Understanding Communication and Concurrency through Types

Kohei Honda (Queen Mary, University of London)

Nobuko Yoshida (Imperial College London)

April 18, 2012

Milner Symposium, Edinburgh

In collaboration with:

Matthew Arrott (OOI)

Gary Brown (Red Hat)

David Frankel (SAP)

Stephen Henrie (OOI)

Matthew Rawlings (ISO TC68 WG4/5)

Alexis Richardson (RabbitMQ/VMware)

Steve Ross-Talbot (Cognizant)

and all our academic colleagues, including our team.

Building and Understanding [Milner 97]

- Computer science is about both understanding and building.
- We understand computing through theories.
- We use that understanding for building.

Standpoints

- The hardware and software environments have started to make it practical to program with communication.
- Communication is a great basis for programming concurrency and distribution.
- It is good to have a general, formal basis for software, especially for communication and concurrency.

Why Communication?

- Communication [Hewitt 77, He & Josephs & Hoare 90, Boudol 91, H & Tokoro 91, Amadio & Castellani & Sangiorgi 96] is an economical way to share data.
- Read/Write is communication ($\sim$ 300 cycles).
- Communication is usable over different scales (from intra-chip to cross-continents).
- Communication is expressive: can represent functions, objects, shared variable concurrency, .. [Milner 73, Hoare 78]
- Communication is explicit: it gives us a notion of "behaviour".

Structuring Sequential Programs

"Our intellectual powers are rather geared to master static relations. (...) For that reason we should do (as wise programmers aware of our limitations) our utmost to shorten the conceptual gap between the static program and the dynamic process, to make the correspondence between the program (spread out in text) and the process (spread out in time) as trivial as possible." [Dijkstra 1968]

Structured vs Unstructured

```
Structured:
IF x<=y THEN
  BEGIN
     z := y-x;
     q :=SQRT(z);
  END
ELSE
  BEGIN
     z := x-y;
     q := -SQRT(z)
  END;
WRITELN(z,q);
```

```
Unstructured:
IF x>y THEN GOTO 2;
z := y-x;
q := SQRT(z);
GOTO 1;
2: z:= x-y;
q:=-SQRT(z);
1: writeln(z,q);
```

Why Types?

- Object:

```
int foobar(String s, int i) { ... }
```

- Function:

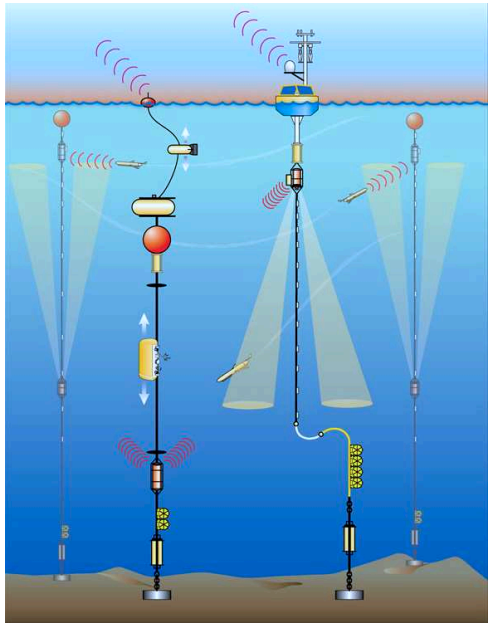
```
Bool -> (Nat -> Nat)
```

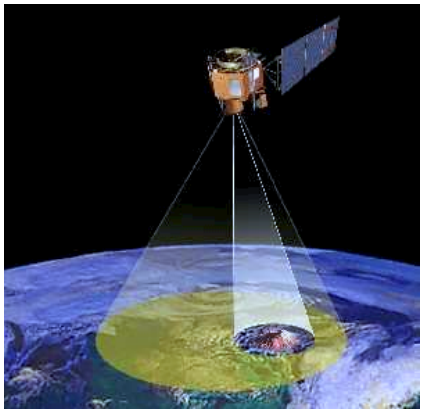
- Benefits:
  - clear, robust interface.
  - efficient static/dynamic checking.
  - gives an effective footing for refined specification and verification.

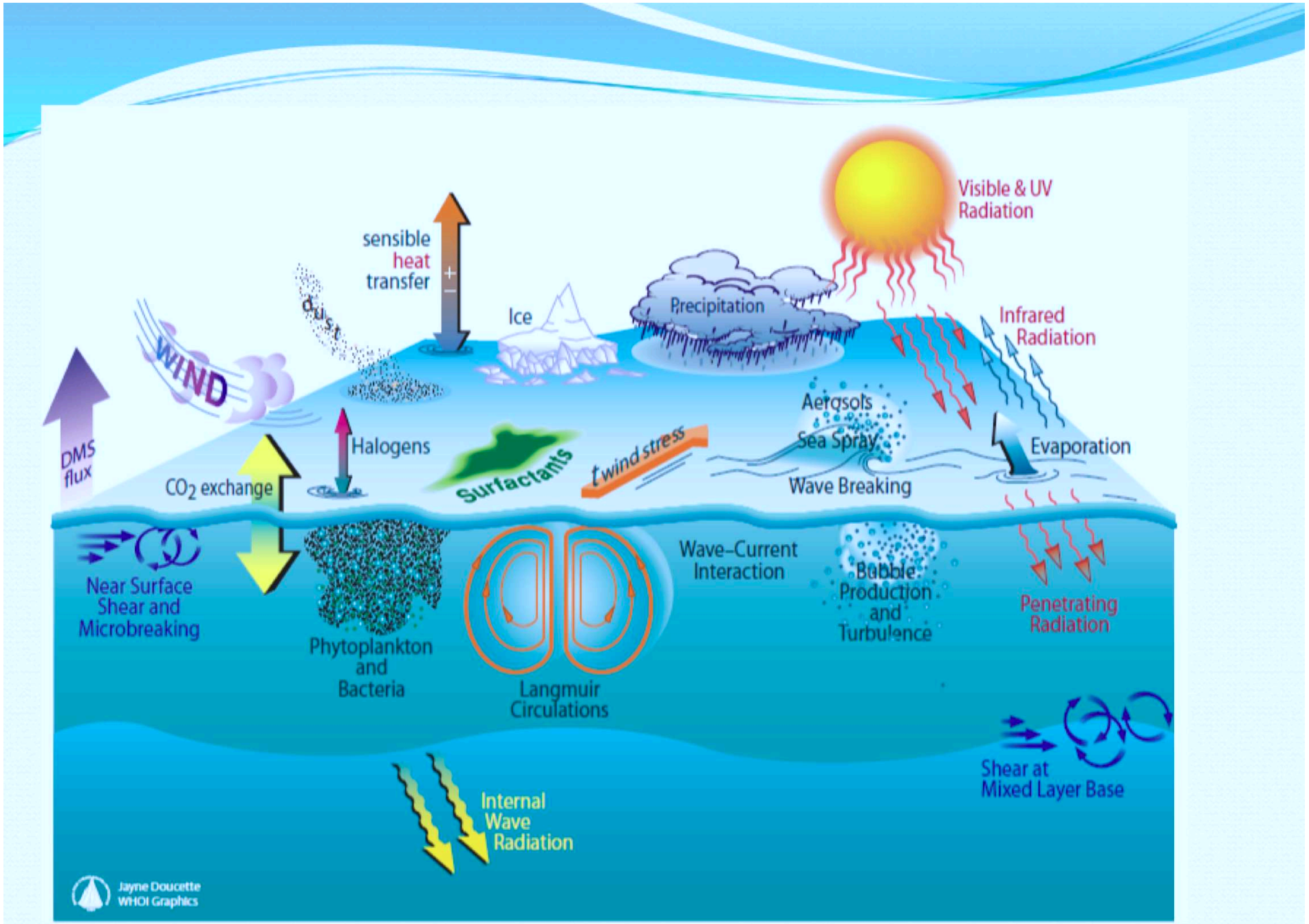
  ... all because of their direct linkage to dynamics of a given computing paradigm.

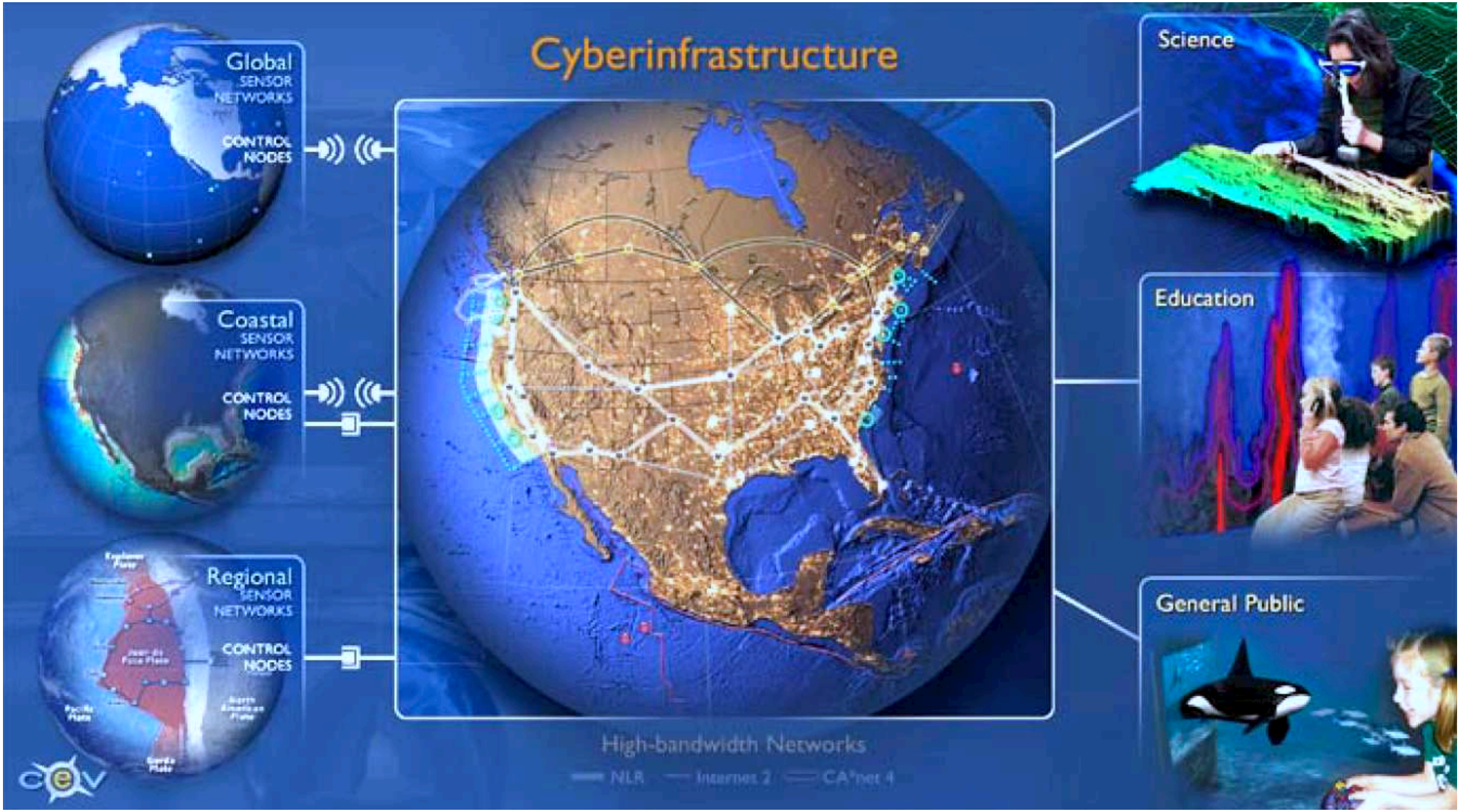
Case Study: Ocean Observatories Initiatives

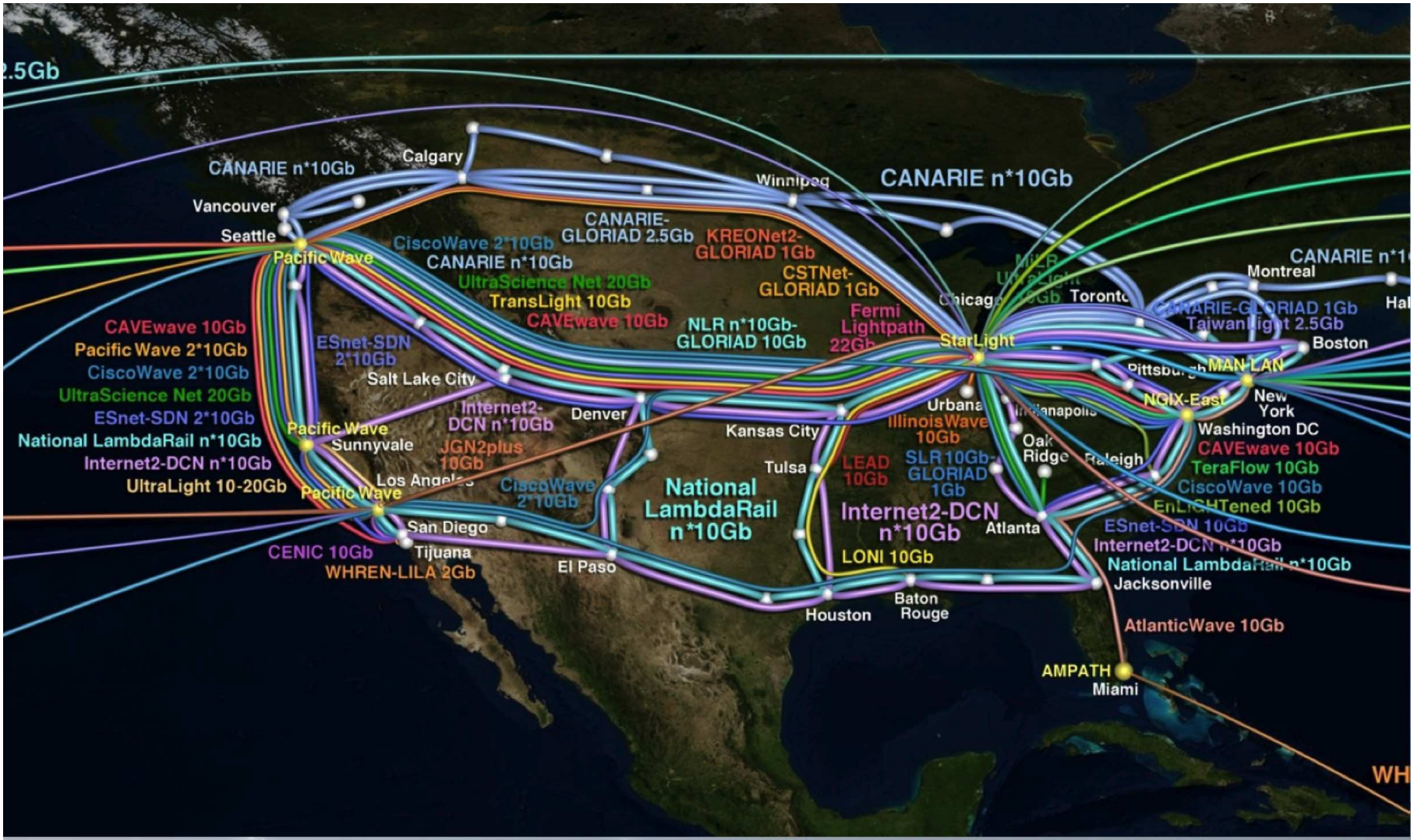
- A NSF project to build a cyberinfrastructure for observing oceans in US and beyond, with usage span of 30 years.
- Integrate real-time data acquisition, processing and data storage for ocean research (e.g. sensor arrays, underwater gliders, ...).



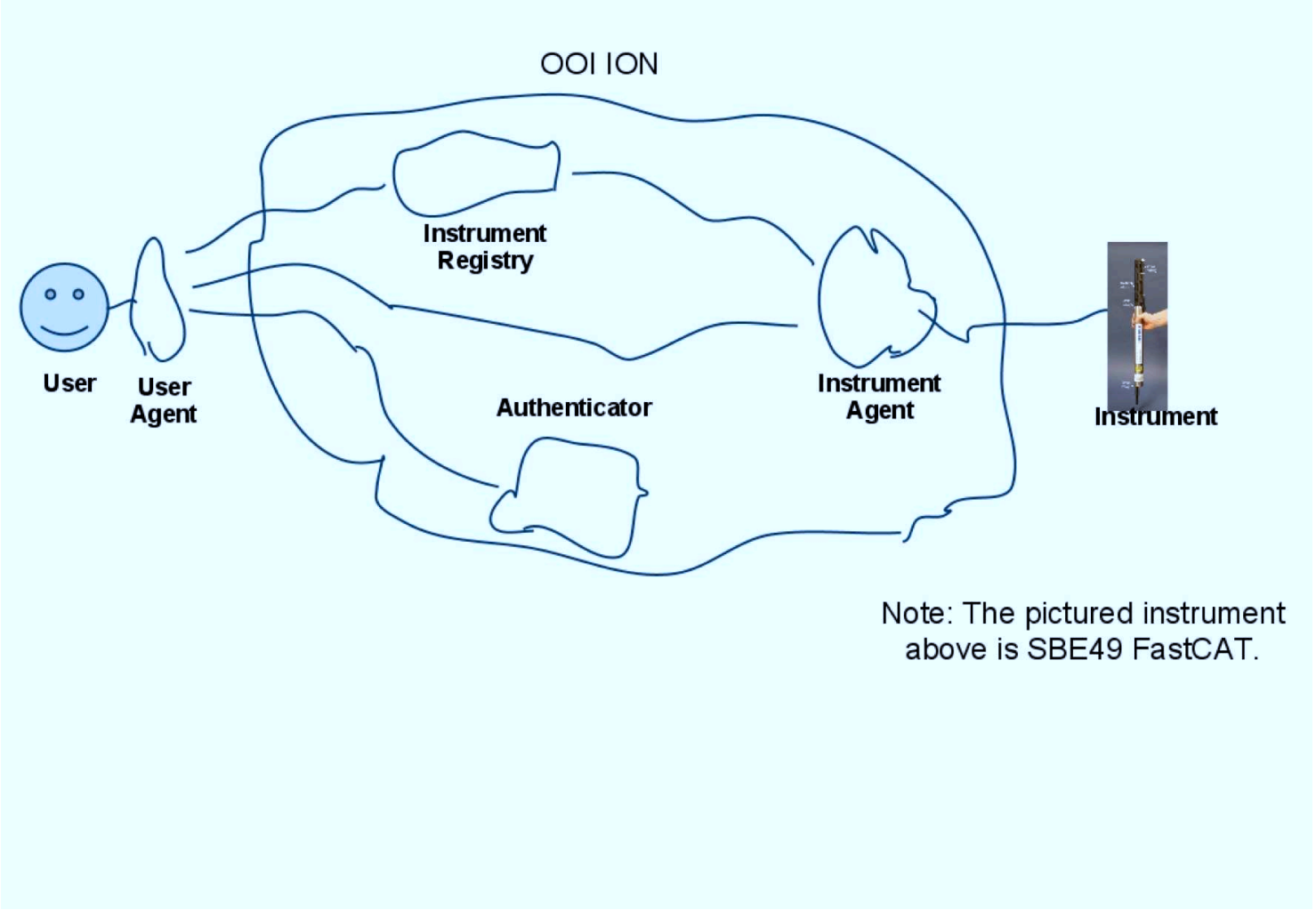



Visible & UV Radiation
sensible heat transfer
+
−
Ice
Precipitation
Infrared Radiation
WIND
dust
DMS flux
Halogens
Aerosols
Sea Spray
Evaporation
CO2 exchange
Surfactants
t wind stress
Wave Breaking
Near Surface Shear and Microbreaking
Wave–Current Interaction
Bubble Production and Turbulence
Penetrating Radiation
Phytoplankton and Bacteria
Langmuir Circulations
Shear at Mixed Layer Base
Internal Wave Radiation
Jayne Doucette
WHOI Graphics

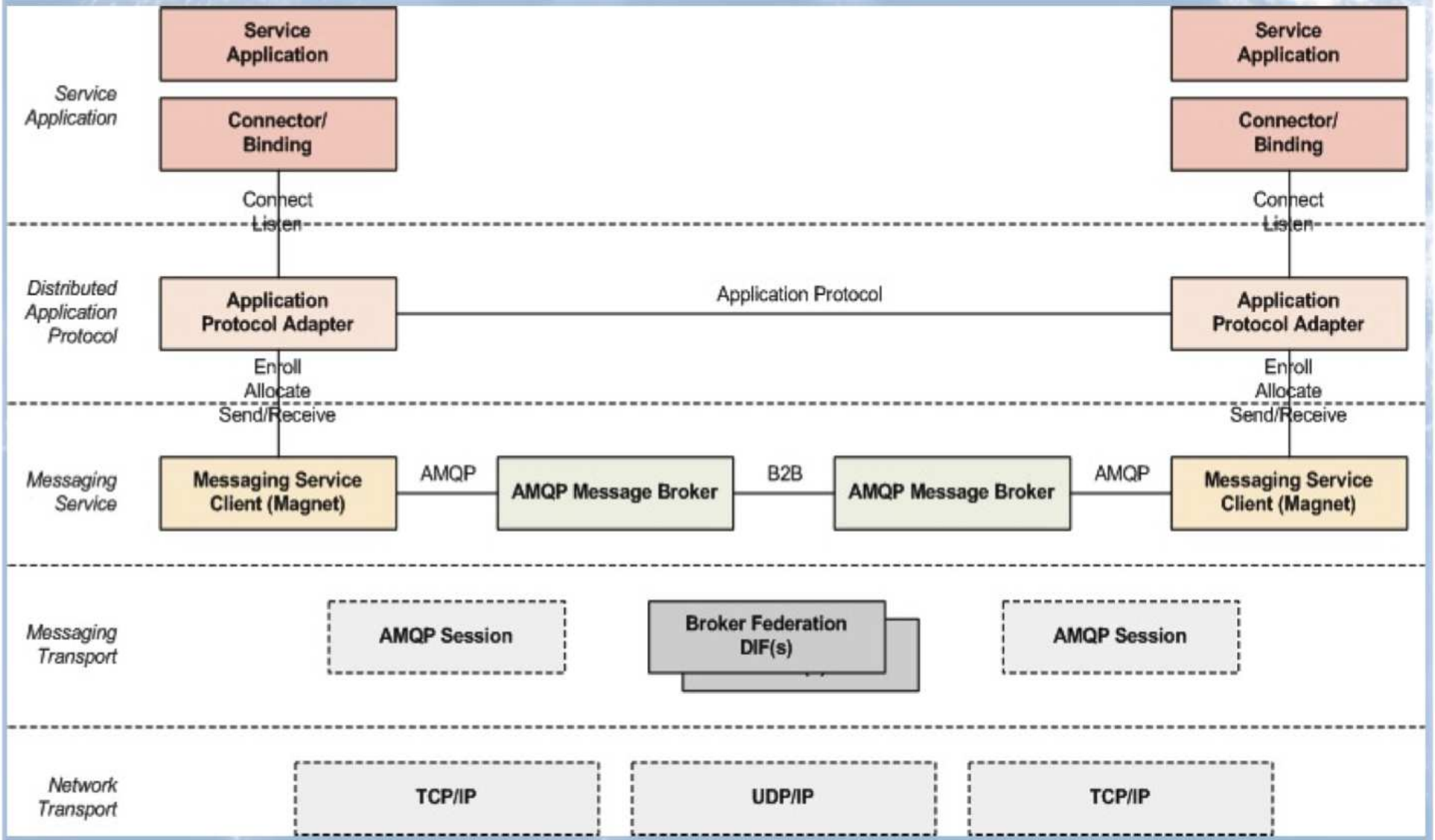

Cyberinfrastructure
Global
SENSOR
NETWORKS
CONTROL
NODES
Coastal
SENSOR
NETWORKS
CONTROL
NODES
Regional
SENSOR
NETWORKS
CONTROL
NODES
High-bandwidth Networks
NLR
Internet 2
CA*net 4
Science
Education
General Public


2.5Gb
CANARIE n*10Gb
Calgary
Winnipeg
CANARIE n*10Gb
Vancouver
Seattle
Pacific Wave
CANARIE-GLORIAD 2.5Gb
KREONet2-GLORIAD 1Gb
CiscoWave 2*10Gb
CANARIE n*10Gb
UltraScience Net 20Gb
TransLight 10Gb
CAVEwave 10Gb
CSTNet-GLORIAD 1Gb
Fermi Lightpath 22Gb
NLR n*10Gb-GLORIAD 10Gb
CAVEwave 10Gb
Pacific Wave 2*10Gb
CiscoWave 2*10Gb
UltraScience Net 20Gb
ESnet-SDN 2*10Gb
National LambdaRail n*10Gb
Internet2-DCN n*10Gb
UltraLight 10-20Gb
ESnet-SDN 2*10Gb
Salt Lake City
Internet2-DCN n*10Gb
Denver
Pacific Wave
Sunnyvale
JGN2plus 10Gb
Los Angeles
Pacific Wave
CiscoWave 2*10Gb
San Diego
Tijuana
CENIC 10Gb
WHREN-LILA 2Gb
El Paso
Kansas City
Tulsa
National LambdaRail n*10Gb
Houston
Baton Rouge
LEAD 10Gb
SLR 10Gb-GLORIAD 1Gb
Internet2-DCN n*10Gb
LONI 10Gb
Atlanta
Urbana
IllinoisWave 10Gb
Oak Ridge
Raleigh
StarLight
Chicago
Toronto
Montreal
CANARIE n*1
CANARIE-GLORIAD 1Gb
TaiwanLight 2.5Gb
Boston
MAN LAN
NGIX-East
New York
Washington DC
CAVEwave 10Gb
TeraFlow 10Gb
CiscoWave 10Gb
EnLIGHTened 10Gb
ESnet-SDN 10Gb
Internet2-DCN n*10Gb
National LambdaRail n*10Gb
Jacksonville
AtlanticWave 10Gb
AMPATH
Miami
WH
Ocean Observatories Initiative

Use Case: Command Instrument



Messaging Service



Challenges

- Can we describe protocols in OOI distributed applications accurately?
- Can we ground them to programming?
- Can we have a simple and efficient execution framework for these programs?
- Can we guarantee their safety?

Session Types

- Session types describe application-level protocols as *types*, giving static abstraction of dynamic interactions.
- Born from the structures commonly found in π-calculus encodings [Pict] of functions, objects and data types.
- Linkage to Linear Logic [Girard 87], sorting [Milner 92, Gay 93], IO sub-types [Pierce& Sangiorgi 93] and linear types [Kobayashi& Pierce & Turner 96, Y 96, Berger & H & Y 01].
- Statically and dynamically ensures type safety, deadlock freedom and progress.

Dialogue between Industry and Academia

Binary Session Types [PARLE'94, ESOP'98]

⇓

Milner, Honda and Yoshida joined W3C WS-CDL (2003)

⇓

Formalisation of W3C WS-CDL [ESOP'07]

⇓

Scribble at π4 Technology

⇓

Multiparty Session Types [POPL'08]

⇓

Beginning

From: "Robin Milner" <Robin.Milner@cl.cam.ac.uk>
Date: Wed, February 11, 2004 1:02 pm

Steve

Thanks for that. I believe that the pi-calculus team ought to be able to do something with it -- you seems to be taking it in that direction already.

Nobuko, Kohei: I thought we ought to try to model use-cases in pi-calculus, with copious explanations in natural language, aiming at seeing how various concept like role, transaction, .. would be modelled in pi. I'm hoping to try this one when I get time; you light like to try toom, and see if we agree!

Robin

Client

$Client(a) = \overline{a}(c).c(y).\overline{y}\triangleleft request\langle \tilde{v} \rangle; Q_c$

2-a

$Q_c = y \triangleright [\, ok \triangleright; \overline{y} \triangleleft reserve\langle client\_id, id \rangle;$
$\quad\quad not\_ok \triangleright .; \overline{y} \triangleleft abort \,]$

2-b ~~~ $\Longrightarrow$ (replaced by) $\overline{y} \triangleleft request2\langle \ldots \rangle$

2-c ~~~ $\Longrightarrow$ (replaced by) $Client(a)$

cf. $A\langle a \rangle = \ldots A\langle a \rangle$

Dialogue between Industry and Academia

Binary Session Types [PARLE'94, ESOP'98]

⇓

Milner, Honda and Yoshida joined W3C WS-CDL (2003)

⇓

Formalisation of W3C WS-CDL [ESOP'07]

⇓

Scribble at π4 Technology

⇓

Multiparty Session Types [POPL'08]

⇓

Dialogue between Industry and Academia

Binary Session Types [PARLE'94, ESOP'98]

⇓

Milner, Honda and Yoshida joined W3C WS-CDL (2003)

⇓

Formalisation of W3C WS-CDL [ESOP'07]

⇓

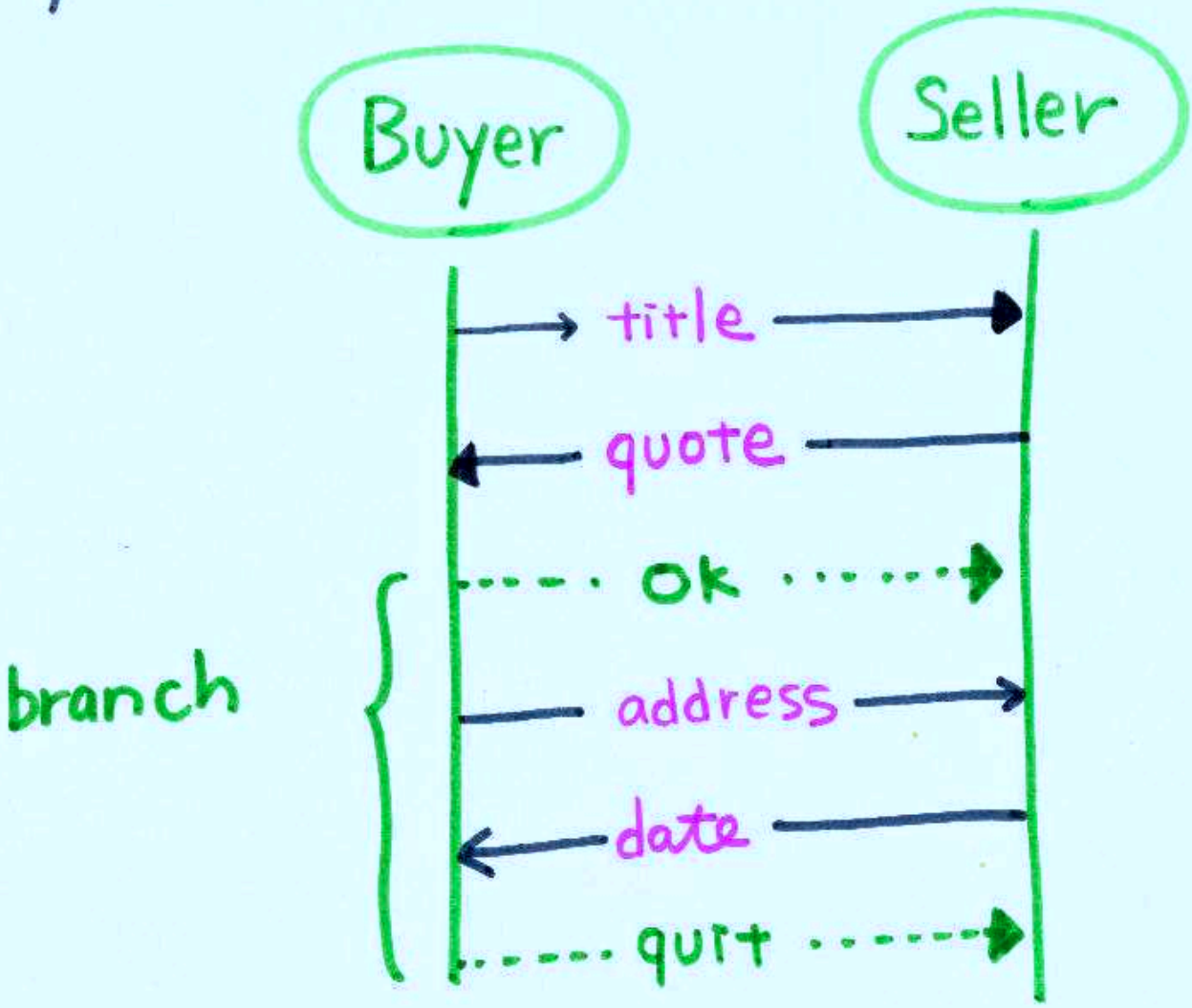
Scribble at π4 Technology

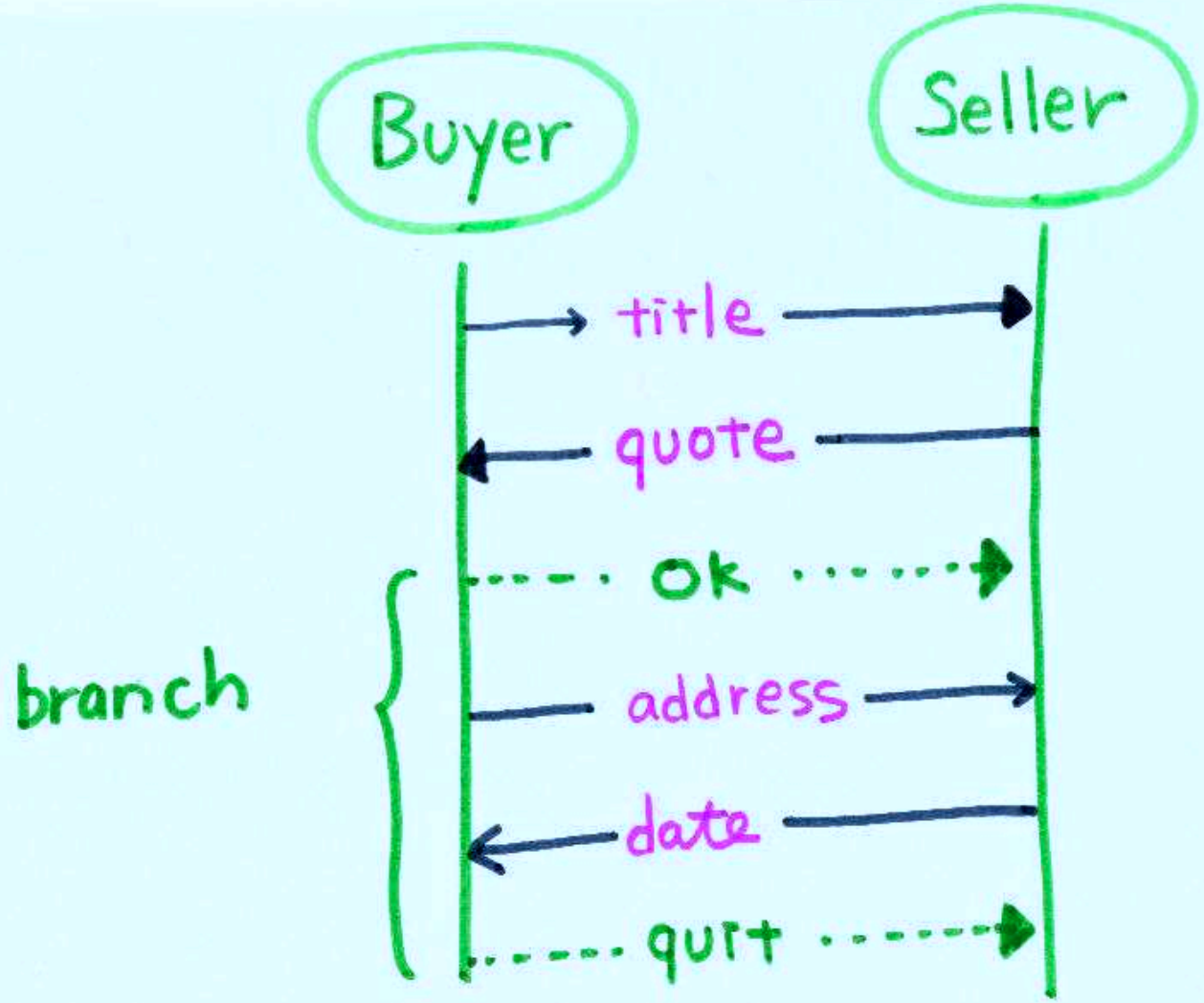
⇓

Multiparty Session Types [POPL'08]

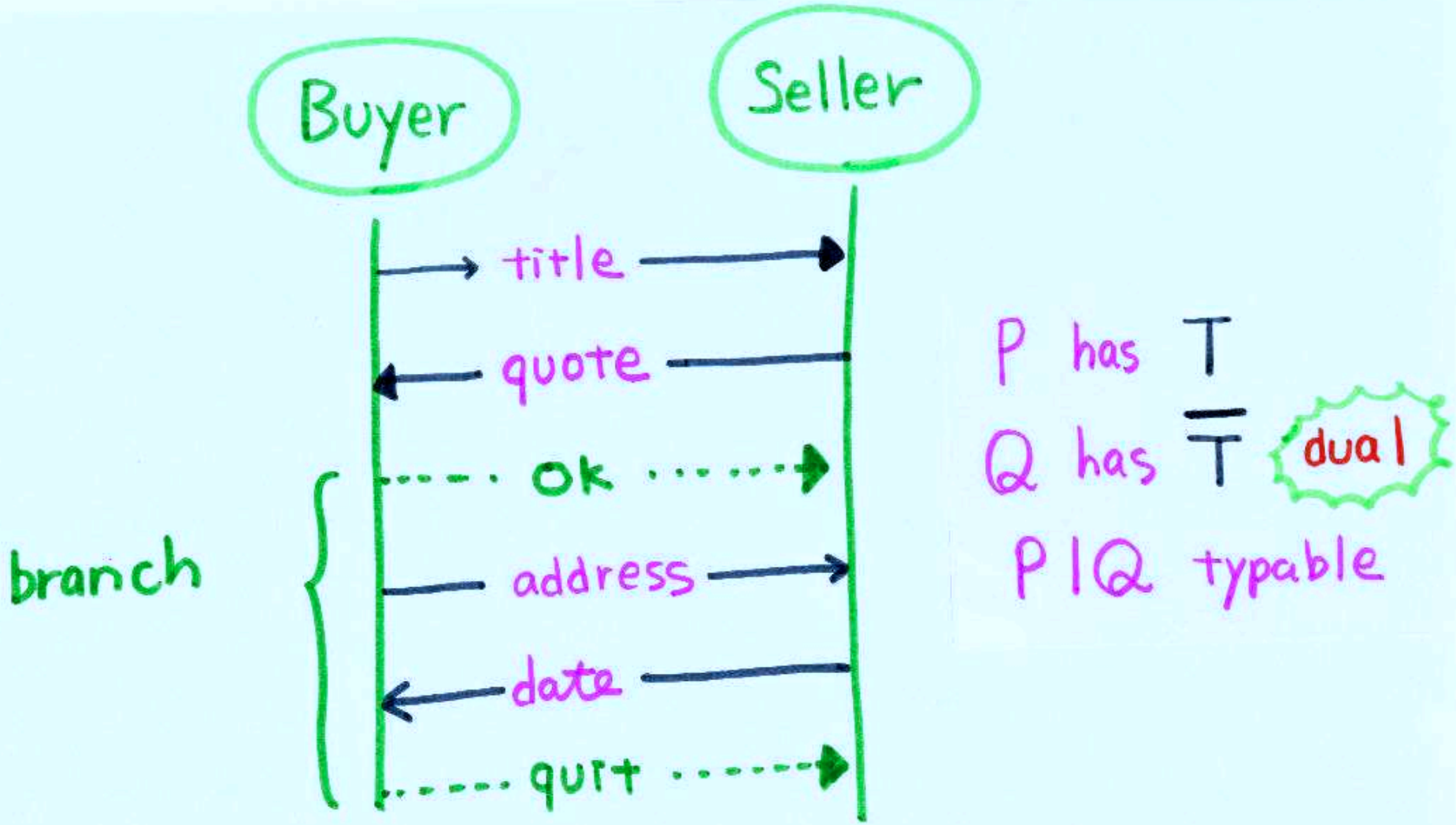
⇓

Binary Session Types : Buyer - Seller Protocol





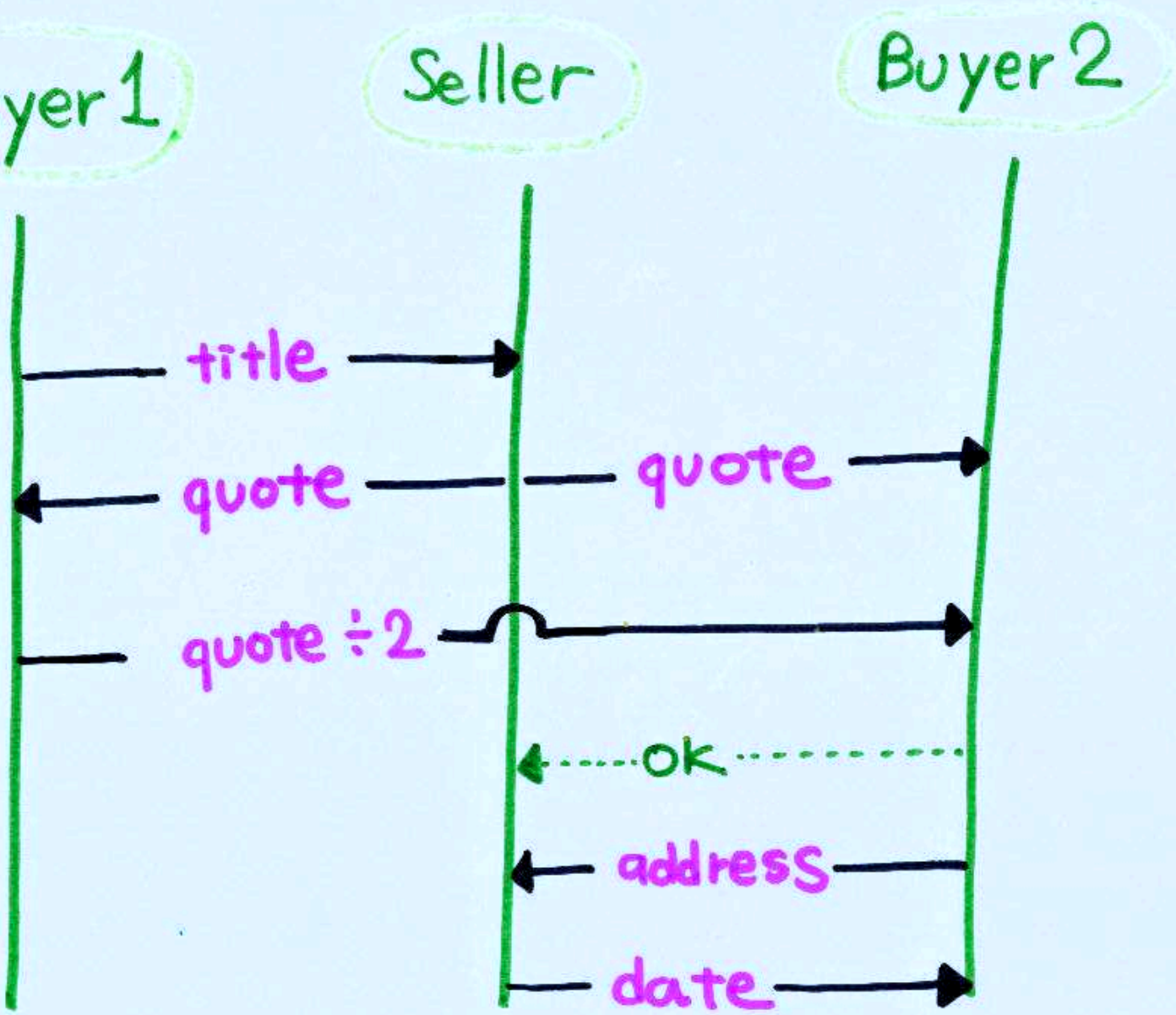
! String ; ? Int ; ⊕ { ok : ! String ; ? Date ; end , quit : end }

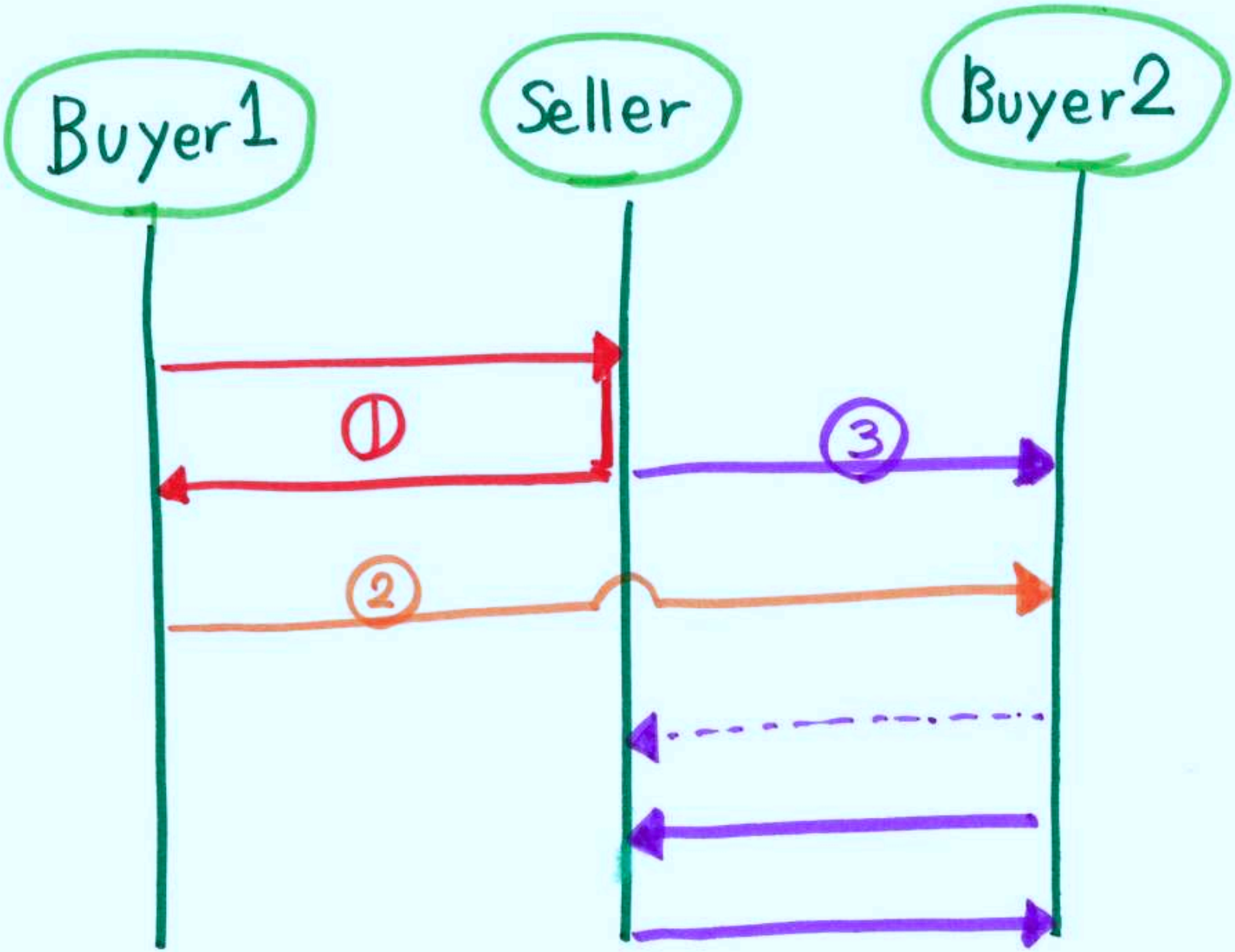


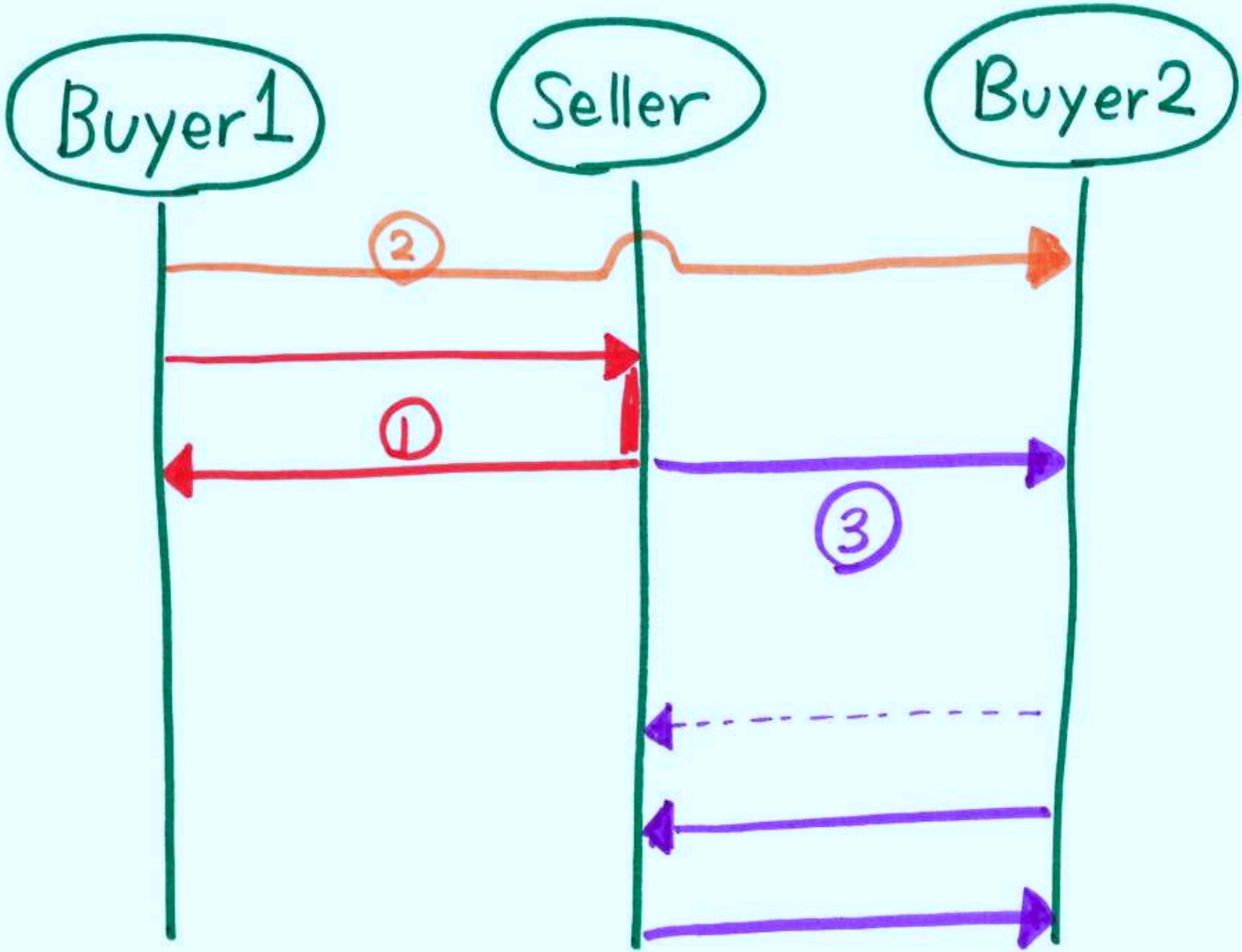
! String ; ? Int ; ⊕{ ok : ! String ; ? Date ; end , quit : end }

dual ? String ; ! Int ; & { ok : ? String ; ! Date ; end , quit : end }

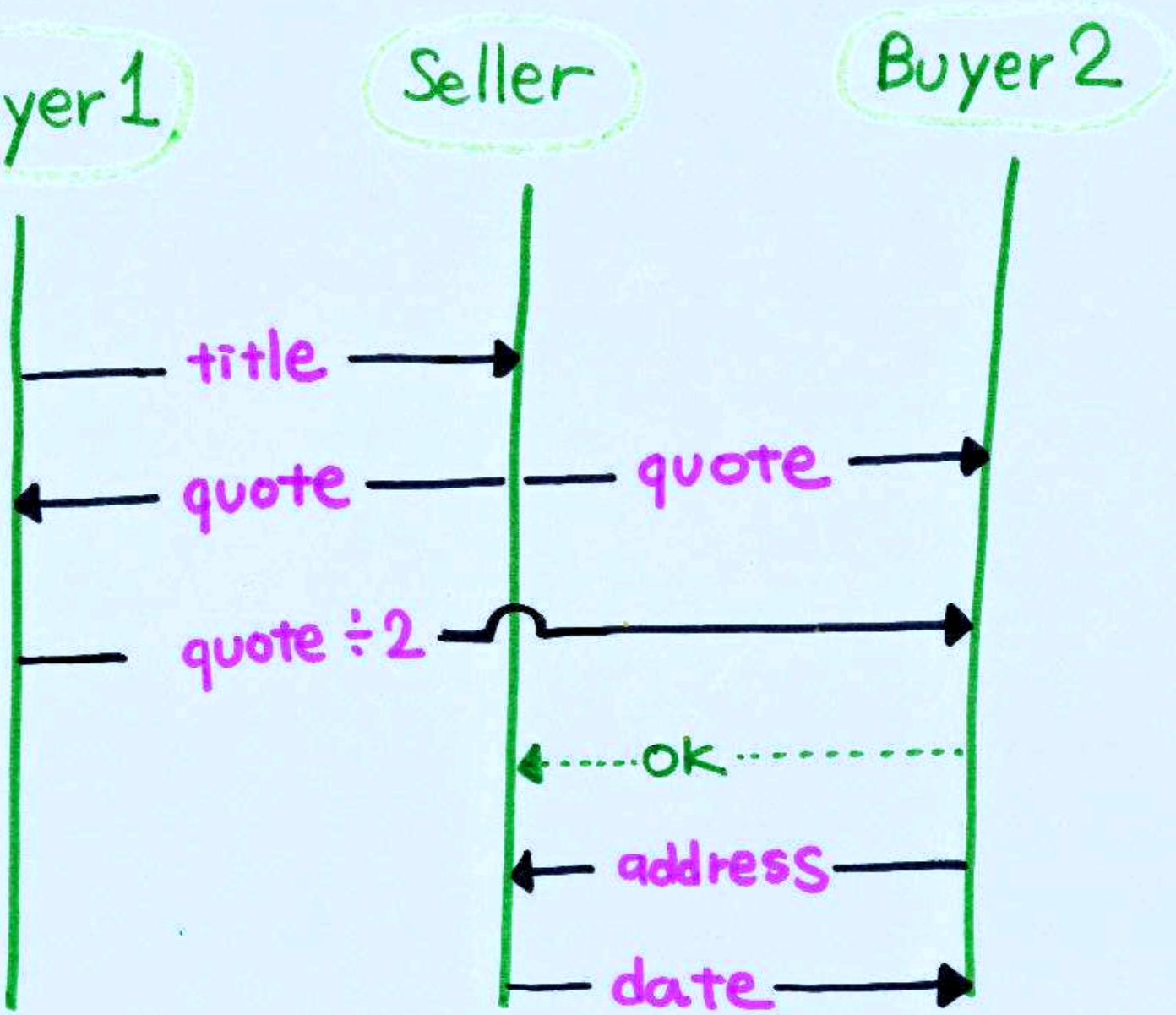
Multiparty Session Types



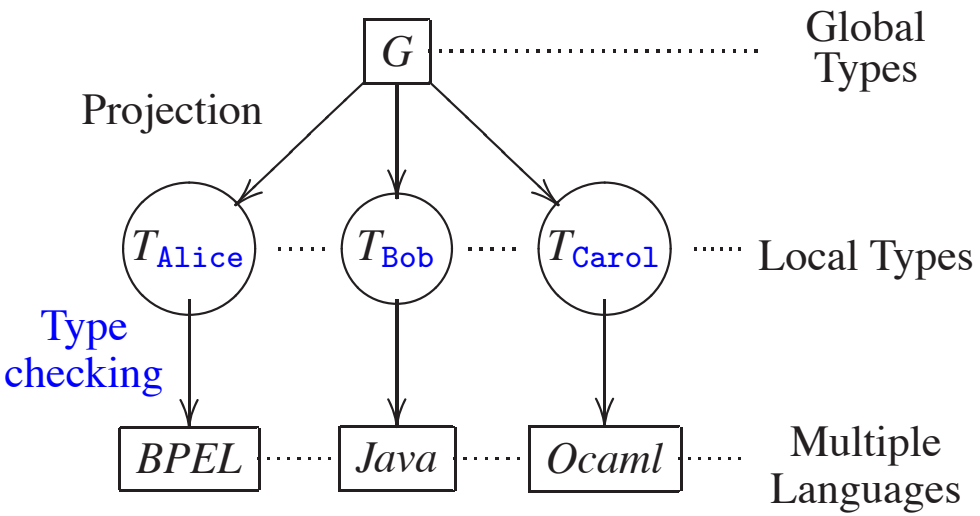

Buyer1
Seller
Buyer2
1
3
2


Buyer1
Seller
Buyer2
2
1
3

Multiparty Session Types



Multiparty Session Types

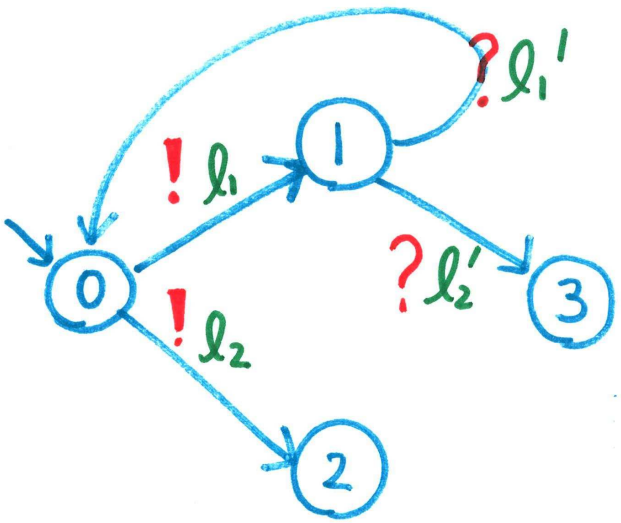


$\texttt{Alice} \to \texttt{Bob} : \langle \mathsf{Nat} \rangle .$
$\texttt{Bob} \to \texttt{Carol} : \langle \mathsf{Nat} \rangle .\mathsf{end}$

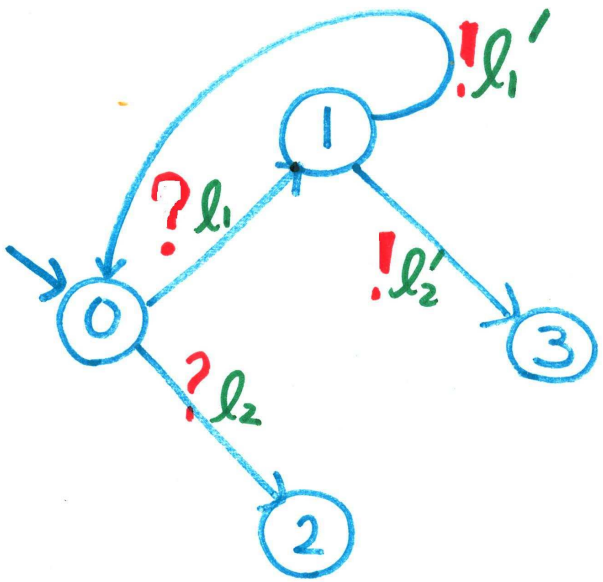
$T_{\texttt{Bob}} = ?\langle \texttt{Alice}, \mathsf{Nat} \rangle ;$
$!\langle \texttt{Carol}, \mathsf{Nat} \rangle ; \mathsf{end}$

$P_{\texttt{Bob}} = s?(\texttt{Alice}, x);$
$s!\langle \texttt{Carol}, x \rangle ; 0$

$T_1 = \mu X.\, \oplus\{ \ell_1\langle \text{Nat} \rangle.\ \&\{ \ell_1'\langle \text{Int} \rangle.\ X,\ \ell_2'\langle \text{Nat} \rangle.\ \text{End} \}$
$\ell_2\langle \text{Int} \rangle.\ \text{End} \}$



Dual

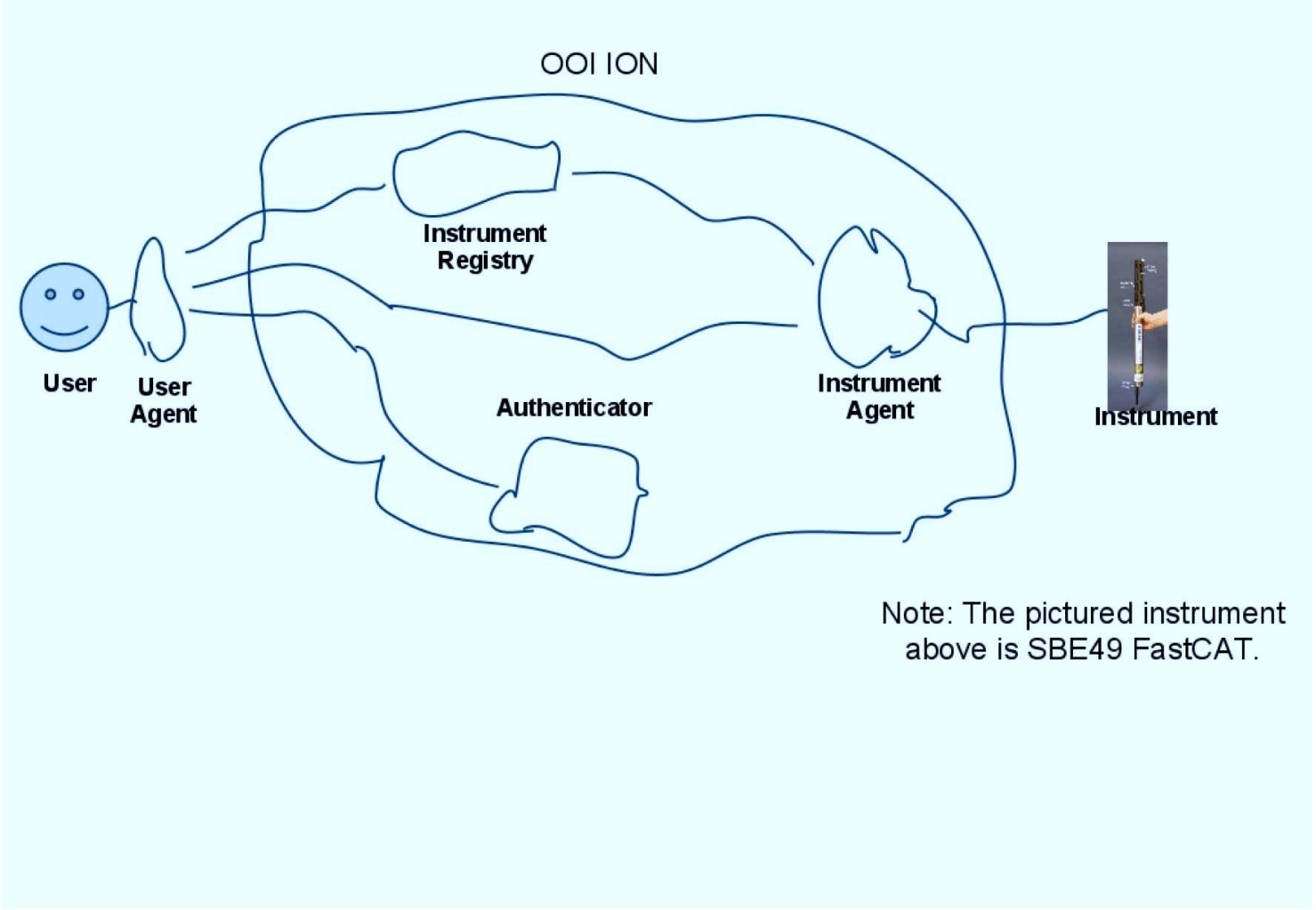


Binary session types correspond to two compatible, deterministic CFSMs with non-mixed states [Gouda et al 86] $\Longrightarrow$ Multiparty Session Automata [ESOP'12].

Applying theories

- We are contributing to OOI through a protocol description language, *Scribble*, and development/execution environments, all based on the underlying theories *without* compromise.
- Development/execution environments centre on a tool chain for protocol validation, endpoint projection, FSM translations, APIs and runtimes, all directly informed by the theory, integrated with the OOI CI environments.
- Developers can develop using “legacy” communication primitives, such as RPC, which are executed as sessions.

Use Case: Command Instrument



Global type

```
protocol CommandInstrument (role user, role registry,
                            role agent, role instrument) {
   request(string instrumentId, int priority) from user to registry;
   maxCommands(int max) from registry to user;
   request(string instrumentId) from user to agent;
   choice at agent {
      accept() from agent to user;
      rec loop {
         choice at user {
            request(string command) from user to agent;
            request(string command) from Agent to Instrument;
            response(DataFormat data) from instrument to agent;
            response(DataFormat data) from agent to user;
            loop;
         } or {
            quit()  from user to agent;
         }
   } or {
      reject(String reason) from agent to user;
   }
}
```

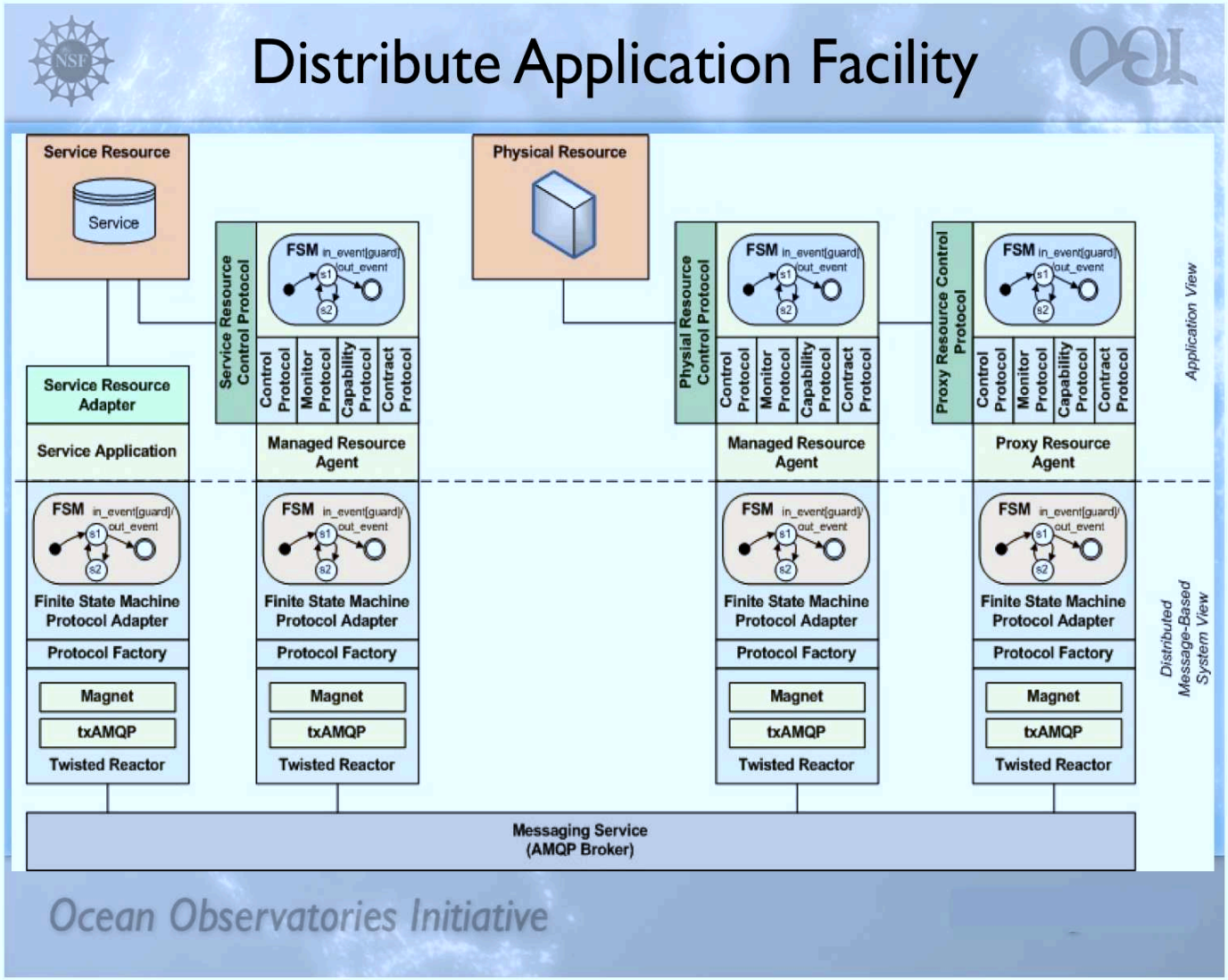
Local type

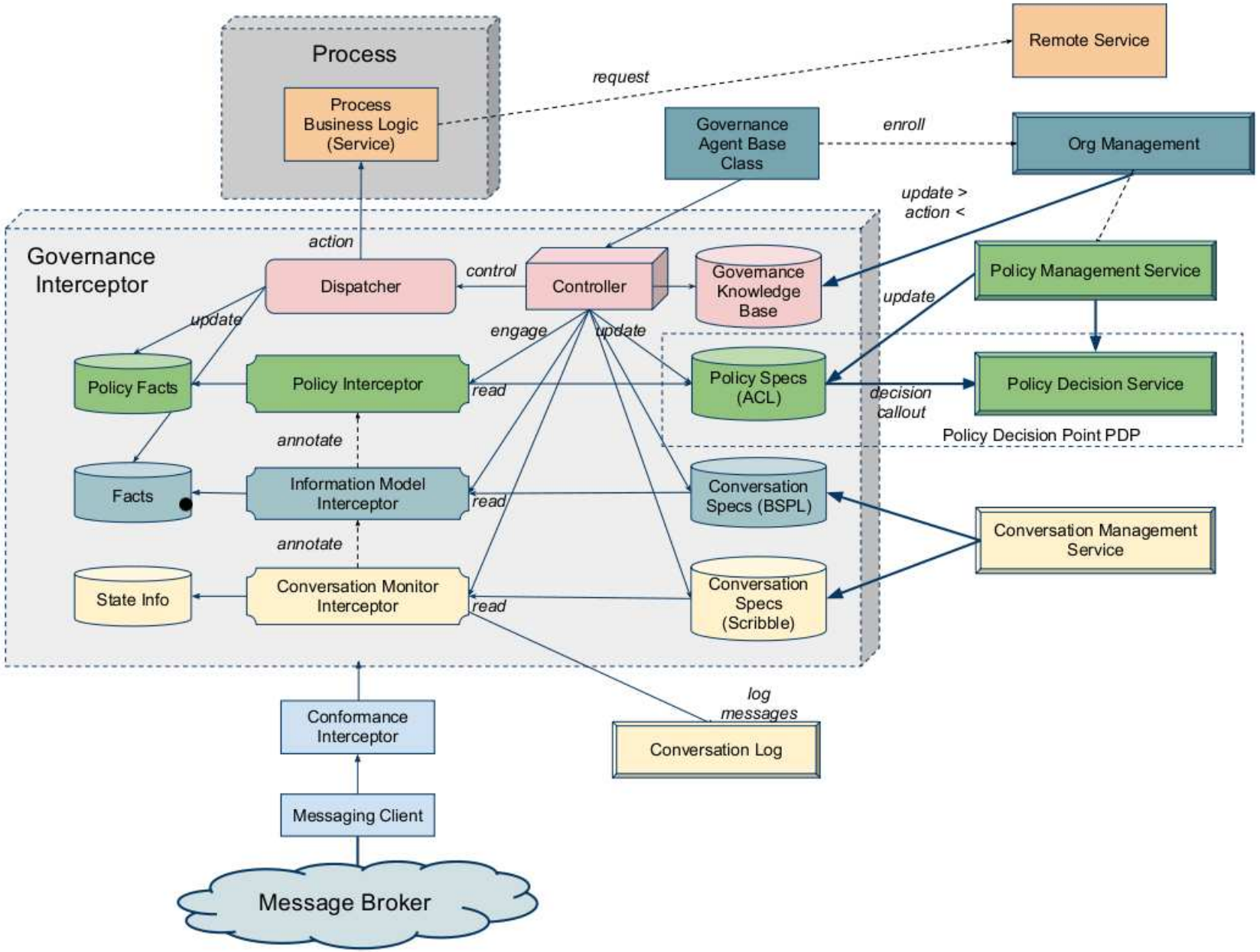
```
protocol CommandInstrument (role client, role registry,
                            role agent) at client {
   request(string instrumentId) to registry;
   maxCommands(int max) from registry;
   request(string instrumentId) to agent;
   choice at agent {
      accept() from agent;
      rec loop {
         choice at user {
            request(string command) to agent;
            response(DataFormat data) from agent;
            loop;
         } or {
            quit() to agent;
         }
   } or {
      reject(String reason) to agebt;
   }
}
```

Program

```
class ClientApp():
    def command_instrument(self, registry, agent, command):
        c = Conversation.create('CommandInstrument', 'client')
        c.request(registry, 'registry')
        c.send('registry', 'request', 'SBE49FastCAT')
        conv_msg = c.receive('registry')
        max = conv_msg.payload
        c.request(agent, 'agent')
        conv_msg = c.receive('agent')
        if (conv_msg.operation == 'accept'):
            count = 0
            while(count < max):
                c.send('agent', 'request', command)
                conv_msg = c.receive('agent')
                data = conv_msg.payload
                # and output data..
                count = count+1;
        elif (conv_msg.operation == 'reject'):
             ....
```

Distribute Application Facility




Process
Process Business Logic (Service)
request
Remote Service
Governance Agent Base Class
enroll
Org Management
update >
action <
Governance Interceptor
action
Dispatcher
control
Controller
Governance Knowledge Base
Policy Management Service
update
update
engage
update
Policy Facts
Policy Interceptor
read
Policy Specs (ACL)
decision
callout
Policy Decision Service
Policy Decision Point PDP
annotate
Facts
Information Model Interceptor
read
Conversation Specs (BSPL)
Conversation Management Service
annotate
State Info
Conversation Monitor Interceptor
read
Conversation Specs (Scribble)
log
messages
Conformance Interceptor
Conversation Log
Messaging Client
Message Broker

Ocean Observatory Initiative

CREATE ACCOUNT | SIGN IN

SEARCH

RESOURCES

- All Resources
- Data Products ✓
- Observatories
- Platforms
- Instruments

Welcome to Release 2 of the Ocean Observatories Initiative Observatory (OOI). You already have access to many OOI features and real-time data. Just click on something that looks interesting on this page to start using the OOI as our Guest.

For personalized services, such as setting up notifications and preserving settings for your next visit, create a free account by clicking on "Create Account" at the top of the page.

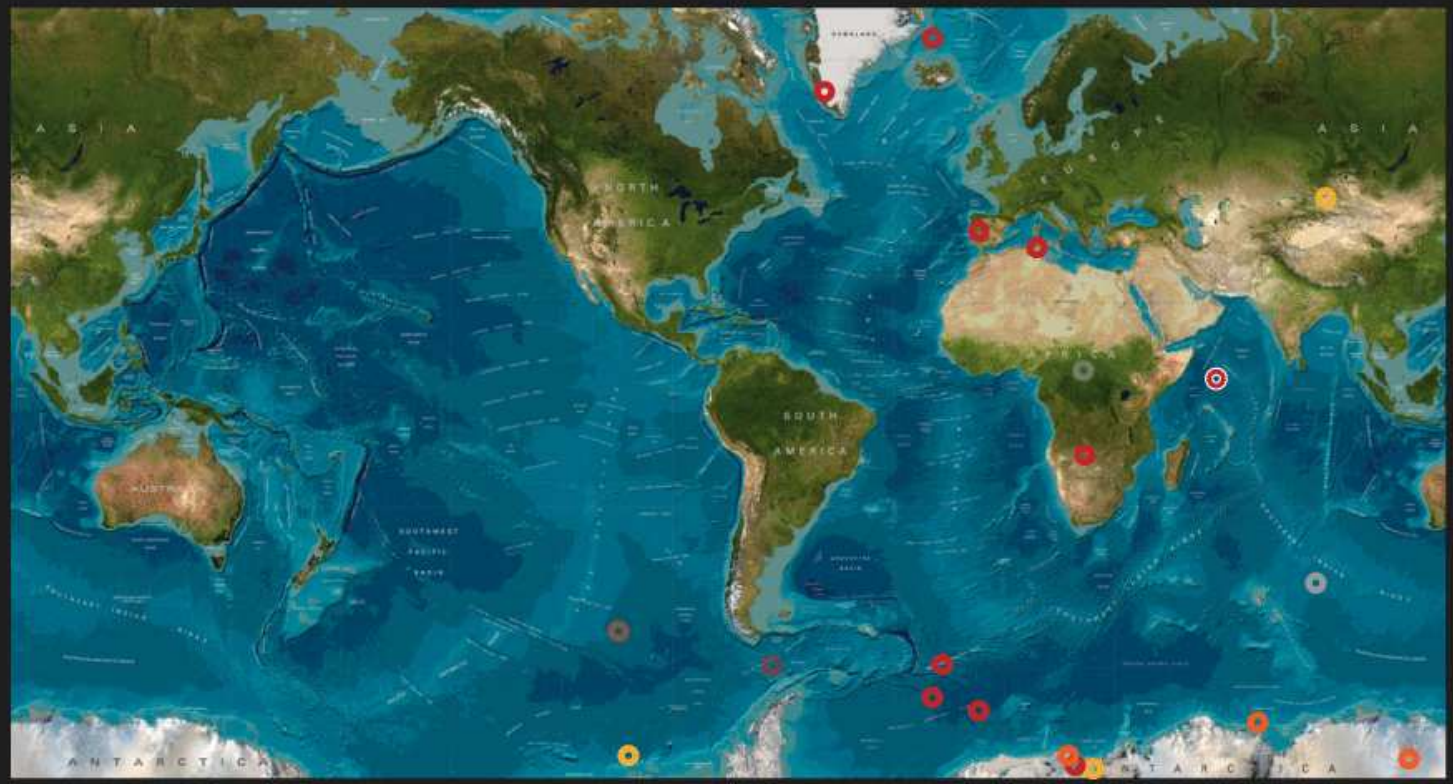
National Science Foundation working with *Consortium for Ocean Leadership*

Funding for the Ocean Observatories Initiative is provided by the National Science Foundation through a Cooperative Agreement with the Consortium for Ocean Leadership. The OOI Program Implementing Organizations are funded through sub-awards from the Consortium for Ocean Leadership.

Location

CURRENT LOCATION

FILTER



DATA LEGEND

- Temperature ✓
- Salinity
- Oxygen ✓
- Density
- Currents
- Sea Surface Height (SSH)
- Chlorophyll
- Turbidity ✓
- pH
- Seismology ✓
- Other ✓

RECENCY

1 Hour
2 hours
3 hours
5 hours
8 hours
12 hours
18 hours
24 hours
48 Hours
72 Hours

RECENT UPDATES

	NAME	DATE	TYPE	EVENT	DESCRIPTION	NOTE
01 m	Oregon Coast North Salinity	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 m	California South 100m pH	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 m	California South salinity	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
03 m	Oregon North Turbidity	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
05 m	Oregon SouthTemperature	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
20 m	Oregon Coast Currents	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 h	California South Seismology	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 h	Oregon Coast South 1000m Ox	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
02 h	California Coast Seismology	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
04 h	California North Seismology	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here

FACEPAGE | RELATED | COMPOSITE | STATUS

Dashboard

RECENT IMAGES

Glider
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

Gorgonian Coral
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

Acoustic Release
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

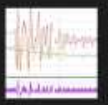
POPULAR RESOURCES

SeaBird CDT
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

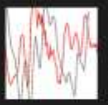
Marine caption
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

Surface Buoy
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

UNUSUAL EVENTS



Oregon Coast Wave Heigh
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24



Water Surface Elevation
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

Programming with Conversations and Protocols

- Does it help programming? Does it offer developers a good programming abstraction?
- Does it help verification? Does it offer a good basis for specifications and verifications?
- Does it help us run programs efficiently with a simple runtime machinery?

Ongoing work

- Enrichment of session types, often stimulated by practice (e.g. dependent types).
- Behavioural equivalences (e.g. when two global services are equivalent?).
- Specification and verification (e.g. assertions, refinement).
- Development and execution frameworks informed by theories (e.g. Scribble).
- Different flavours of MPST-based programming (e.g. OCaml, Java, Scala, Haskell, Python, C, ...).

Conclusion

"Types are the leaven for computer programming; they make it digestible." [Milner 02]

- Communication and concurrency are becoming the norm in computing.
- To harness their power, we need to understand their nature, and identify effective principles to build, specify and verify them: concurrency theories will help as core scientific principles.
- Diverse theories of types for processes can help, thanks to their catalytic power.

Notes

- Robin's mail in page 14 is reproduced as it was sent to us. In the last sentence of the mail, he clearly meant to say: "you might like to try too".
- The figure in page 7 has been tracked to *Dictionary of Computer and Internet Terms* (Barron's Educational Series). These programs, in Pascal, use a sign (plus or minus) to indicate whether x is greater than y.