Computation on Wide Area Networks

Luca Cardelli

Microsoft Research

Lipari, July 2001

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

Part 1

Global Interaction

Luca Cardelli

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.

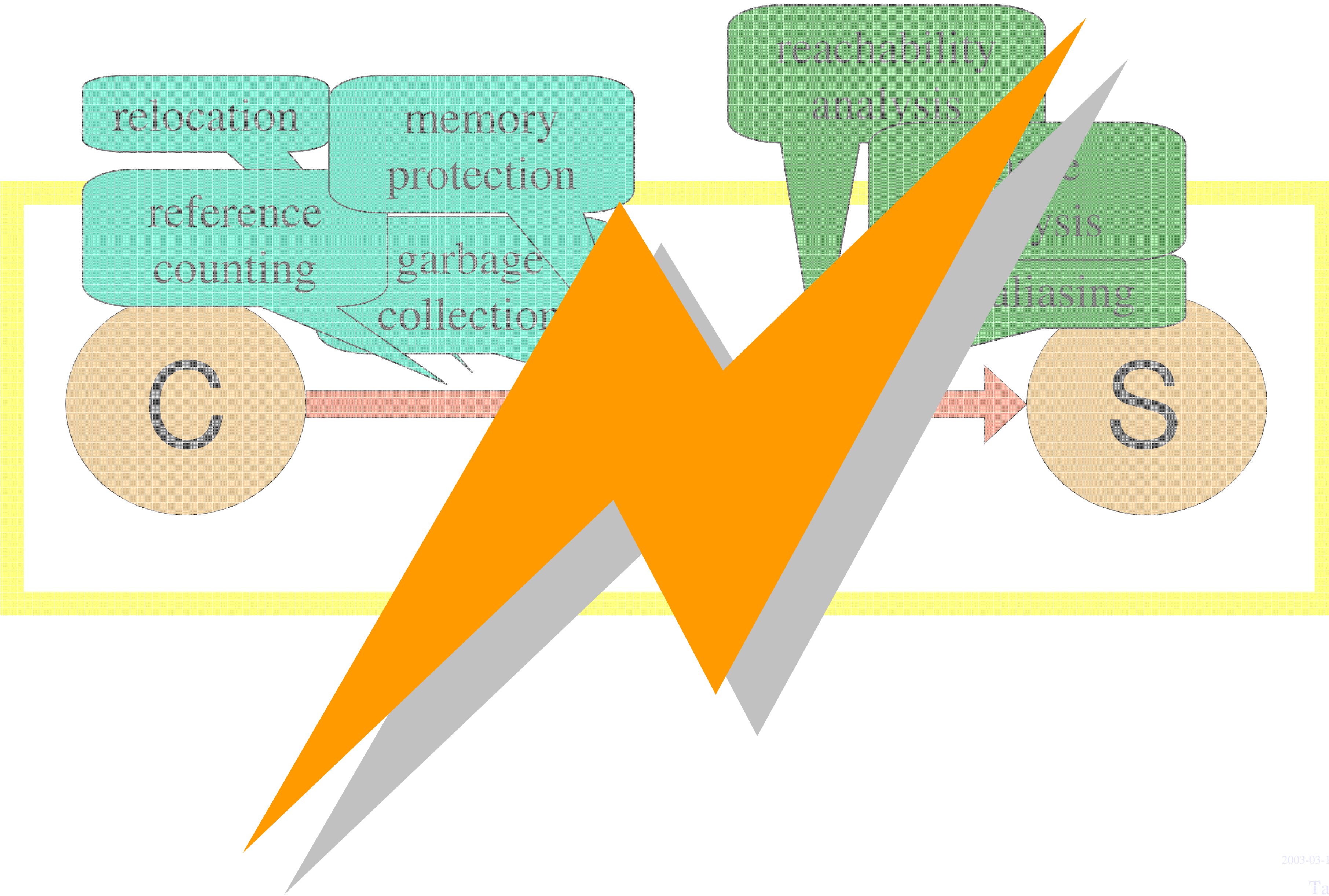
Outline

- Global Communication
  - Why it is different from, e.g., send/receive.
- Global Computation
  - Why it is different from, e.g., method invocation.
- Global Languages
  - Why they are different from, e.g. Pascal and ML.

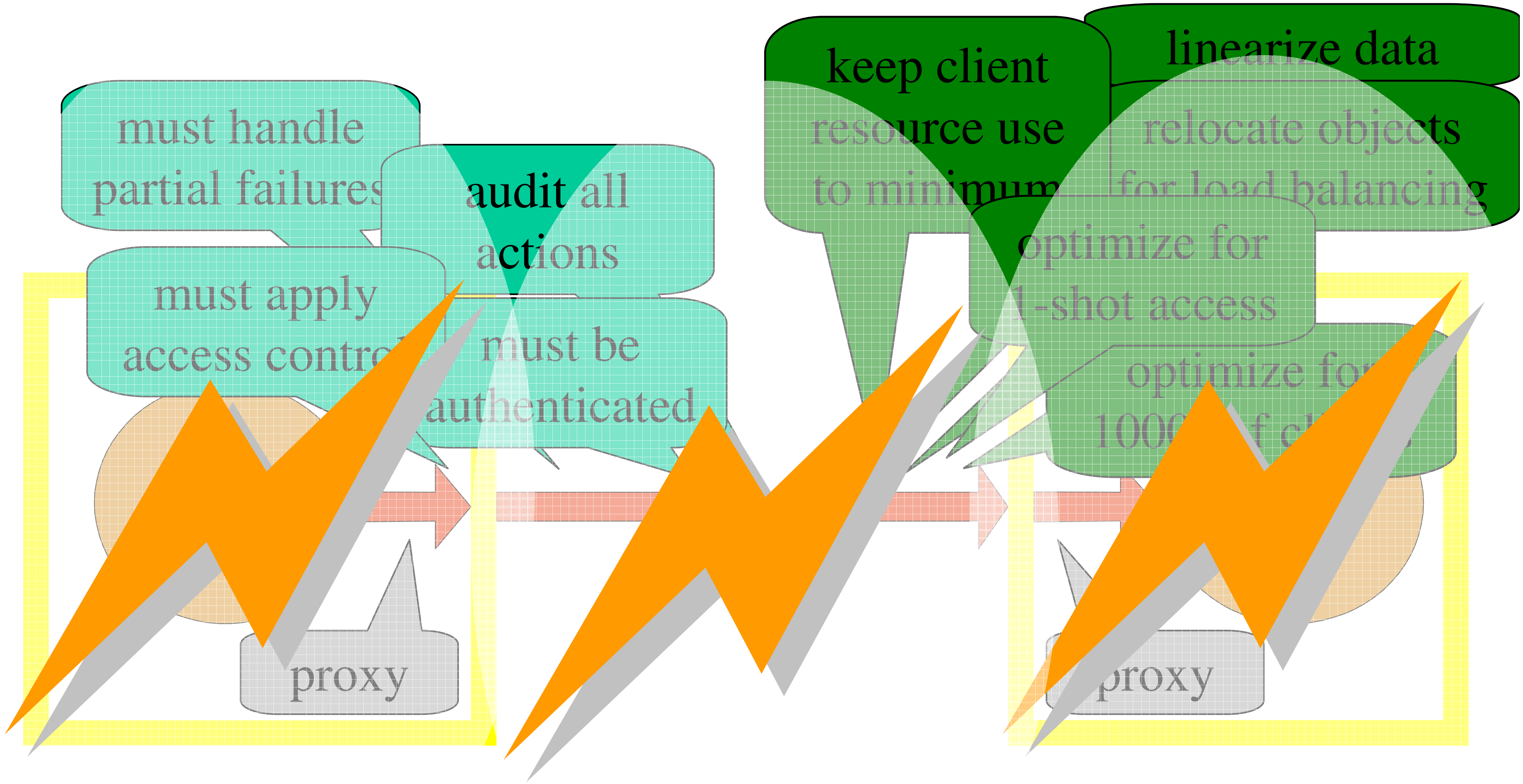
1. Global Communication

- 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

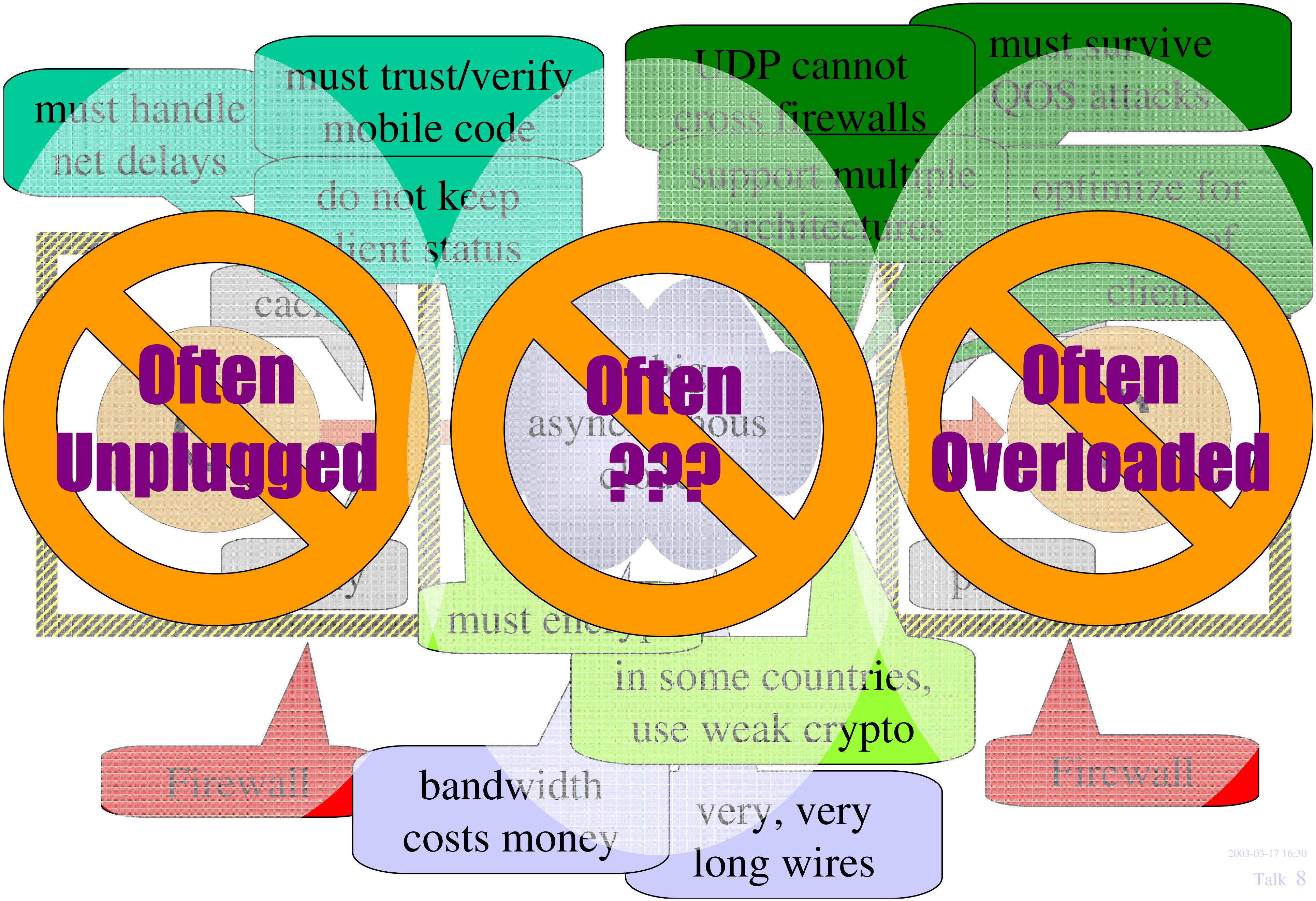


LAN Wires



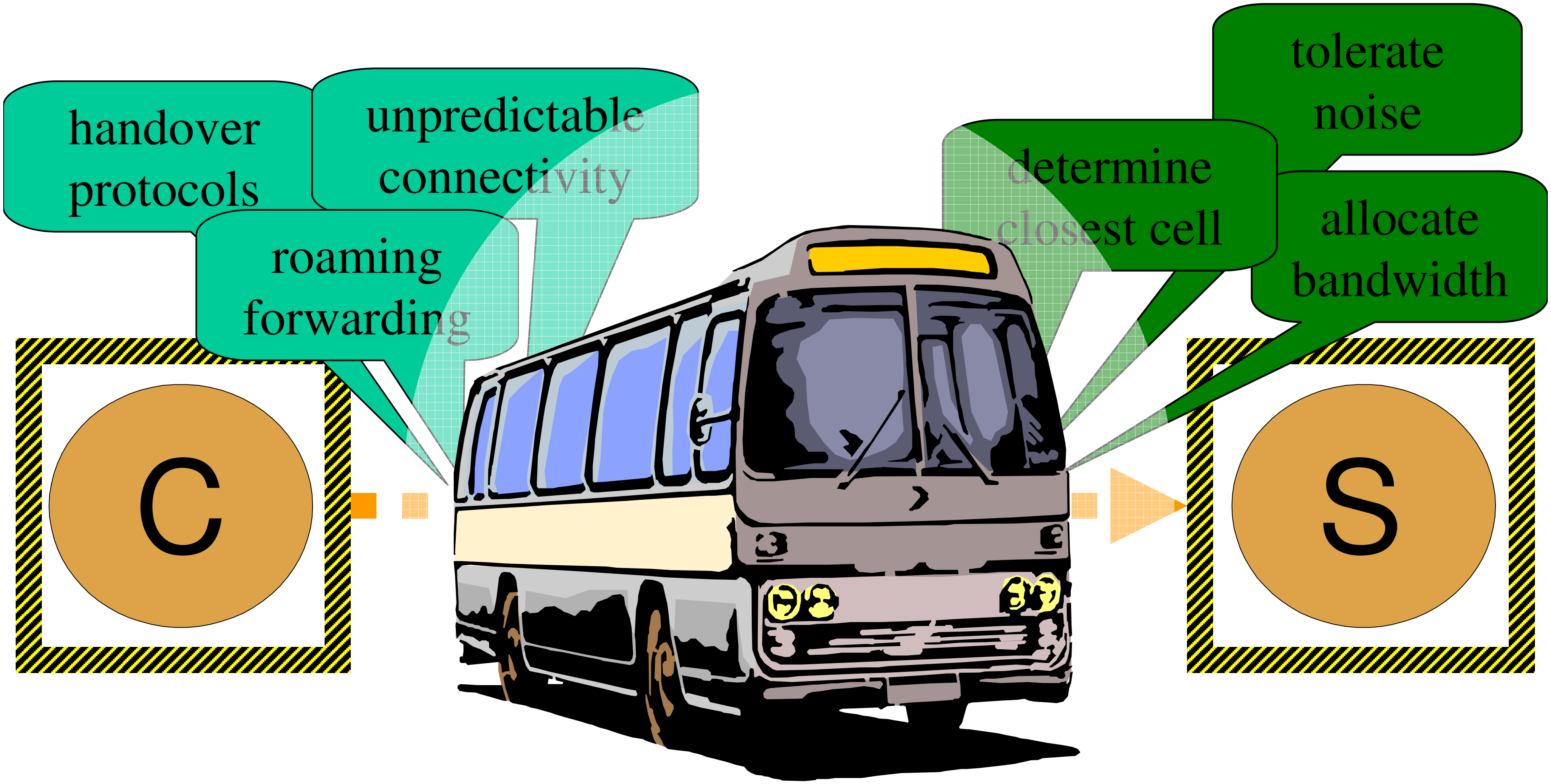
must handle partial failures
audit all actions
must apply access control
must be authenticated
keep client resource use to minimum
linearize data
relocate objects for load balancing
optimize for 1-shot access
proxy
proxy

WAN Wires



must handle net delays
must trust/verify mobile code
do not keep client status
UDP cannot cross firewalls
must survive QOS attacks
support multiple architectures
optimize for
Often Unplugged
Often ???
Often Overloaded
in some countries, use weak crypto
bandwidth costs money
very, very long wires
Firewall
Firewall

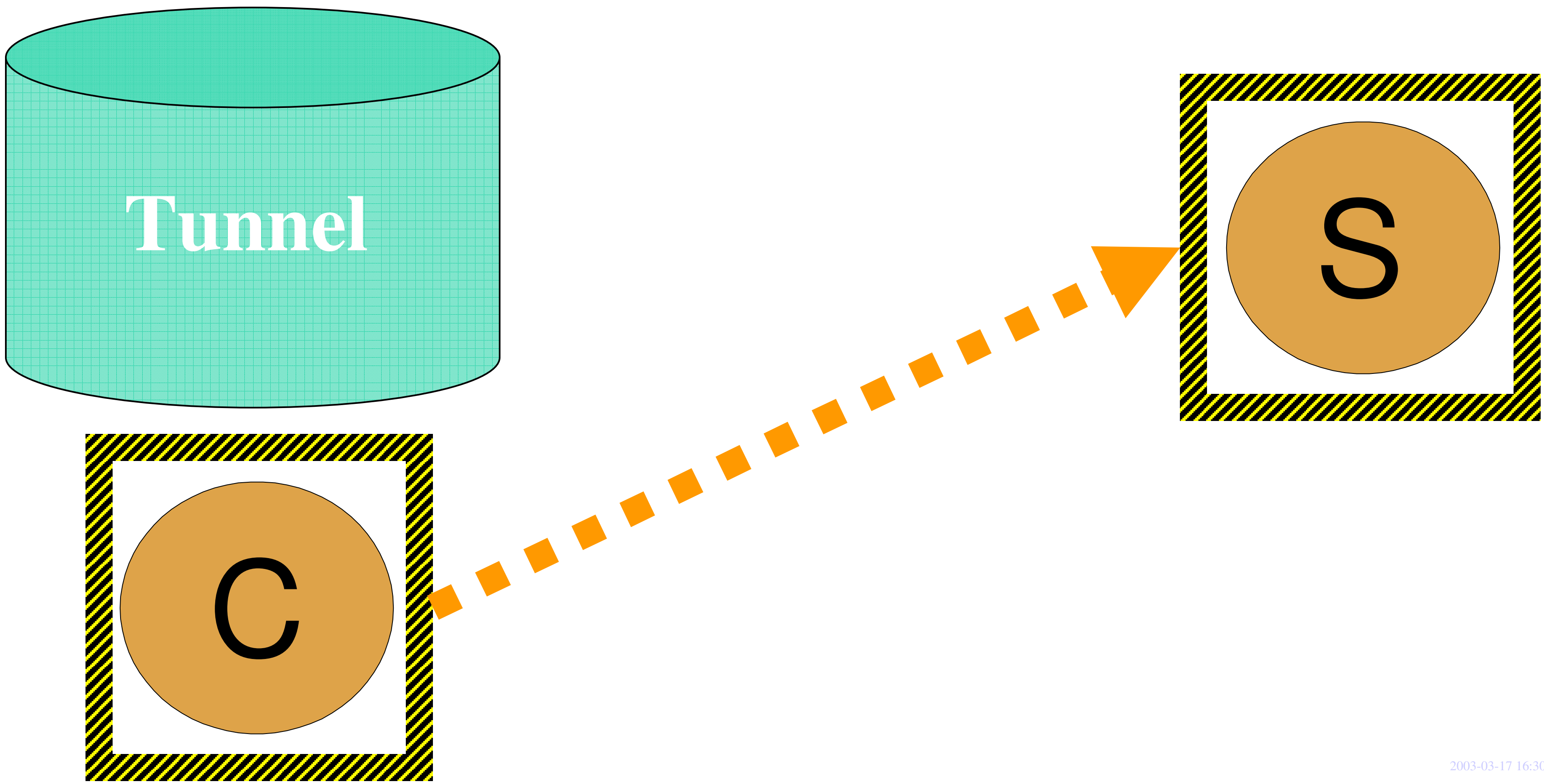
Mobile ("Wireless") Wires



Mobile obstacles

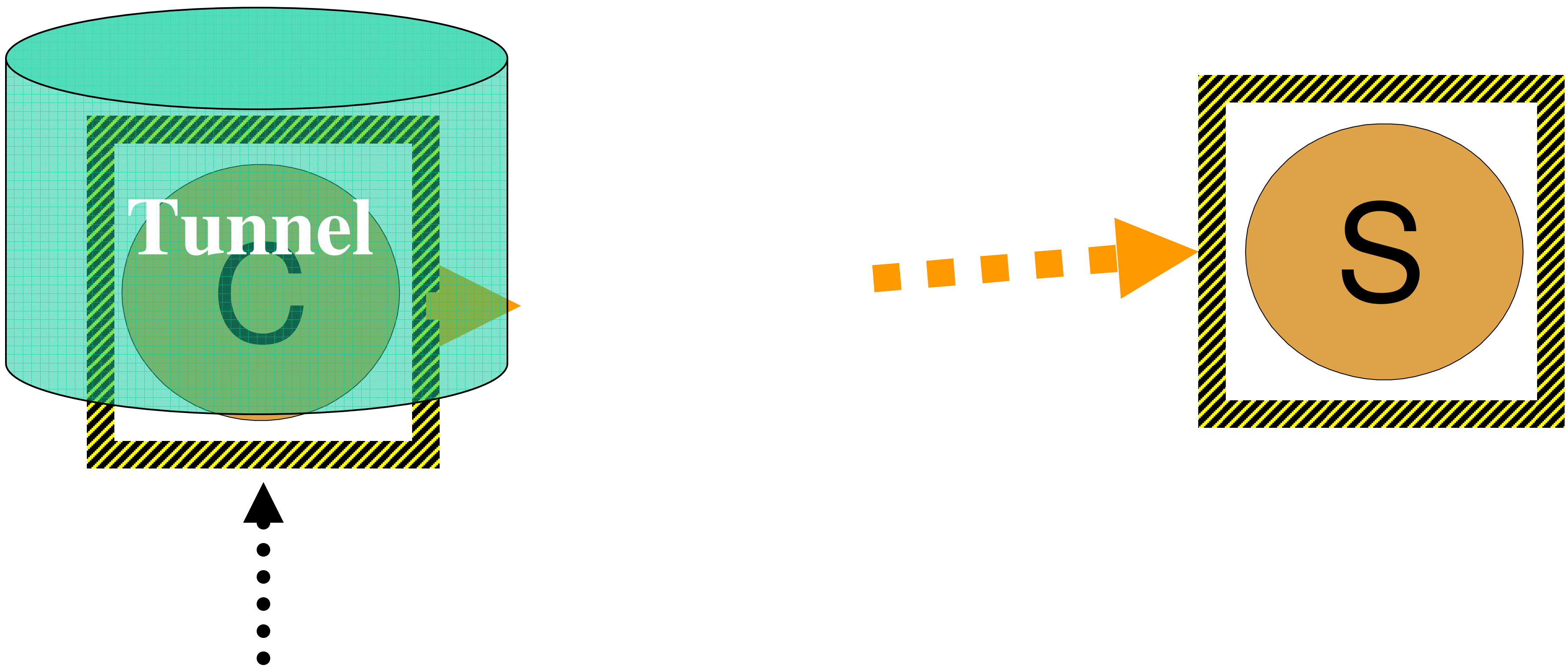
Tunnel Effect

Mobile devices
going around obstacles

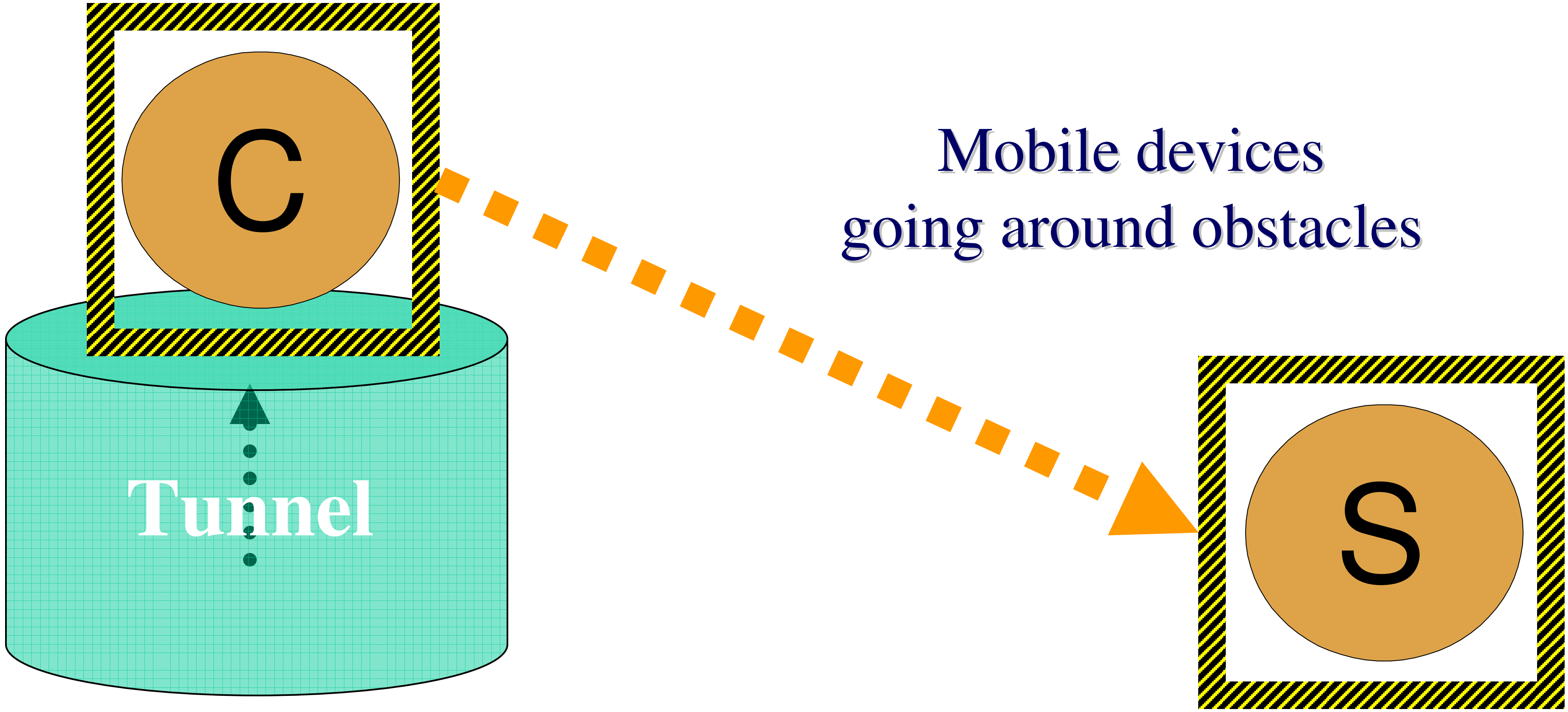


Tunnel Effect

Mobile devices
going around obstacles



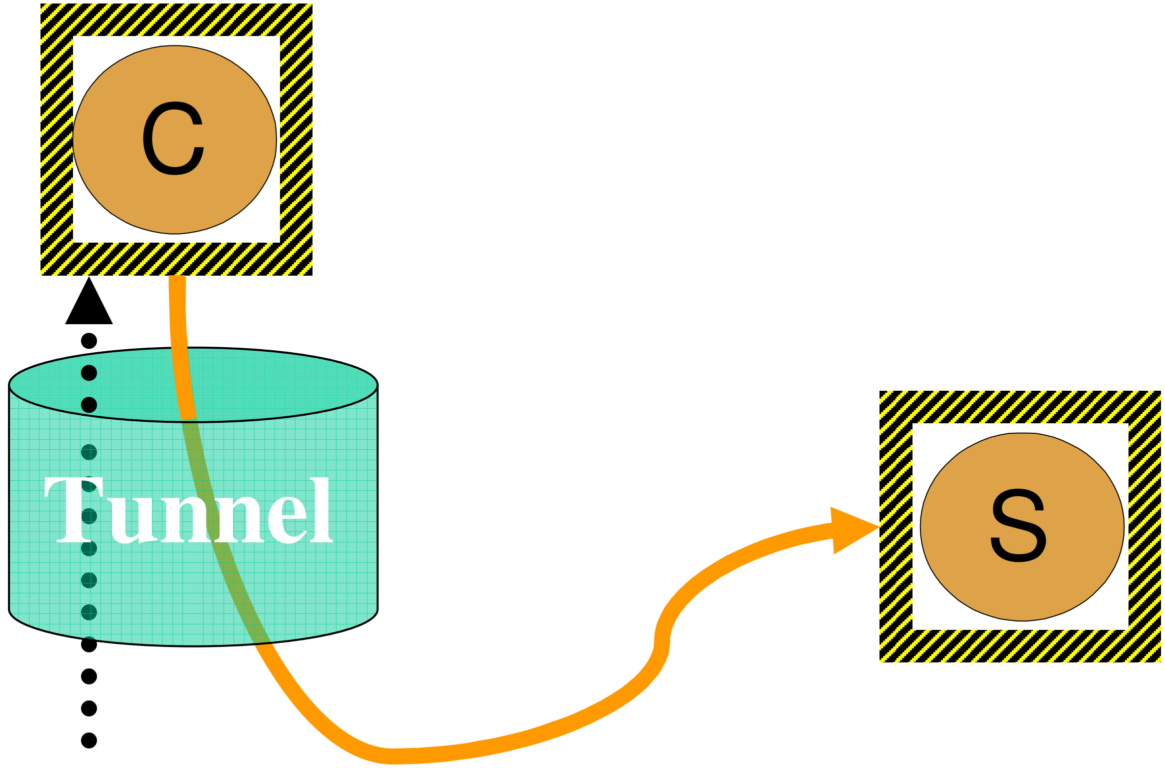
Tunnel Effect



Mobile devices
going around obstacles

Tunnels vs Reliable Communication

- Reliable communication = continuous unbreakable wires



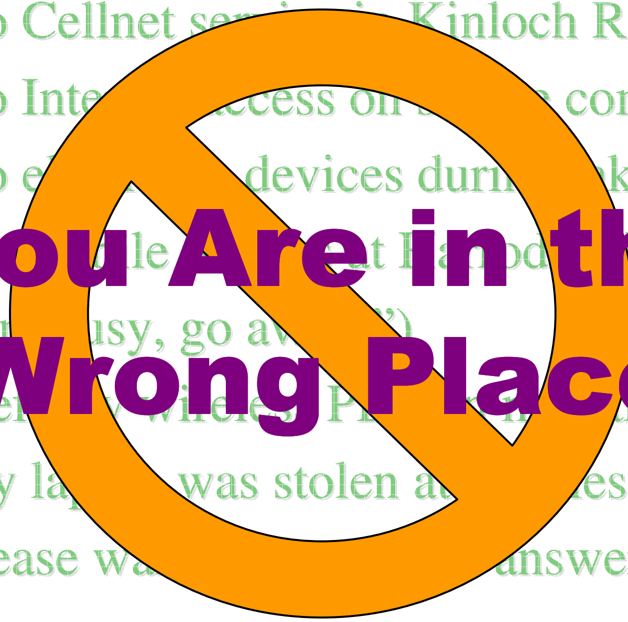
- Reliable communication + Tunnels
  = wires get tangled (and untangling them is hard)
  = eventually one can no longer move (or the wire breaks).

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

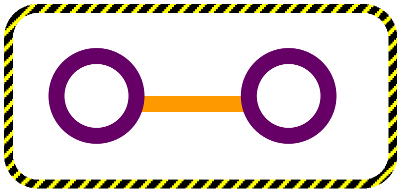
- **No longer to be regarded as a failure**
  It is a state of affairs, due to many causes:
  - Congestion („“The server could not be reached.”)
  - Obstructions (“Infrared device out of sight.”)
  - Geography (“No Cellnet se[illegible] Kinloch Rannoch.”)
  - Security (“No Inte[illegible]ccess o[illegible] computer.”)
  - Safety (“No e[illegible] devices duri[illegible]keoff and landing.”)
  - Policy (“No[illegible]le[illegible]t H[illegible]od[illegible]
  - Privacy (“I ar[illegible]usy, go av[illegible]”)
  - Psyche (“I le[illegible]wireles[illegible]P[illegible]r pants.”)
  - Crime (“My la[illegible]was stolen a[illegible]es De Gaulle’s.”)
  - Physics (“Please wa[illegible]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…

Connectivity Depends on Location

- Proximity:

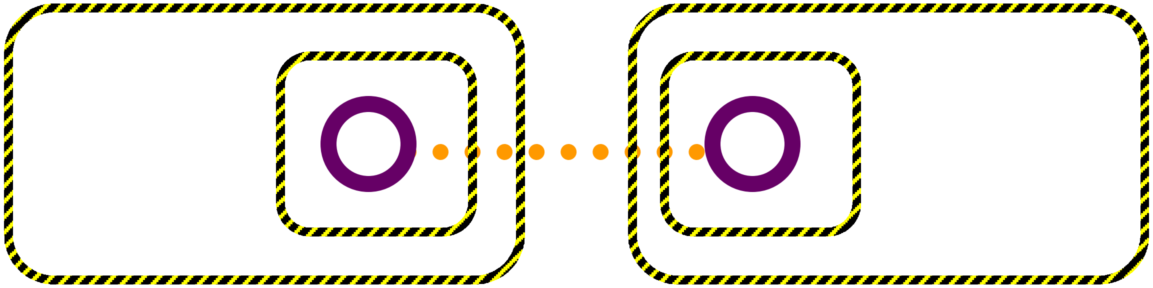


Ok. Fast (bounded delay), reliable, secure.

- 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 (π-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 *locations* where communication is attempted.
  - We need to make the *capability to communicate* independent from the *ability to communicate*.
  - Capability without ability: security by location access control.
  - Ability without capability: security by resource access control.

2. Global Computation

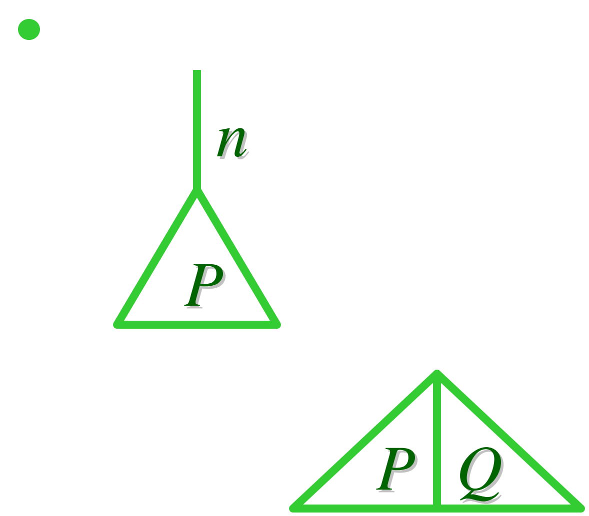
- How do we embed the features and restrictions of global communication in a computational model?
- We must abandon the familiar notion of function call/handshake.
  - We cannot afford to have every function call over the network to block waiting for an answer. (π vs. async-π.)
- We must even abandon the familiar notion of symmetric multi-party (even async) channel communication.
  - We cannot afford to solve consensus problems all the time. (async-π 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. (π 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:

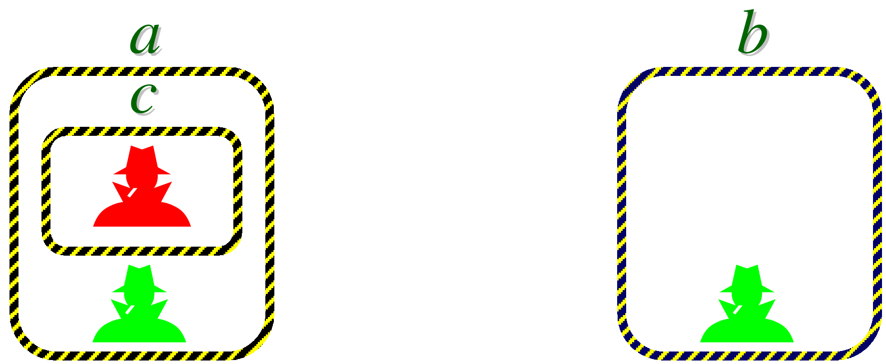
Processes		
$\mathbf{0}$	(void)	
$n[P]$	(location)	
$P \mid Q$	(composition)	

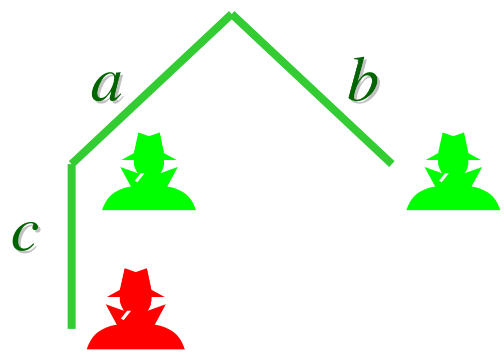
Tree Representation



Mobility

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

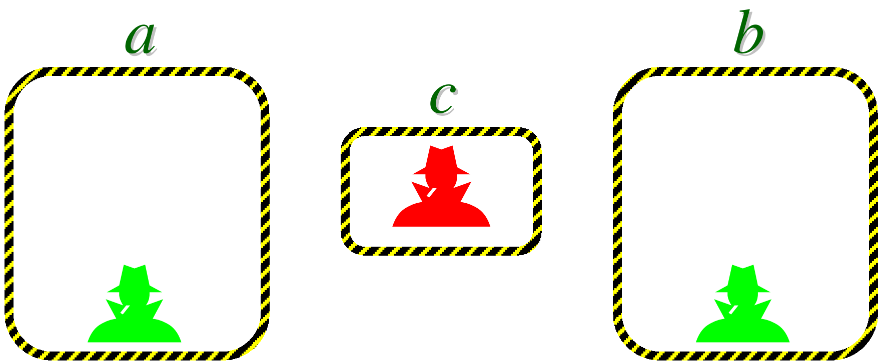


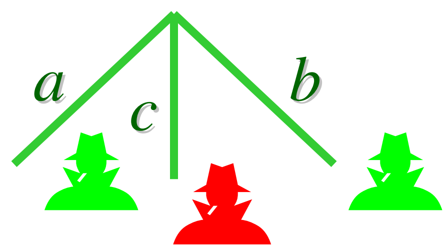


$$a[Q \mid c[out\ a.\ in\ b.\ P]] \qquad \mid b[R]$$

Mobility

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

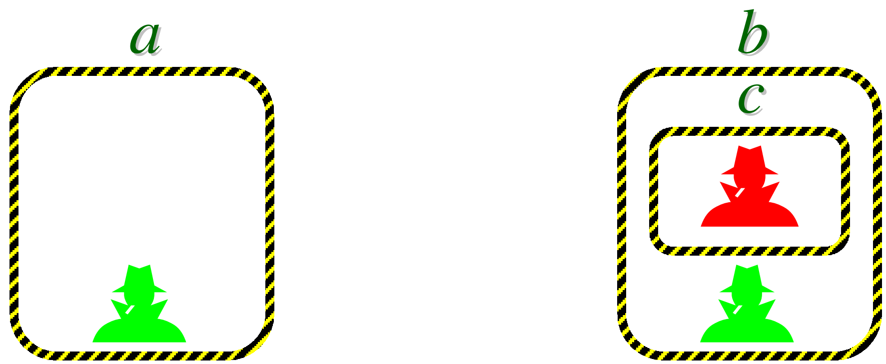


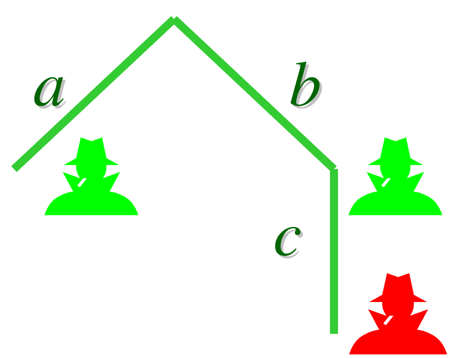


$a[Q] \quad | \; c[\mathit{in}\ b.\ P] \; | \; b[R]$

Mobility

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

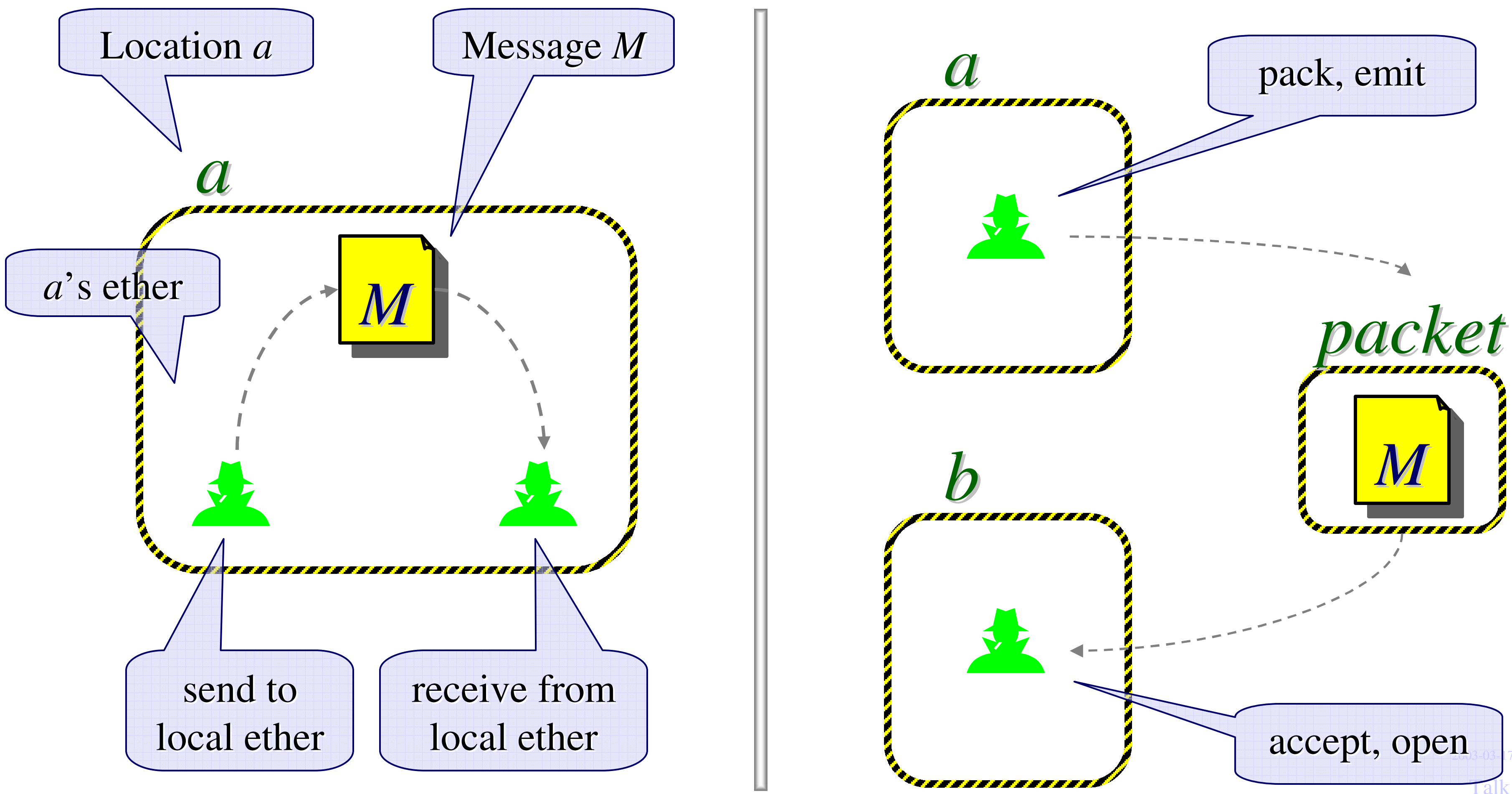




$$a[Q] \quad | \; b[R \mid c[P]]$$

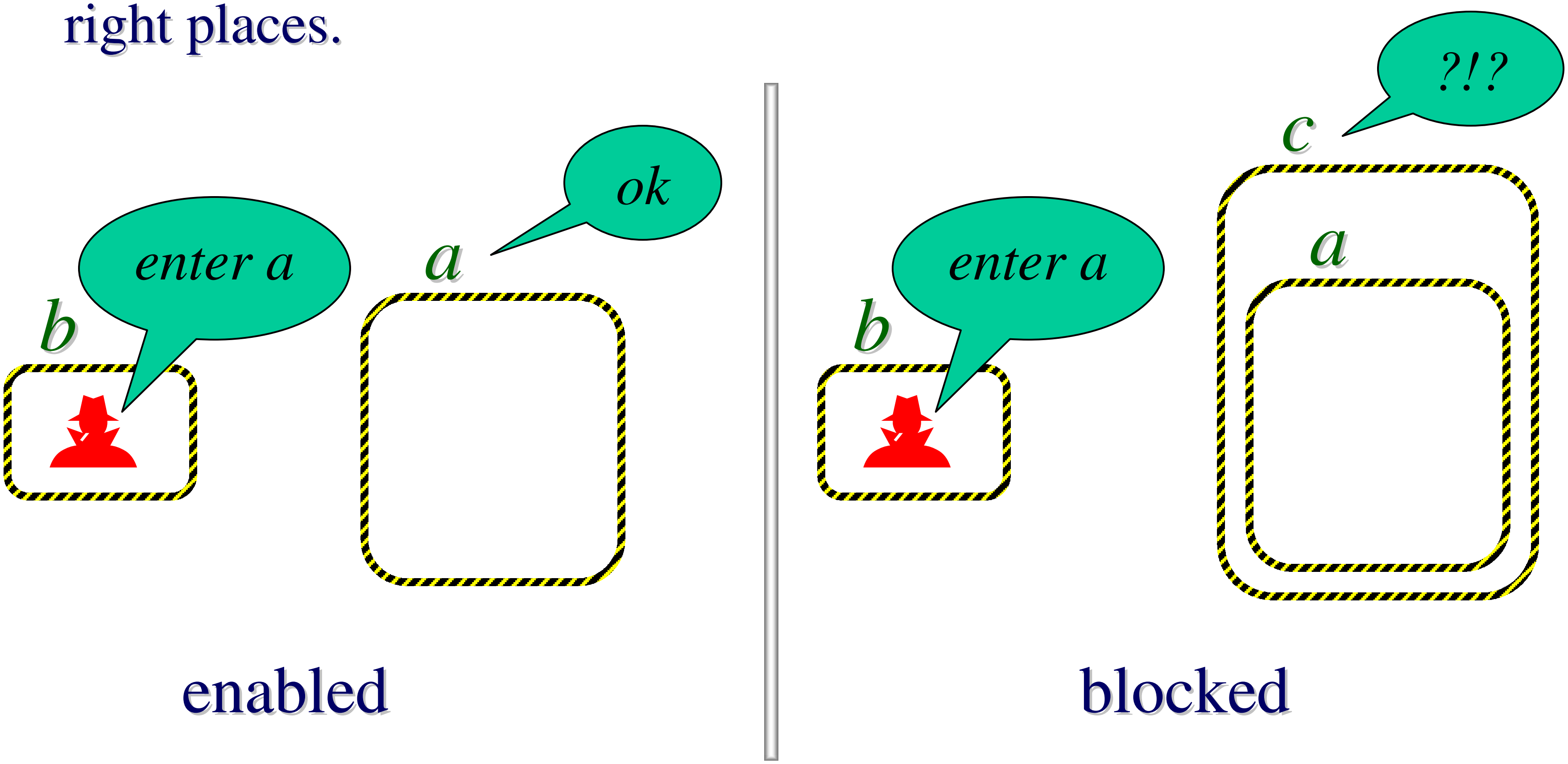
Communication

- 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.
  - π-calculus restriction accounts for private capabilities.
  - As for communication, capabilities can be exercised only the the right places.



Calculi for Concurrency

- 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
  - …

Difference

- Mobility is more general than concurrency
  - One can always use channel communication within each location.
- Mobility is more restrictive than concurrency
  - One cannot have reliable channel communication across locations.

Daring Classification

	Will work fine on a: **LAN** (bounded-delay, integrated management, uniform access)	Will work fine on a: **WAN** (unbounded-delay, federated management, restricted access)
F- (fixed processes and locations)	Calculi (synch/asynch-)π, d-π Infrastructure DOOP Apps File Servers	Calculi π-i, join Infrastructure SOAP, B2C, B2B, P2P Apps Email, Web, Napster
M- (mobile processes or locations)	Calculi d-join Soft Infrastructure AGLETS Soft Apps ? Hard Infrastructure Wireless Ethernet Hard Apps Meeting Trance	Calculi ambients, …, seals Soft Infrastructure Mobile Code Soft Apps Applets, Worms Hard Infrastructure Wireless Telephony Hard Apps Mobile B2C, B2B

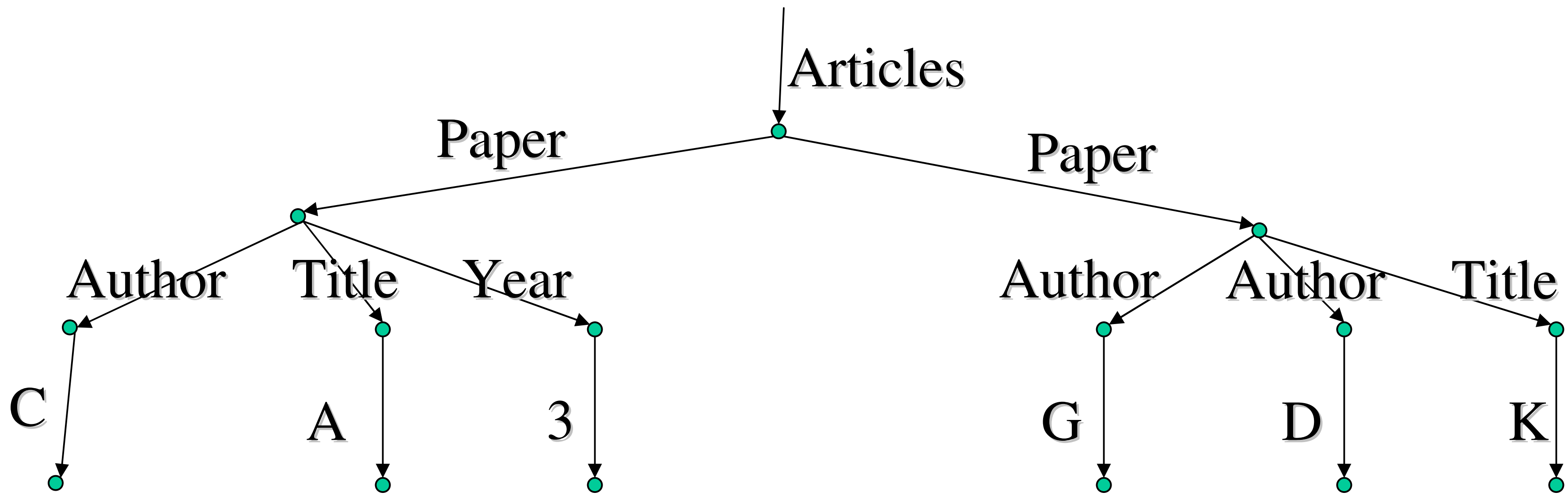
3. Global Languages

- The ambient calculus is a minimal formalism designed for theoretical study. As such, it is not a “programming language”.
- Still, the ambient calculus is designed to match fundamental WAN characteristics.
- By building languages on top of a well-understood WAN semantics, we can be confident that languages will embody the intended semantics.
- We now discuss how ambient characteristics might look like when extrapolated to programming languages.

Global Data

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

  (Abiteboul, Buneman, Suciu: "Data on the Web" Morgan Kaufman'00.)



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

WAN Observable Phenomena

- Physical Locations
  - Observable because of the speed of light limit
  - Preclude instantaneous actions
  - Require mobile code
- Virtual Locations
  - Observable because of administrative domains
  - Preclude unfettered actions
  - Require security model and disconnected operation

- Variable Connectivity
  - Observable because of free-will actions, physical mobility
  - Precludes purely static networks
  - Requires bandwidth adaptability
- Failures
  - Unobservable because of asynchrony, domain walls
  - Preclude reliance on others
  - Require blocking behavior, transaction model

Mobility and Barriers

- Mobility is all about barriers:
  - Locality = **barrier topology**.
  - Process mobility = **barrier crossing**.
  - Security = **(in)ability to cross barriers**.
  - Communication = **interaction within a barrier**.
  - No immediate action at a distance (= across barriers).
- Ambients embed this barrier-based view of mobility, which is grounded on WAN observables.
- A "wide-area language" is one that does not contain features violating this view of computation.

Wide Area Languages

- Languages for Wide Area Networks:
- WAN-sound
  - No action-at-a-distance assumption
  - No continued connectivity assumption
  - No security bypasses
- WAN-complete
  - Able to emulate surfer/roamer behavior
- Some steps towards Wide Area Languages:
  - Ambient Calculus (with Andy Gordon)
  - Service Combinators (with Rowan Davies)

Outline of WAL Features

- No “hard” pointers.
  - Remote references are URLs, symbolic links, or such.
- Migration/Transportation
  - Thread migration.
  - Data migration.
  - Whole-application migration.
- Dynamic linking.
  - A missing library or plug-in may suddenly show up.
- Patient communication.
  - Blocking/exactly-once semantics.
- Built-in security primitives.

Ambients as a Programming Abstraction

- Our basic abstraction is that of mobile computational ambients.
- The ambient calculus brings this abstraction to an extreme, by representing everything in terms of ambients at a very fine grain.
- In practice, ambients would have to be medium or large-grained entities. Ambient contents should include standard programming subsystems such as modules, classes, objects, and threads.
- But: the ability to smoothly move a collection of running threads is almost unheard of in current software infrastructures. Ambients would be a novel and non-trivial addition to our collection of programming abstractions.

Names vs. Pointers

- The only way to denote an ambient is by its name.
  - One may possess a name without having immediate access to any ambient of that name (unlike pointers).
  - Name references are never “broken” but may be “blocked” until a suitable ambient becomes available.
- Uniformly replace pointers (to data structures etc.) by names.
  - At least across ambient boundaries.
  - This is necessary to allow ambients to move around freely without being restrained by immobile ties.

Locations

- Ambients can be used to model both physical and virtual locations.
  - Some physical locations are mobile (such as airplanes) while others are immobile (such as buildings).
  - Similarly, some virtual locations are mobile (such as agents) while others are immobile (such as mainframe computers).
- Mobility distinctions are not part of the basic semantics of ambients.
  - Can be added as a refinement of the basic model, or
  - Can be embedded in type systems that restrict the mobility of certain ambients.

Migration and Transportation

- Ambients offer a good paradigm for application migration.
  - If an ambient encloses a whole application, then the whole running application can be moved without need to restart it or reinitialize it.
  - In practice, an application will have ties to the local window system, the local file system, etc. These ties, however, should only be via ambient names.
  - After movement the application can smoothly move and reconnect its bindings to the new local environment. (Some care will still be needed to restart in a good state).

Communication

- The communication primitives of the ambient calculus (local to an ambient) do not support global consensus or failure detection.
- These properties should be preserved by any higher-level communication primitives that may be added to the basic model, so that the intended semantics of communication over a wide-area networks is preserved.
  - RPC, interpreted as mobile packets that transport and deposit messages to remote locations.
  - Parent-child communication
  - Communication between siblings.

Synchronization

- The ambient calculus is highly concurrent.
  - It has high-level synchronization primitives that are natural and effective (as shown in the examples).
  - It is easy to represent basic synchronization constructs, such as mutexes:

    $release\ n; P \quad \triangleq \quad n[\,] \mid P$

    release a mutex called n, and do P

    $acquire\ n; P \quad \triangleq \quad open\ n.\ P$

    acquire a mutex called n, then do P

- Still, additional synchronization primitives are desirable.
  - A useful technique is to synchronize on the change of name of an ambient:

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

  - (See also the Seal calculus by Castagna and Vitek.)

Static and Dynamic Binding

- The names of the ambient calculus represent an unusual combination of static and dynamic binding.
  - The names obey the classical rules of static scoping, including consistent renaming, capture-avoidance, and block nesting.
  - The navigation primitives behave by dynamically binding/linking a name to any ambient that has the right name.
- Definitional facilities can similarly be derived in static or dynamic binding style. E.g.:
  - Statically bound function definitions.
  - Dynamically bound resource definitions.

Modules

- An ambient containing definitions is similar to a module/class.
  - Remote invocation is like qualified module access.
  - open is like inheritance.
  - copy is like object generation from a prototype.
- Unusual "module" features:
  - Ambients are first class modules: one can choose at run time which particular instance of a module to use.
  - Ambients support dynamic linking: missing subsystems can be added to a running system by placing them in the right spot.
  - Ambients support dynamic reconfiguration. Module identity is maintained at run time. The blocking semantics allows smooth suspension and reactivation. The dynamic hierarchical structure allows replacement of subsystems.

Security

- Ambient security is based on boundaries and capabilities, as opposed to a cryptography, or access-control.
- These three models are all interdefinable. In our case:
  - Access control is obtained by using ambients to implement RPC-like invocations that have to cross boundaries and authenticate every time.
  - Cryptography is obtained by interpreting ambient names (by assumption unforgeable) as encryption keys.
- The ambient security model is high level.
  - It maps naturally to administrative domains and sandboxes.
  - It allows the direct discussion of virus, trojan horses, infection of mobile agents, firewall crossing, etc.

4. Summary

- Global Communication
  - Broadens communication mechanisms.
  - But also restricts the ways in which we can communicate.

    "Connected anytime anywhere to anything." 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".

- Global Languages
  - The fundamental observables have changed.
  - Languages must change as well.

    "I'll just use Pascal to write a mail server." NOT!

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 infrastucture(s), both practical and theoretical, that will make all this easy.