”.
Conclusions

• **Global problems**
  - New challenge for most aspects of computation.

• **Which require global solutions**
  - Uniform solutions hard to implement ("reboot the internet").
  - Federated solutions more likely.
  - Everybody must be able to connect to everybody.
  - Everybody must be able exchange data.
  - Everybody must be able to invoke everybody’s programs.

• **Challenges for the present and future**
  - Build the infrastructure(s), both practical and theoretical, that will make all this easy.
Acknowledgments: Andrew Herbert for “wire slide” concept.