Global Computing

Luca Cardelli

Microsoft Research

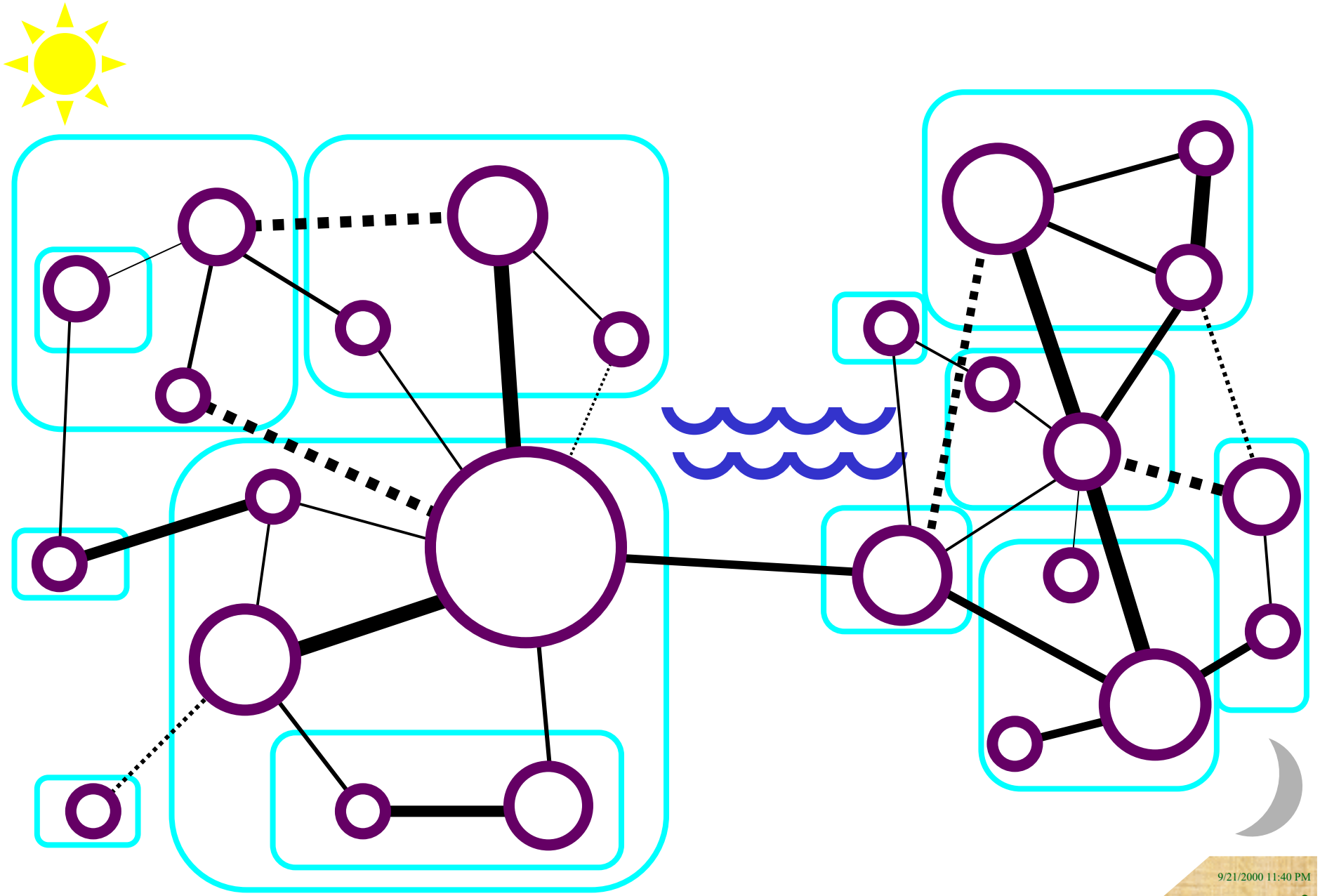
Edinburgh, 2000-09-21..22

Introduction

- We are building infrastructure that allows us to be connected “everywhere all the time”.
  - Global wireless speech and data network.
  - Local/reactive/synchronous.
- At the same time, we are building infrastructure that allows us to be isolated and protected from intrusion.
  - Universal mailboxes, spam filters, Faraday cages, firewalls.
  - Remote/deferred/asynchronous.
- We cannot have it both ways. We will have to describe what we want to be “local”, and we will have to adapt to what must necessarily be “remote”.

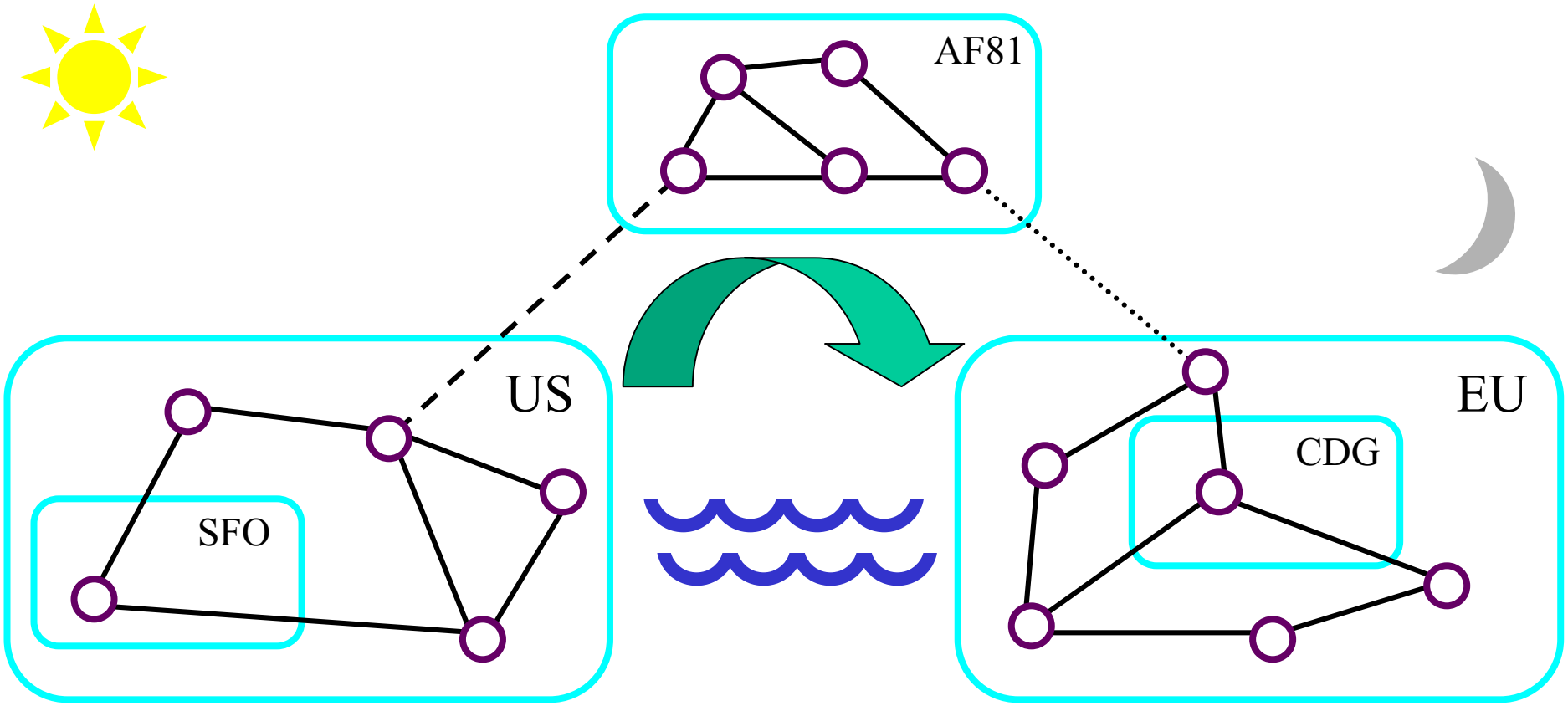
Ping globally, act locally.

Wide Area Networks



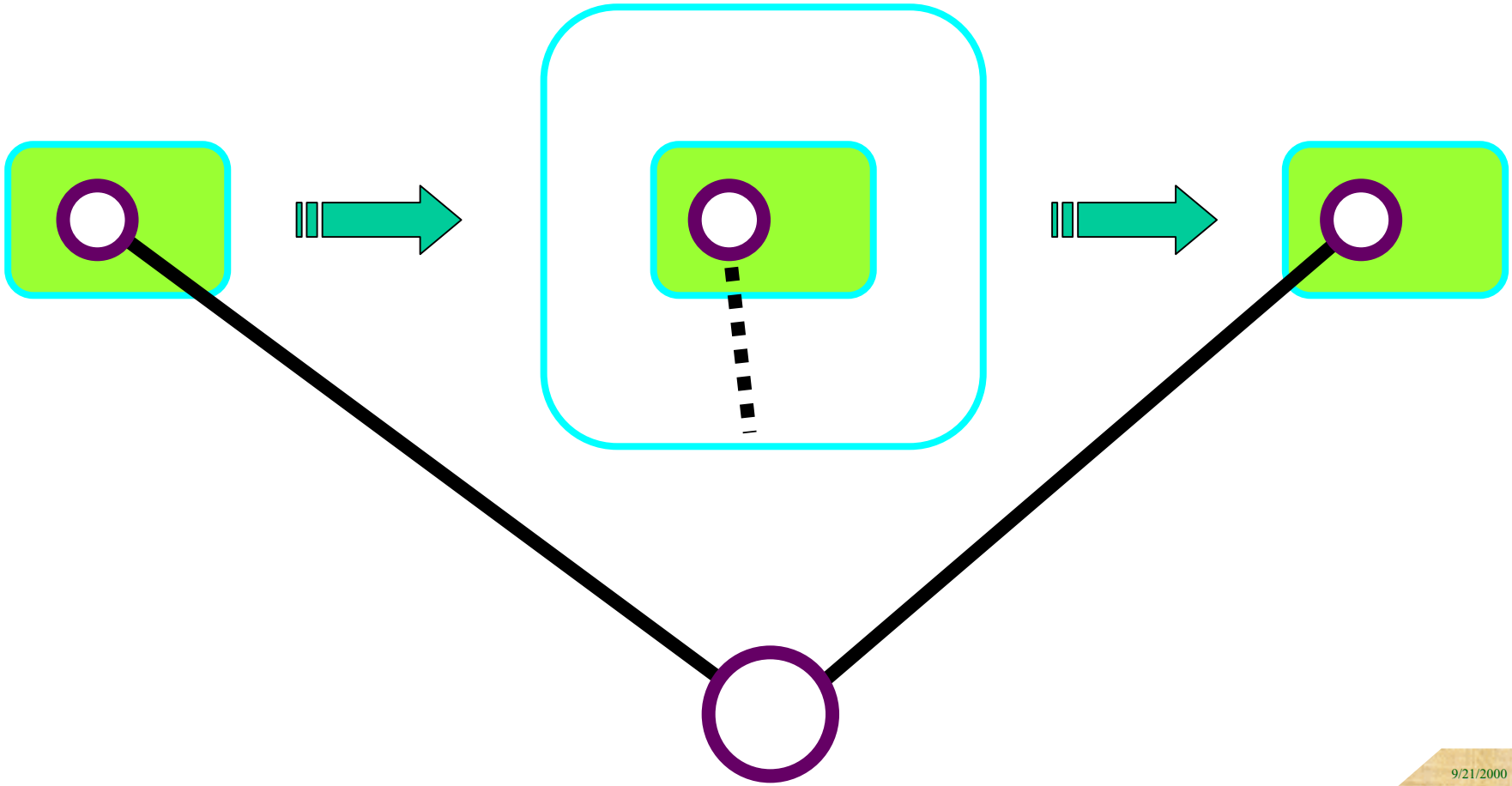
Mobile Computing

- (Software) Active computations move around.
- (Hardware) Mobile devices transport active computations.



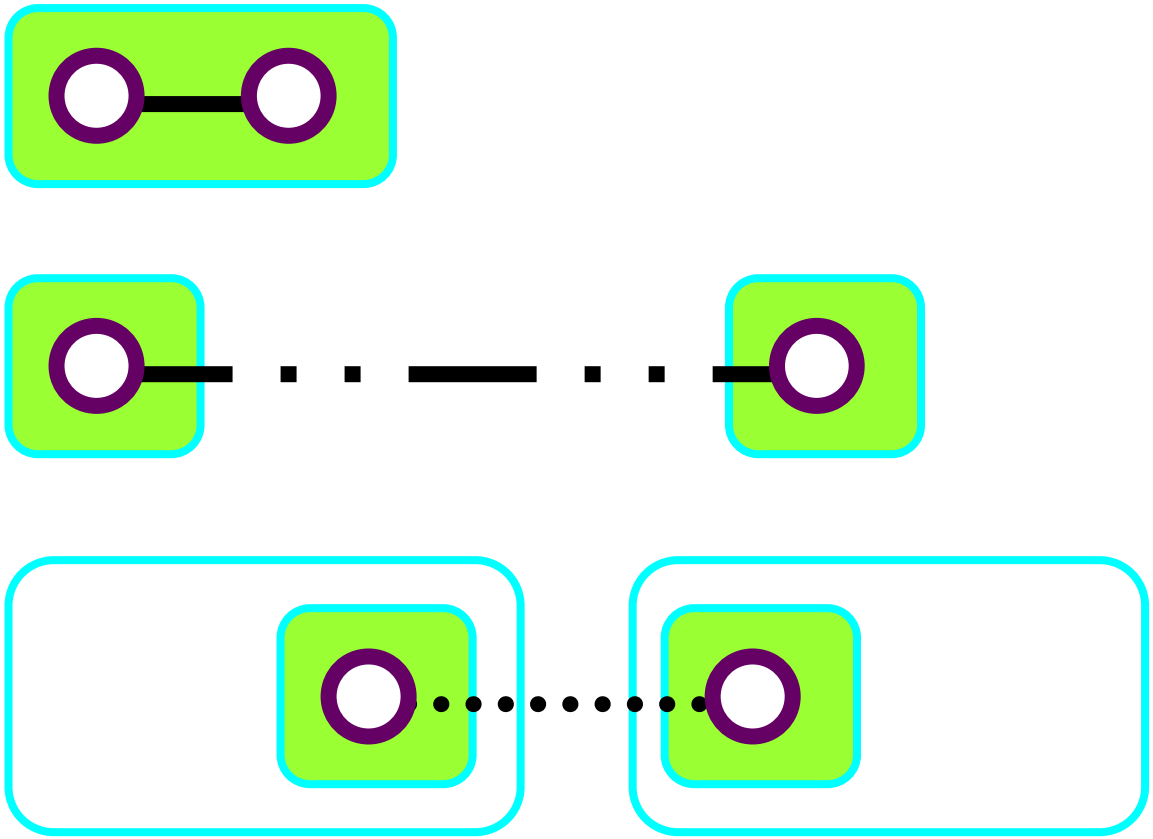
Connectivity Depends on Location

- Tunneling.
  - Accidental disconnection (bad infrastructure, solar flares).
  - Intentional disconnection (privacy, security, quiet).



Connectivity Depends on Proximity

- No remote real-time control.
- No secure (unencrypted) remote links.



Globally Reactive: Mars Probe

- 8 minutes light speed delay to Mars.
  - Unpredictable real-time conditions (aerobraking, landing).
  - No remote real-time control (fundamental physical limit).
  - Async communication (no RPC!); local reactive intelligence.
- Takes years to get there.
  - Needs to upload software along the way (“agents”).
  - Needs to upgrade communication protocols along the way.
- Billions of dollars, but still no guaranteed connectivity.
  - Power-saving modes.
  - Planetary occlusions.
  - Heath shields.
  - Alien intervention, metric conversions…

Some Approaches

- Agents (Telescript, etc.).
  - Move globally, coordinate locally.
- Spaces (Linda, etc.).
  - Communicate globally, coordinate locally.
- Dynamic hierarchies (Ambients, D-Join, etc.).
  - Various synchrony assumptions; wide spectrum between D-Join, Ambients, and Seal.
- Distributed transactions, Workflow, B2B, ...
  - Various unconventional but creative models.

- Most of these have some inner elegance, but also many open questions in terms of theoretical expressiveness and practical feasibility on global scales.

How Much Asynchrony?

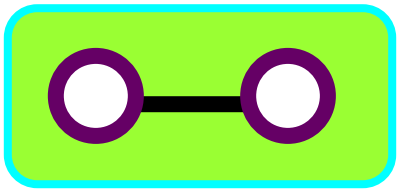
- Communication: 2 choices
  - *Send*: synchronous or asynchronous
    - ***send*** *ch msg. P*
    - ***send*** *ch msg*
  - *Receive*: always synchronous
    - ***receive*** *ch x. P*
- Mobility: 4 choices
  - *Go*: synchronous or asynchronous
    - ***go in*** *server. P*
    - ***go in*** *server*
  - *Let*: synchronous or asynchronous
    - ***let*** *agent* ***in***. *P*
    - ***let*** *agent* ***in***
- What is the expressive power of various combinations?

How Can/Should Things Interact?

- Remote interaction: Asynchronous only.
  - No instantaneous remote communication.
  - No instantaneous remote mobility.
  - But we cannot make everything asynchronous...
- Local interaction: Synchronous is better.
  - Synchronous local communication: good for awareness of communication for both parties (sender and receiver) and particularly good for reactive/real-time systems.
  - Synchronous local mobility: good for awareness of movement for both parties (agent and host).
- They fit together (global+reactive).
  - Mobility can be used to turn remote/asynch into local/synch.
    (E.g.: send agent to do real-time control of Mars probe)
- But exactly how far is remote?
  - Parent, children, siblings, grandparents, grandchildren, cousins…

Local Interaction

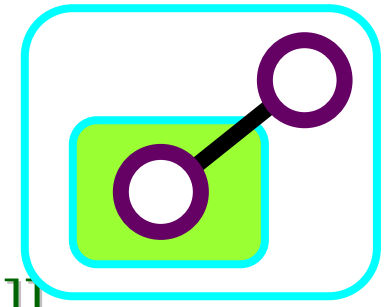
- Local Communication
  - No-brainer: synchronous (shared ether).
  - Can *everything else* be asynchronous?
- (No “Local Mobility”.)



Interaction with Parent

- Communication with parent:
  - Async: may deliver message to the wrong place.

    parent[…
      child[***send*** *parent* “going out” | ***go out*** *parent*]]
  - So, we need synchronous parent communication.



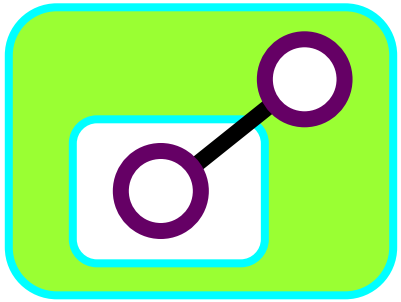
- Mobility w.r.t. parent:
  - Parent wants to control permission for children to exit.

    parent[*child*[***go out*** *parent*] | ***let*** *child* ***out***]
  - Parent may be moving and does not want children to get lost by exiting at random places.

    parent[***go*** *childcare*. ***let*** *child* ***out***. ***go*** *work* | *child*[..]]
  - So, we need mobility w.r.t. parent to be synchronous.

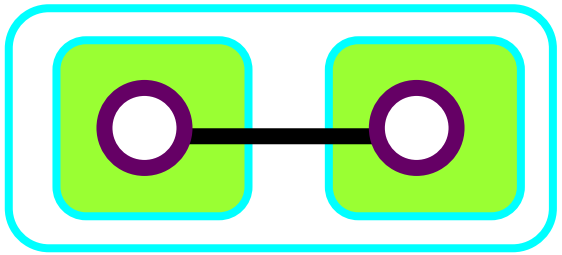
Interaction with Children

- Communication with children.
  - A parent wants to deliver a message to a children before giving it permission to exit:
    parent[***send*** *child “be back at 10”*. ***let*** *child* ***out*** | *child*[..]]
  - So, communication with children should be synchronous.
- Mobility w.r.t. children (*objective move*).
  - Want to move something before doing something else:
    parent[***put*** *cat* ***out***. *HaveDinner* | *cat*[…]]
  - So, this should be synchronous too.



Interaction with Siblings

- Communication with siblings.
  - An agent may want to leave a message for another agent to pick up later. Asynchronous interaction.
  - An agent may want to know that another agent is present. Synchronous interaction.



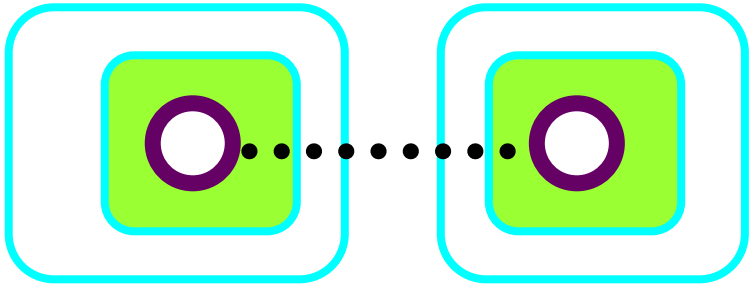
- Mobility w.r.t. siblings.
  - Synchronous:

    passenger[***go in*** *taxi*] | *taxi*[***let*** *passenger* ***in***. *Depart*]
  - Asynchronous:

    passenger[***go in*** *train*] | *train*[(!***let*** *passenger* ***in***) | *Depart*]

Interaction with Grandparents, Grandchildren ...

- Communication with distant relatives.
  - Synchrony here imposes too many constraints on remote sites (too much distributed locking), so it should be asynchronous.
- Mobility w.r.t. distant relatives.
  - Synchrony here has the flavor of instantaneously crossing multiple firewalls, so long-range movement should be asynchronous (step-by-step).



Conclusions (?)

- Global mobility/communication primitives
  - How many primitives are useful?
  - How many primitives are necessary?
- Synchrony vs. asynchrony
  - Gradual transition from synch to asynch depending on distance.
  - What degree of synchrony is required in theory?
  - And in practice?
- Static properties of mobile systems (for reliability, security…)
  - Type systems
  - Analysis systems
  - Program logics

Panel Discussion

- The Killer App for global computing is Business-to-Business E-commerce.
  - Global typed data (XML-based).
  - Global interaction protocols (HTTP-based).
  - Concurrency required: at least π-calculus style.
  - Mobility required: mobile data access, transportable functionality.
  - Cooperative but with $$ involved. (Reasonable security.)
  - Asynchronous (order shipment) and reactive (credit verification).
  - Reliability and resilience issues; huge databases, huge traffic.
- Europe is way behind.
  - 3 dog years in B2C, possibly more in B2B.
  - This will have a major impact on the efficiency of the general economy.