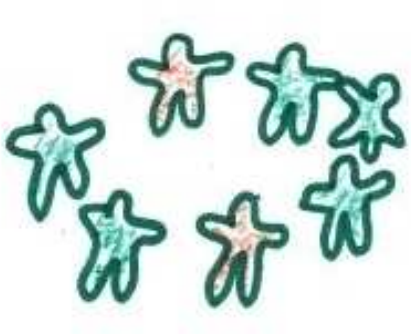
Scribble, Runtime Verification and Multiparty Session Types



http://mrg.doc.ic.ac.uk/

Nobuko Yoshida

Imperial College London

In collaboration with:

Matthew Arrott (OOI)

Gary Brown (Red Hat)

Stephen Henrie (OOI)

Bippin Makoond (Cognizant/Qualit-e)

Michael Meisinger (OOI)

Matthew Rawlings (ISO TC68 WG4/5)

Alexis Richardson (RabbitMQ/Pivotal)

Steve Ross-Talbot (Cognizant/Qualit-e)

and all our academic colleagues

Laura Bocchi, Tzu-Chun Chen, Tiago Cogumbreiro, Romain Demangeon, Pierre-Malo Denielóu, Juliana Franco, Luca Fossati, Dimitrios Kouzapas, Julien Lange, Rumyana Neykova, Nicholas Ng, Weizhen Yang

The Kohei Honda Prize for Distributed Systems Queen Mary, University of London

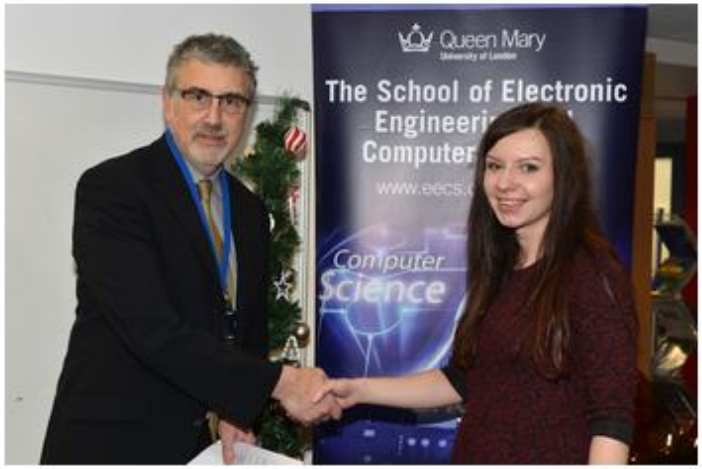
Posted with permission from QMUL on 17th Dec 2013. Original article written by Edmund Robinson.

This prize was instituted in 2013 and is awarded annually to one undergraduate student and one postgraduate student in recognition of their achievement in applying the highest quality scientific and engineering principles in the broad area of Distributed Systems. This is the area in which Dr Honda concentrated most of his teaching, and it is also the area in which he conducted his research. Its primary funding comes from a donation from his family, who wished to commemorate Dr Honda in this way. Additional funding has come from Dr Honda's own ETAPS Award. This prize is sponsored by Springer Verlag, and awarded annually by the ETAPS committee in recognition of an individual's research contribution. Dr Honda received the first such award posthumously, and the awarding panel expressed a wish that the funding be used to supplement this prize fund. The laudation for this award, written by Dr Honda's colleague, Prof Vladimiro Sassone is included later.

About Dr Honda

Kohei Honda was born and lived the first part of his life in Japan. Like many scientists he was fascinated by the idea of finding basic explanatory theories, like the physicists looking for grand unified theories of the universe. Kohei, though, was passionately interested in finding the right basic explanatory theory for the process of computation. Most academics agree that the basic theory

Winners 2013



Ms Anna Pawlicka
2013 winner (Undergraduate) source: QMUL

Mr. Valdmir Negacevshi
2013 winner (Postgraduate) source: QMUL

Outline

- Background
- Multiparty Session Types
- Scribble and Applications to a Large-scale Cyberinfrastructure
- Monitoring Theory
- Summary

Communication is Ubiquitous

- Internet, the WWW, Cloud Computing, the next-generation manycore chips, message-passing parallel computations, large-scale cyberinfrastructure for e-Science.
- The way to organise software is increasingly based on communications.
- Applications need *structured* series of communications.

- Question
  - How to formally abstract/specify/implement/control communications?

Communication is Ubiquitous

- Internet, the WWW, Cloud Computing, the next-generation manycore chips, message-passing parallel computations, large-scale cyberinfrastructure for e-Science.
- The way to organise software is increasingly based on communications.
- Applications need *structured* series of communications.

- Question
  - How to formally abstract/specify/implement/control communications?

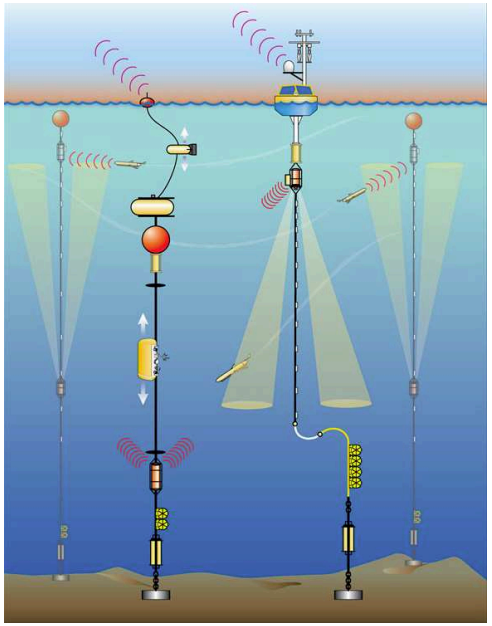
Communication is Ubiquitous

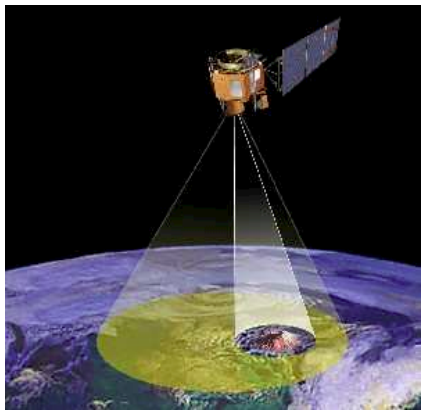
- Internet, the WWW, Cloud Computing, the next-generation manycore chips, message-passing parallel computations, large-scale cyberinfrastructure for e-Science.
- The way to organise software is increasingly based on communications.
- Applications need *structured* series of communications.

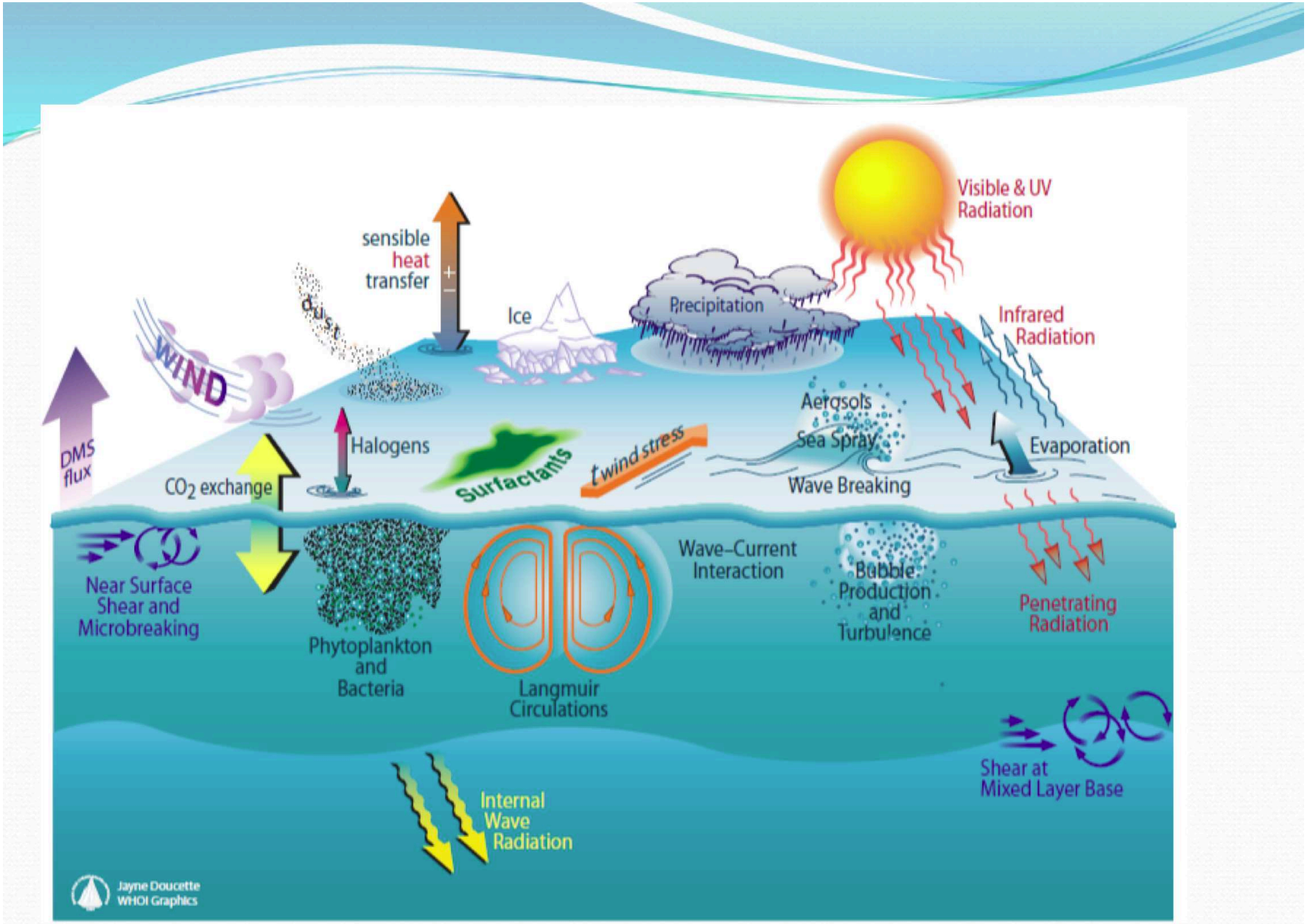
- Question $\Longrightarrow$ Multiparty session type theory
  - How to formally abstract/specify/implement/control communications?

Ocean Observatories Initiative

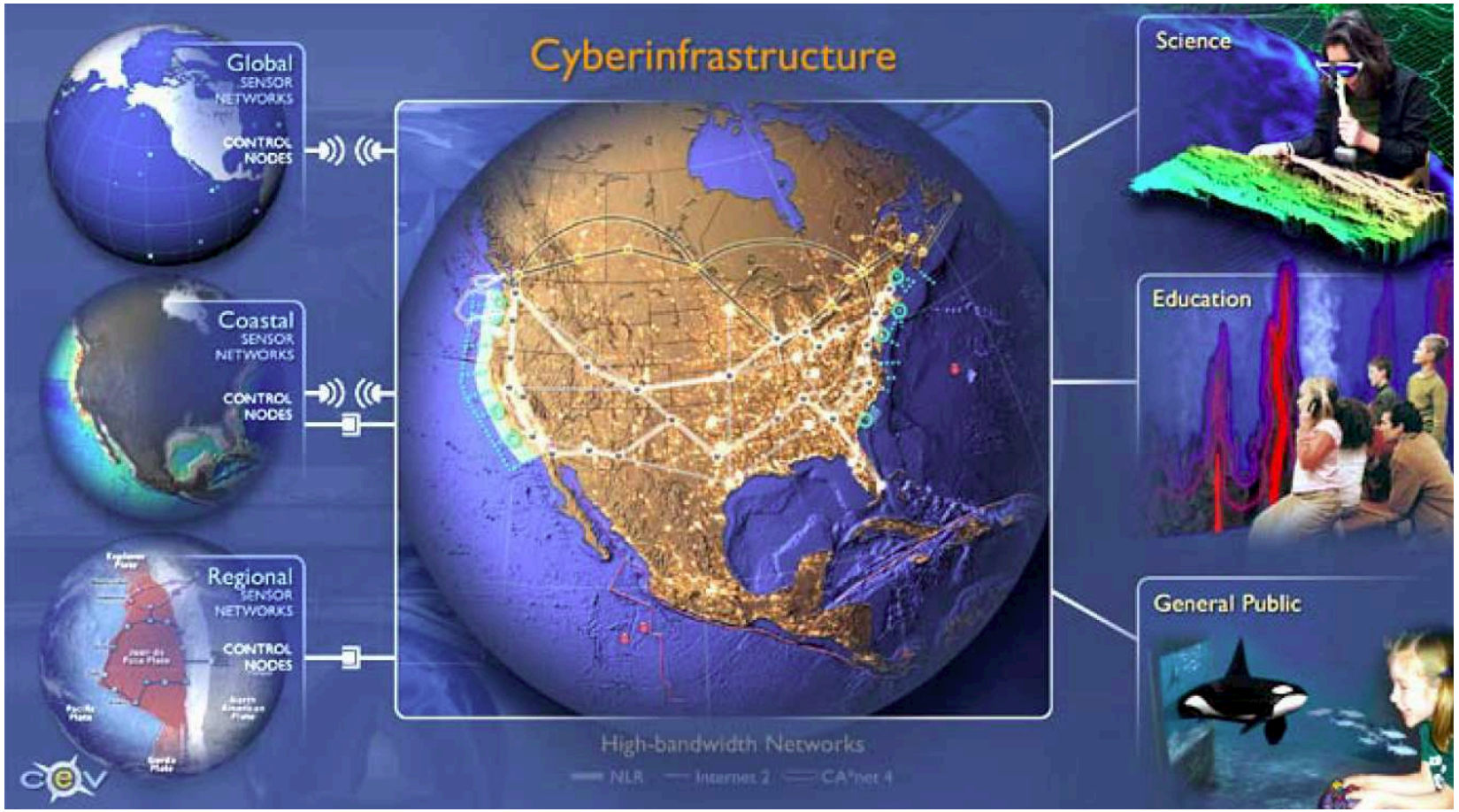
- A NSF project (400M$, 5 Years) to build a cyberinfrastructure for observing oceans around US and beyond.
- Real-time sensor data constantly coming from both off-shore and on-shore (e.g. buoys, submarines, under-water cameras, satellites), transmitted via high-speed networks.

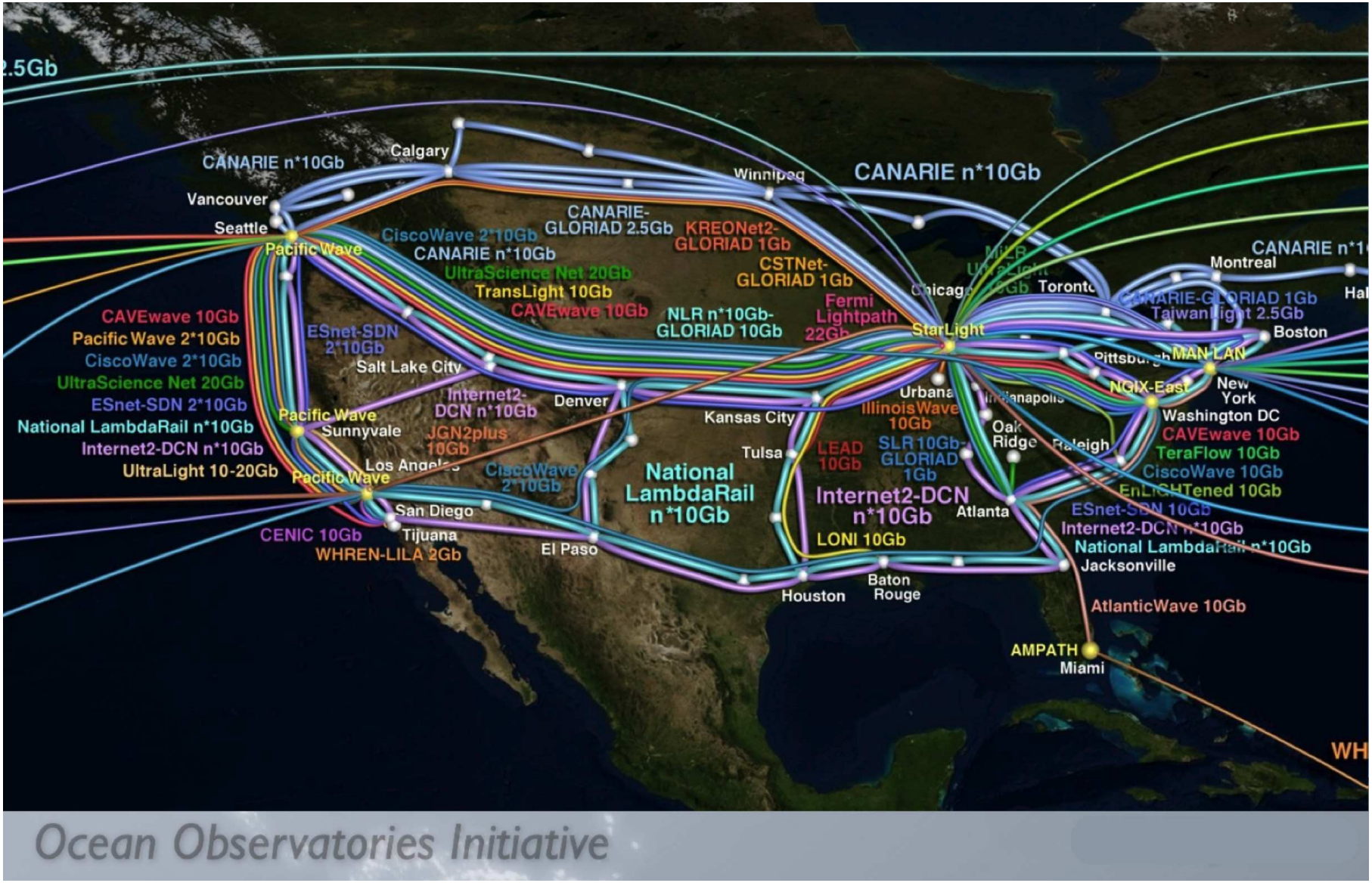





Visible & UV Radiation
sensible heat transfer
+
−
Ice
Precipitation
Infrared Radiation
dust
WIND
DMS flux
Halogens
Aerosols
Sea Spray
Evaporation
CO_2 exchange
Surfactants
t wind stress
Wave Breaking
Near Surface Shear and Microbreaking
Phytoplankton and Bacteria
Wave–Current Interaction
Bubble Production and Turbulence
Penetrating Radiation
Langmuir Circulations
Shear at Mixed Layer Base
Internal Wave Radiation
Jayne Doucette
WHOI Graphics

Ocean Observatories Initiative





Ocean Observatories Initiative

Challenges

- The need to specify, catalogue, program, implement and manage *multiparty message passing protocols*.
- Communication assurance
  - Correct message ordering and synchronisation
  - Deadlock-freedom, progress and liveness
  - Dynamic message monitoring and recovery
  - Logical constraints on message values
- Shared and used over a long-term period (e.g. 30 years in OOI).

Why Multiparty Session Types?

- Robin Milner (2002): *Types are the leaven of computer programming; they make it digestible*.

  $\Longrightarrow$ Can describe communication protocols as *types*

  $\Longrightarrow$ Can be materialised as *new communications programming languages* and *tool chains*.

- *Scalable* automatic verifications (deadlock-freedom, safety and liveness) without *state-space explosion problems* (*polynomial time complexity*).

- Extendable to *logical verifications* and flexible *dynamic monitoring*.

Dialogue between Industry and Academia

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

$\Downarrow$

Milner, Honda and Yoshida joined W3C WS-CDL (2002)

$\Downarrow$

Formalisation of W3C WS-CDL [ESOP'07]

$\Downarrow$

Scribble at $\pi 4$ Technology

CDL Equivalent

- Basic example:

```
package HelloWorld {
    roleType YouRole, WorldRole;
    participantType You{YouRole}, World{WorldRole};
    relationshipType YouWorldRel between YouRole and WorldRole;
    channelType WorldChannelType with roleType WorldRole;

    choreography Main {
        WorldChannelType worldChannel;

        interaction operation=hello from=YouRole to=WorldRole
                relationship=YouWorldRel channel=worldChannel {
            request messageType=Hello;
        }
    }
}
```

Scribble Protocol

- *"Scribbling is necessary for architects, either physical or computing, since all great ideas of architectural construction come from that unconscious moment, when you do not realise what it is, when there is no concrete shape, only a whisper which is not a whisper, an image which is not an image, somehow it starts to urge you in your mind, in so small a voice but how persistent it is, at that point you start scribbling" - Kohei Honda 2007*

- Basic example:

```
protocol HelloWorld {
    role You, World;
    Hello from You to World;
}
```

Dialogue between Industry and Academia

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

⇓

Milner, Honda and Yoshida joined W3C WS-CDL (2002)

⇓

Formalisation of W3C WS-CDL [ESOP'07]

⇓

Scribble at $\pi 4$ Technology

⇓

Multiparty Session Types [POPL'08]

⇓

Dialogue between Industry and Academia

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

⇓

Milner, Honda and Yoshida joined W3C WS-CDL (2002)

⇓

Formalisation of W3C WS-CDL [ESOP'07]

⇓

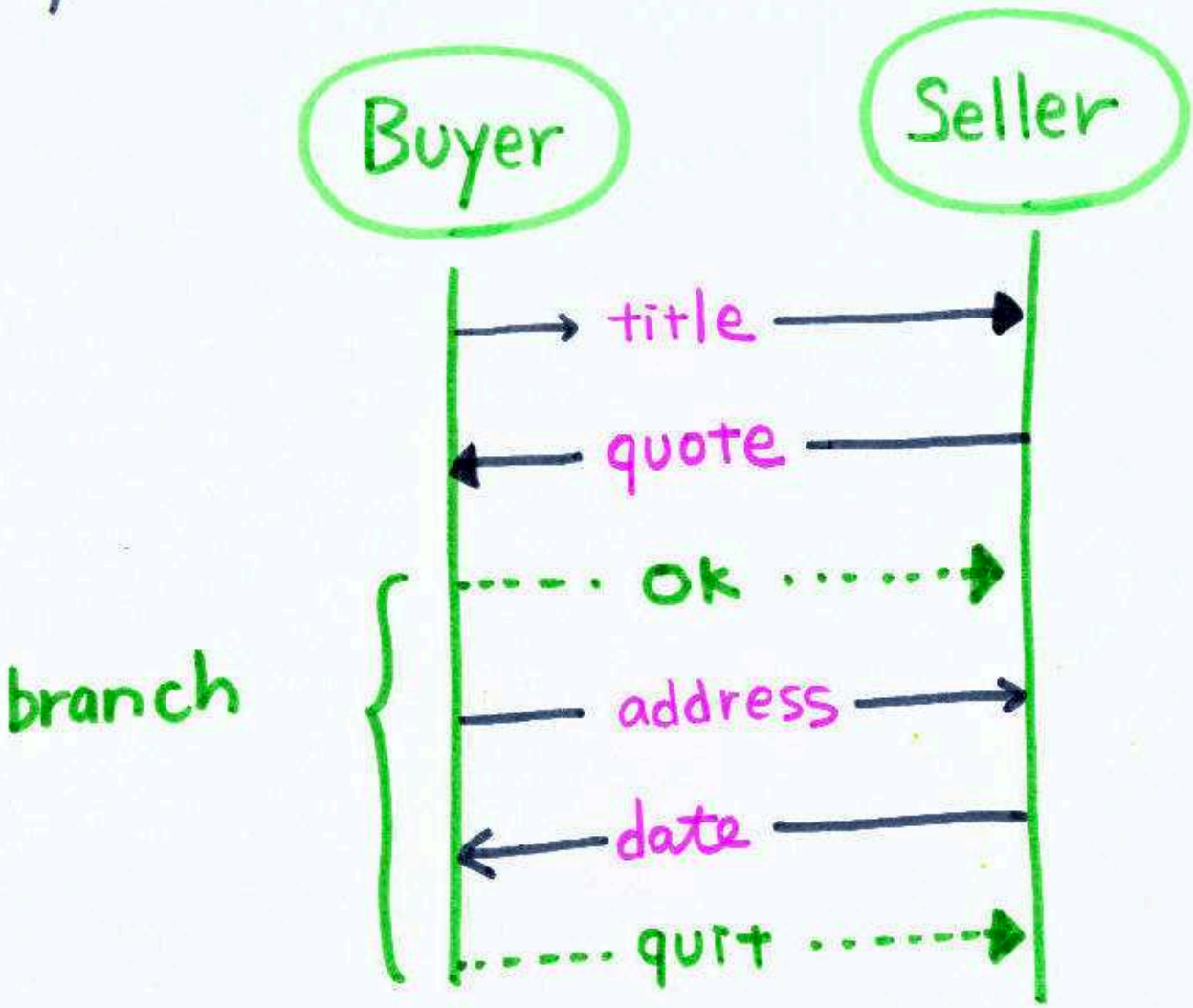
Scribble at $\pi 4$ Technology

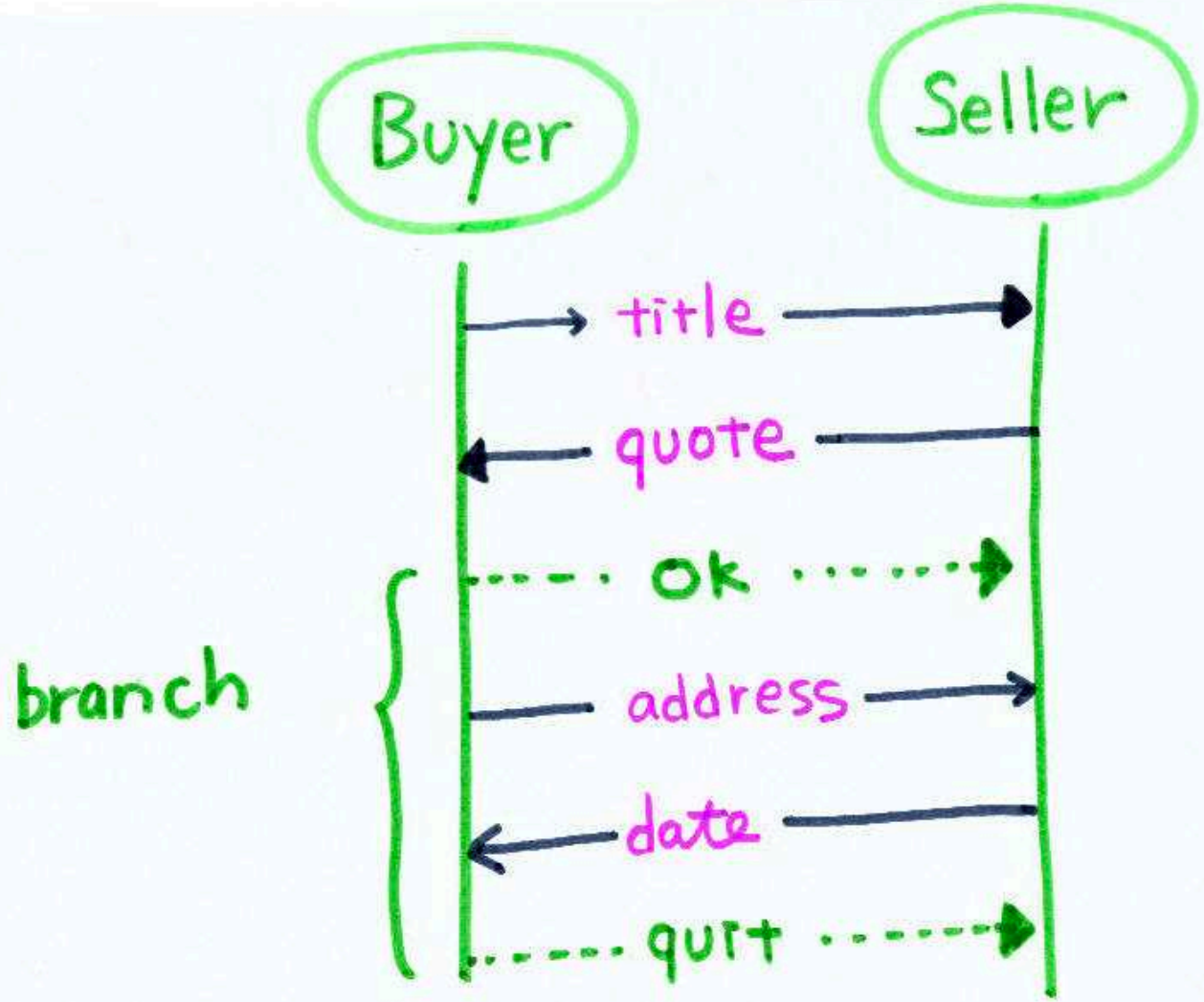
⇓

Multiparty Session Types [POPL'08]

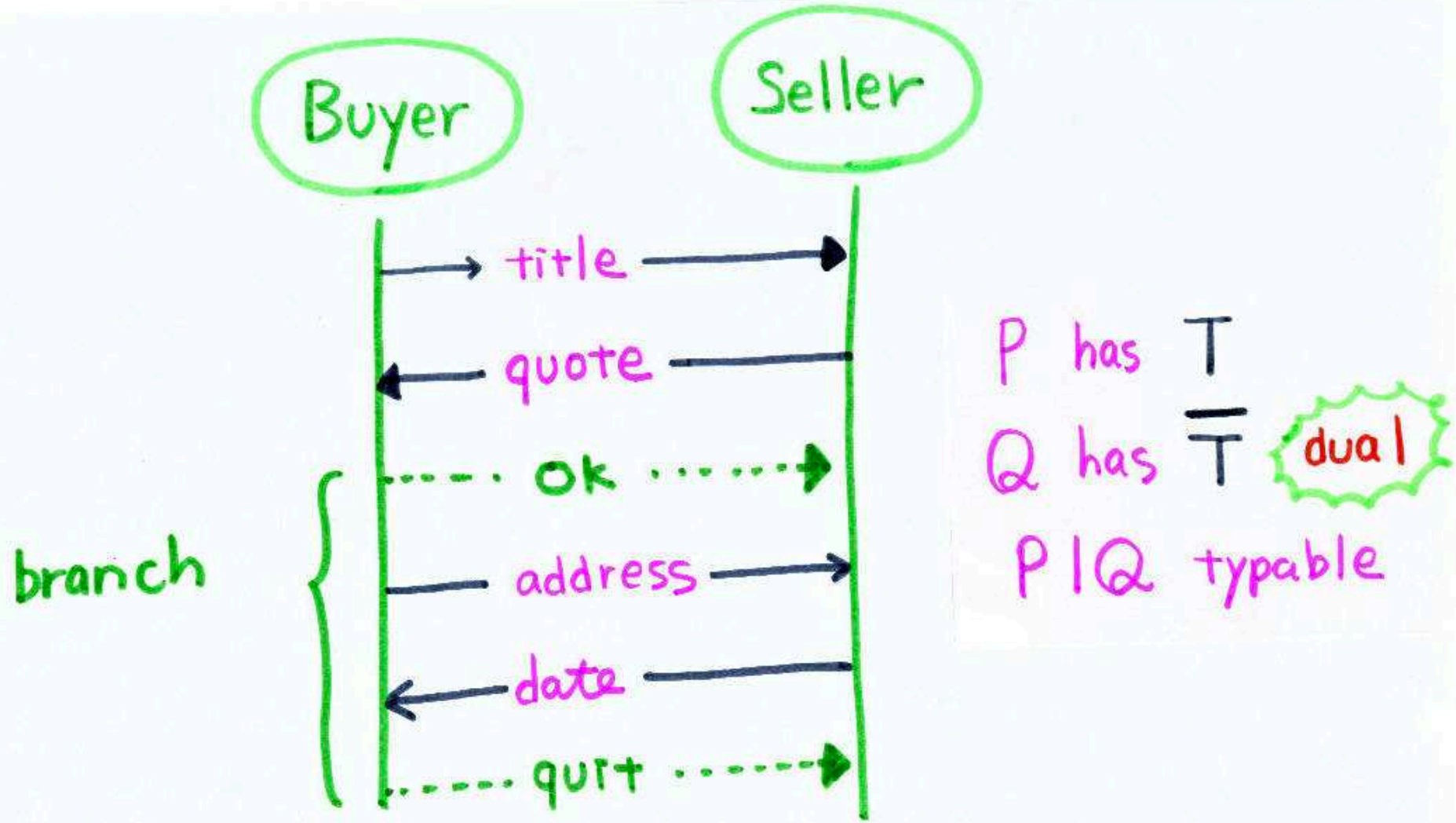
⇓

Binary Session Types : Buyer - Seller Protocol





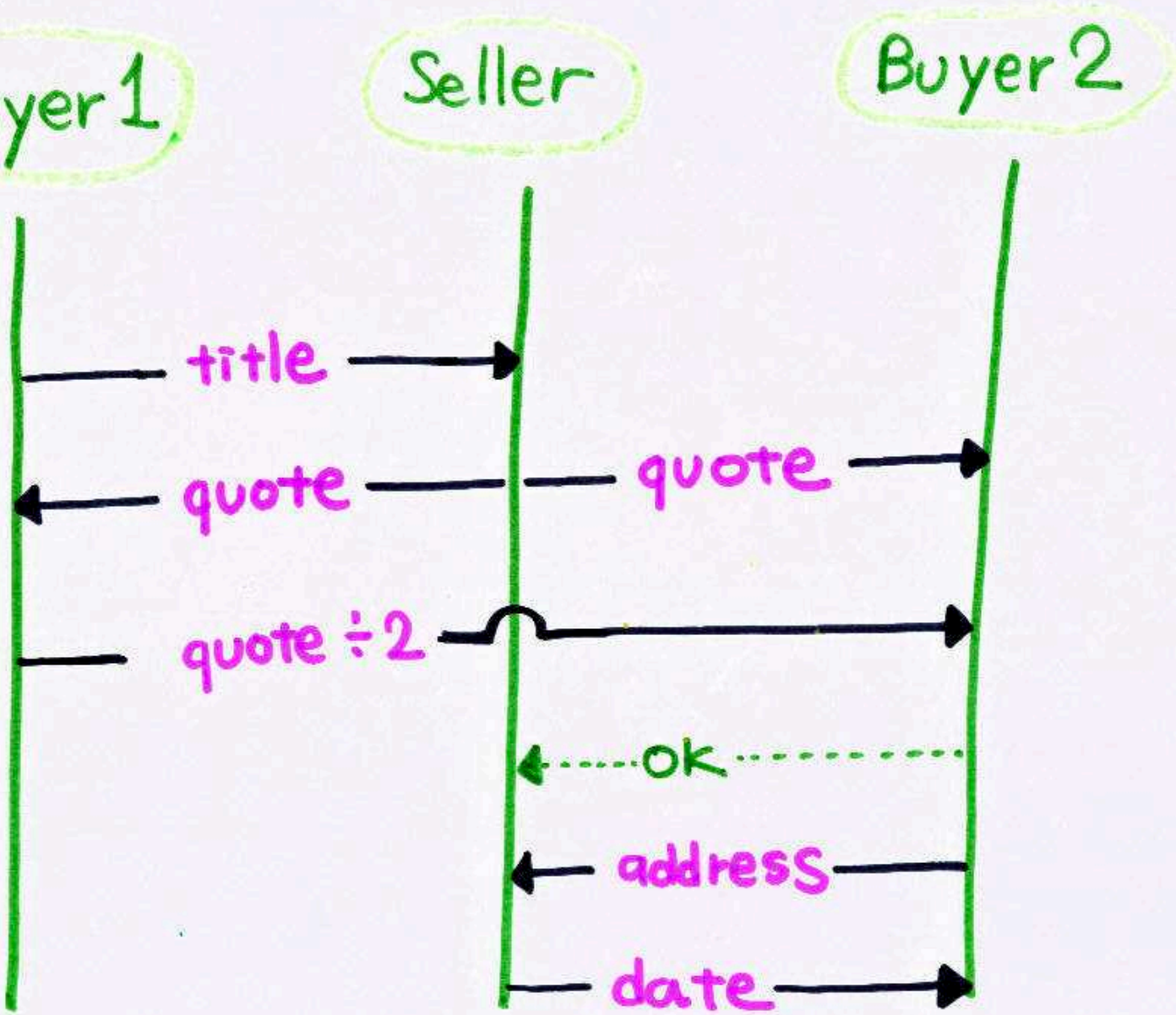
! String ; ? Int ; ⊕ { ok : ! String ; ? Date ; end , quit : end }

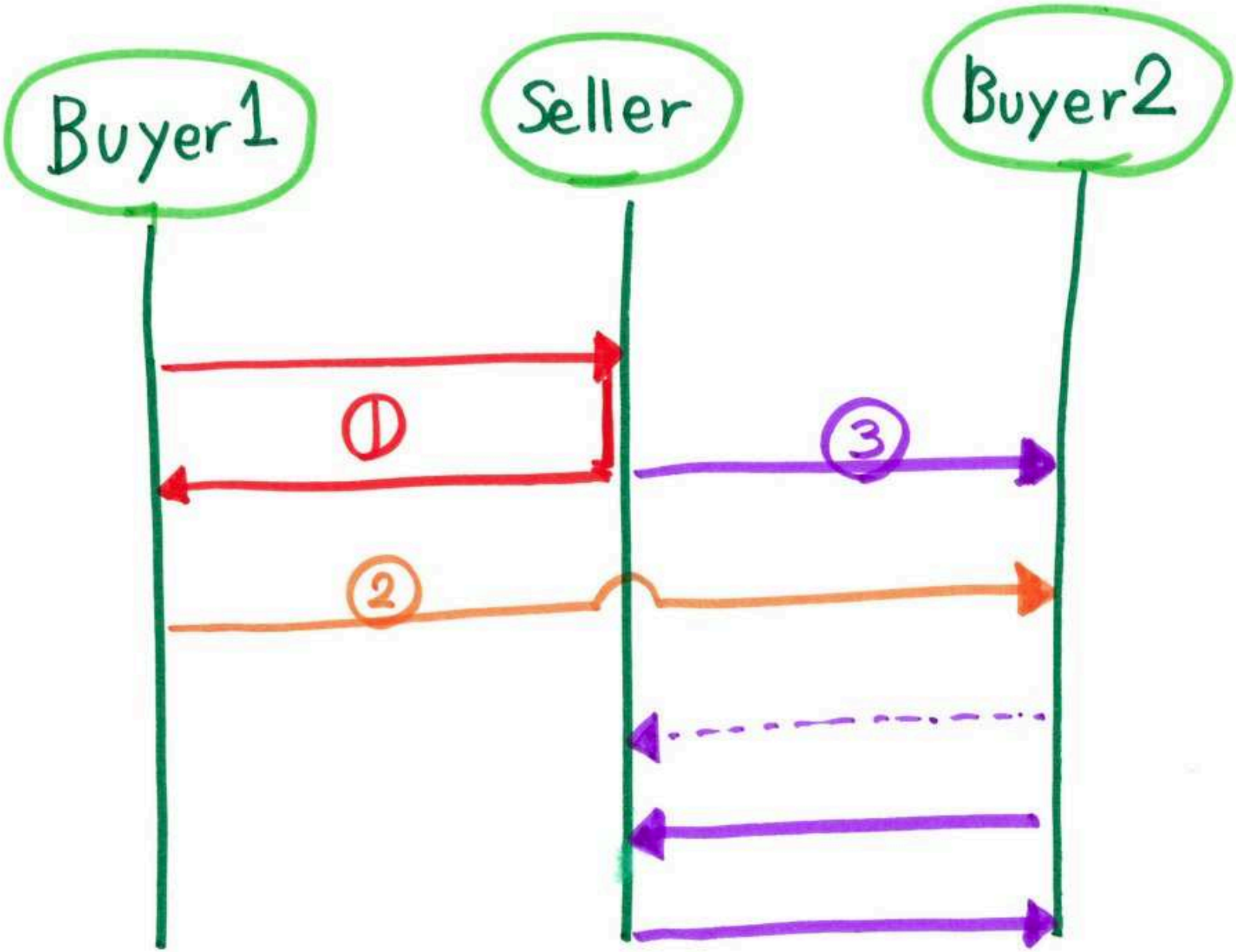


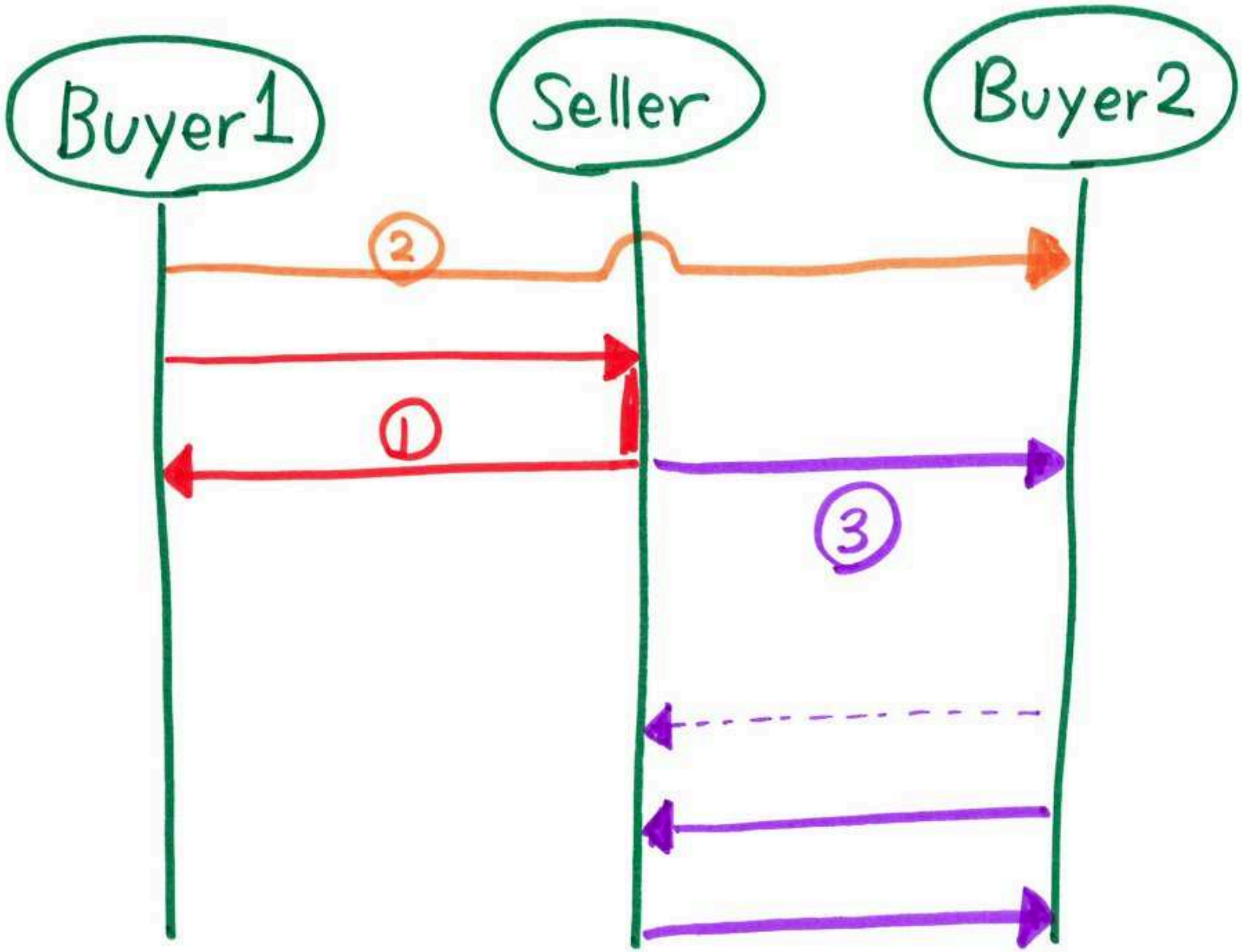
! String ; ? Int ; ⊕ { ok : ! String ; ? Date ; end , quit : end }

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

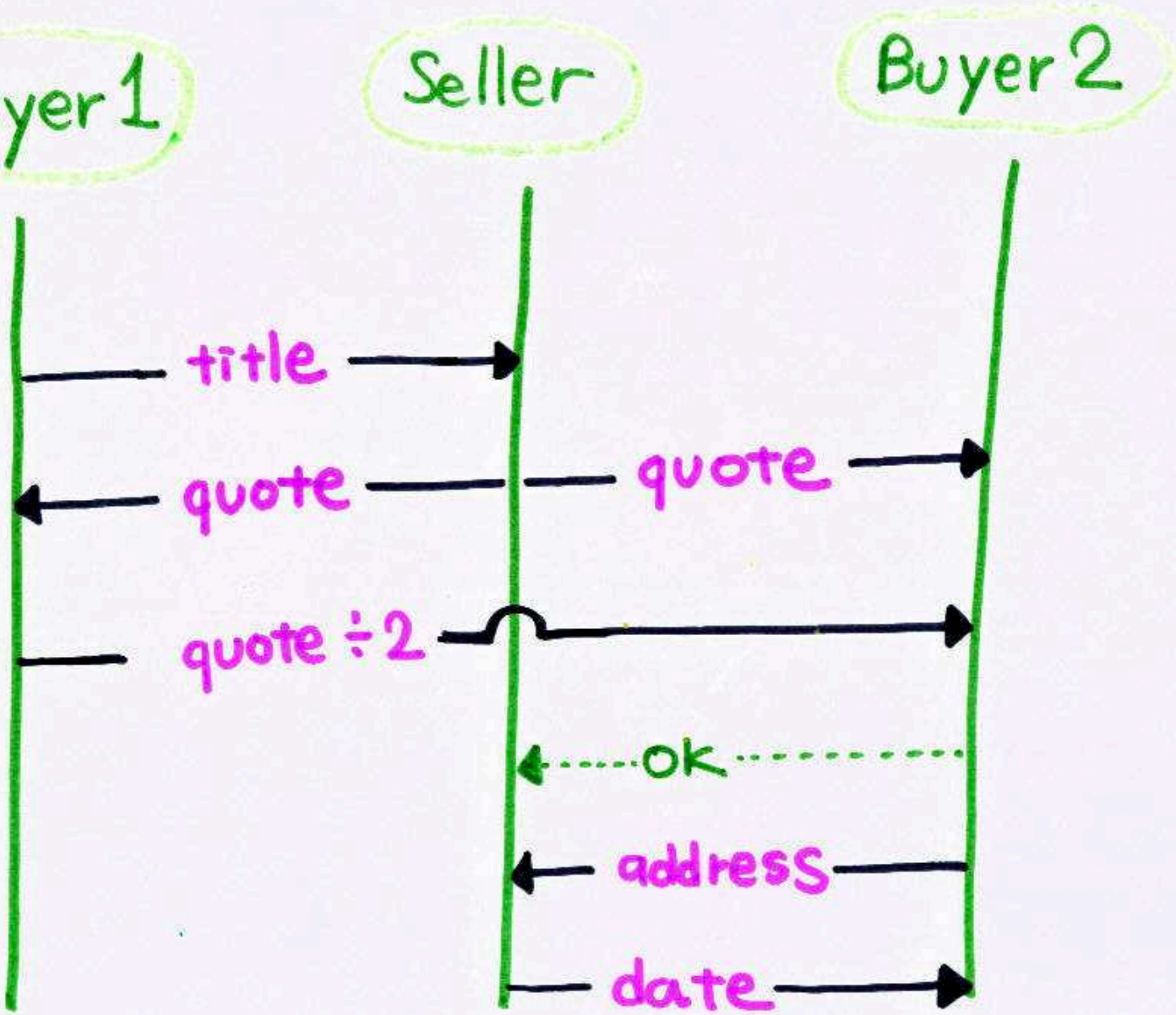
Multiparty Session Types



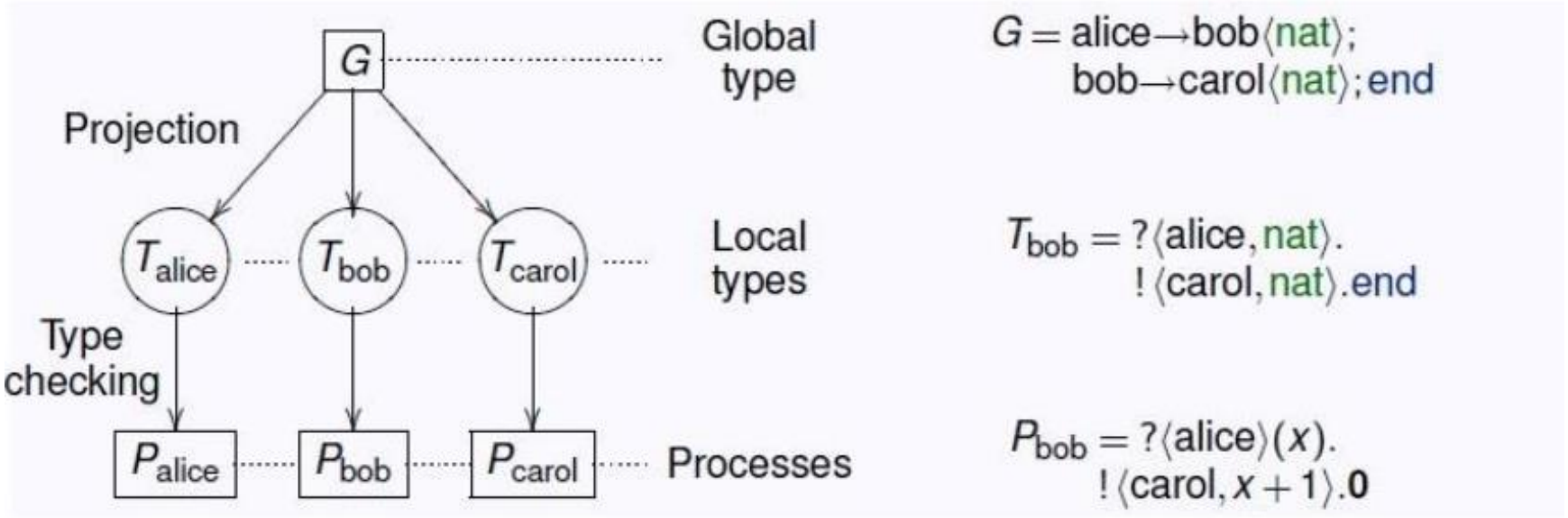

Buyer1
Seller
Buyer2
1
3
2


Buyer1
Seller
Buyer2
2
1
3

Multiparty Session Types

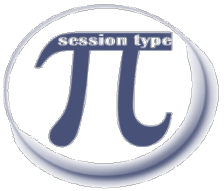


Session Types Overview



- Properties
  - Communication safety (no communication mismatch)
  - Communication fidelity (the communication follow the protocol)
  - Progress (no deadlock/stuck in a session)

Evolution Of MPST

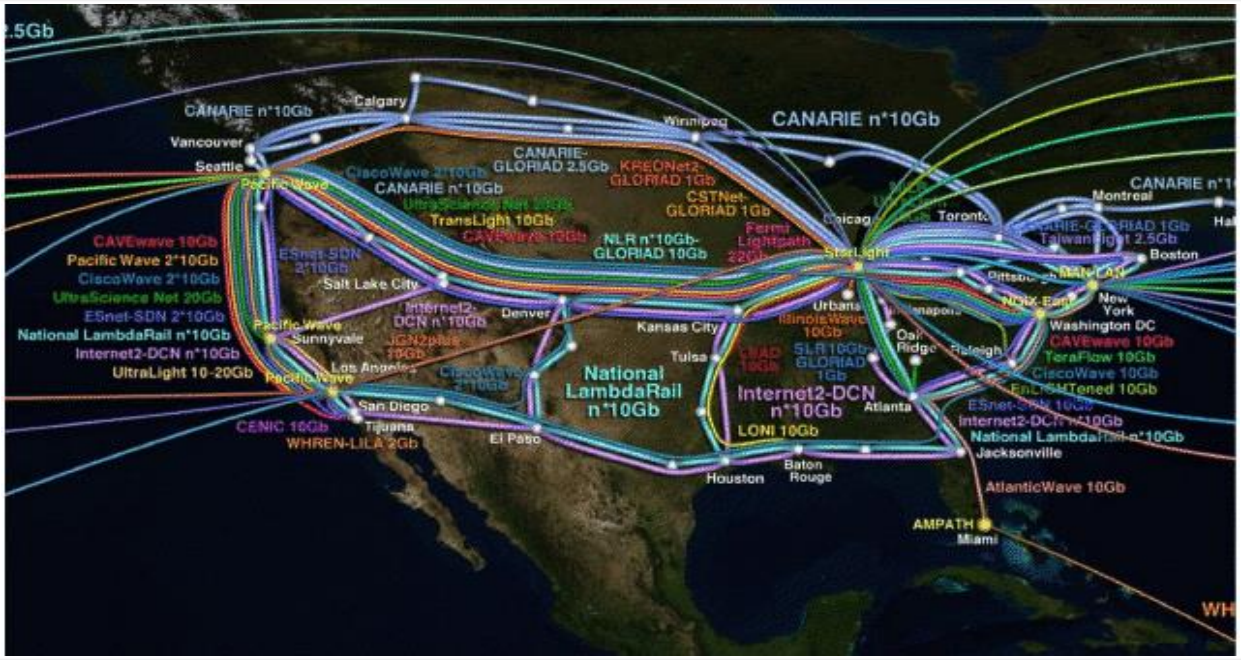


- Binary Session Types [THK98, HVK98]
- Multiparty Session Types [POPL'08]
- A Theory of Design-by-Contract for Distributed Multiparty Interactions [Concur'11]
- Multiparty Session Types Meet Communicating Automata [ESOP'12, ICALP'13]
- Network Monitoring through Multiparty Session Types [FMOODS'13]

- SPY: Local Verification of Global Protocols [RV'13]
- Distributed Runtime Verification with Session Types and Python [RV'13]

Ocean Observatory Initiative (OOI)

OOI aims: to deploy an infrastructure (global network) to expand the scientists' ability to remotely study the ocean



Usage: Integrate real-time data acquisition, processing and data storage for ocean research,…

OOI: verification challenges

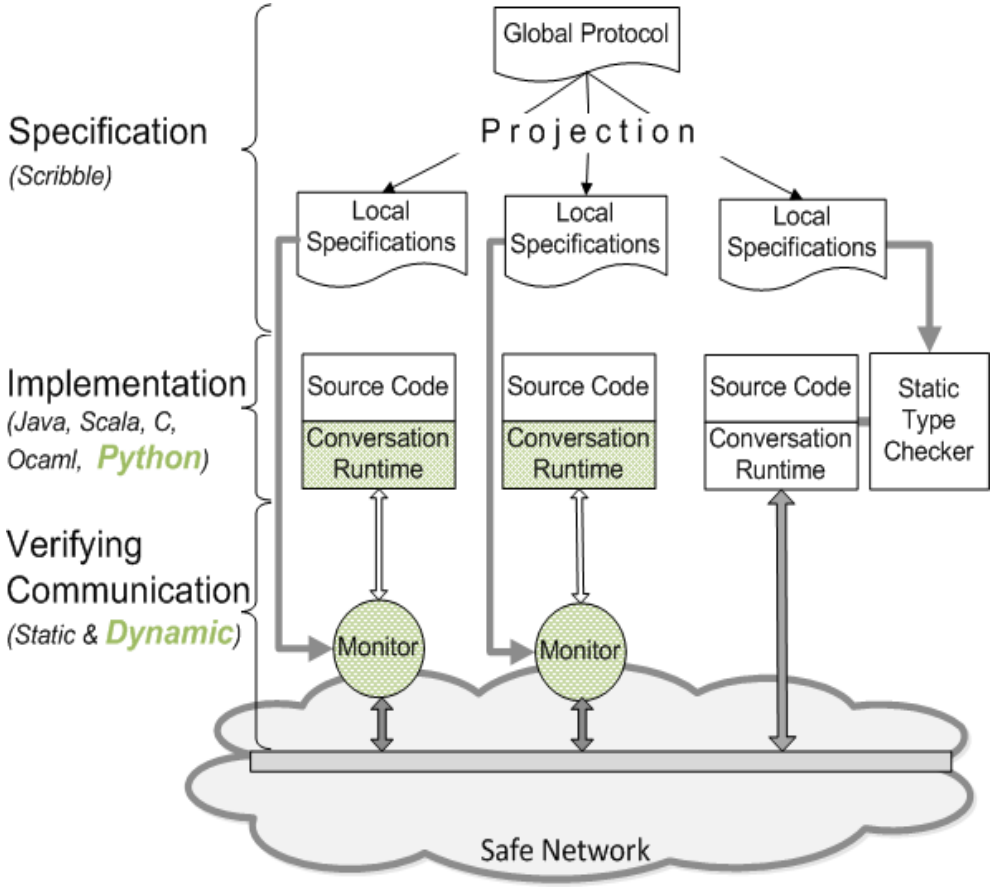
- applications written in **different** languages, running on **heterogeneous** hardware in an **asynchronous** network.
- different authentication domains, external **untrusted** applications
- various distributed protocols
- requires correct, safe interactions

Session Types for Runtime Verification

- Methodology
  - Developers design

protocols in a dedicated language - Scribble

  - Well-fomedness is checked by Scribble tools
  - Protocols are projected into local types
  - Local types generate monitors



Content

1. Writing correct global protocols with **Scribble Compiler**

2. Verify programs via ***local monitors***

3. Build additional verification modules via ***annotations***

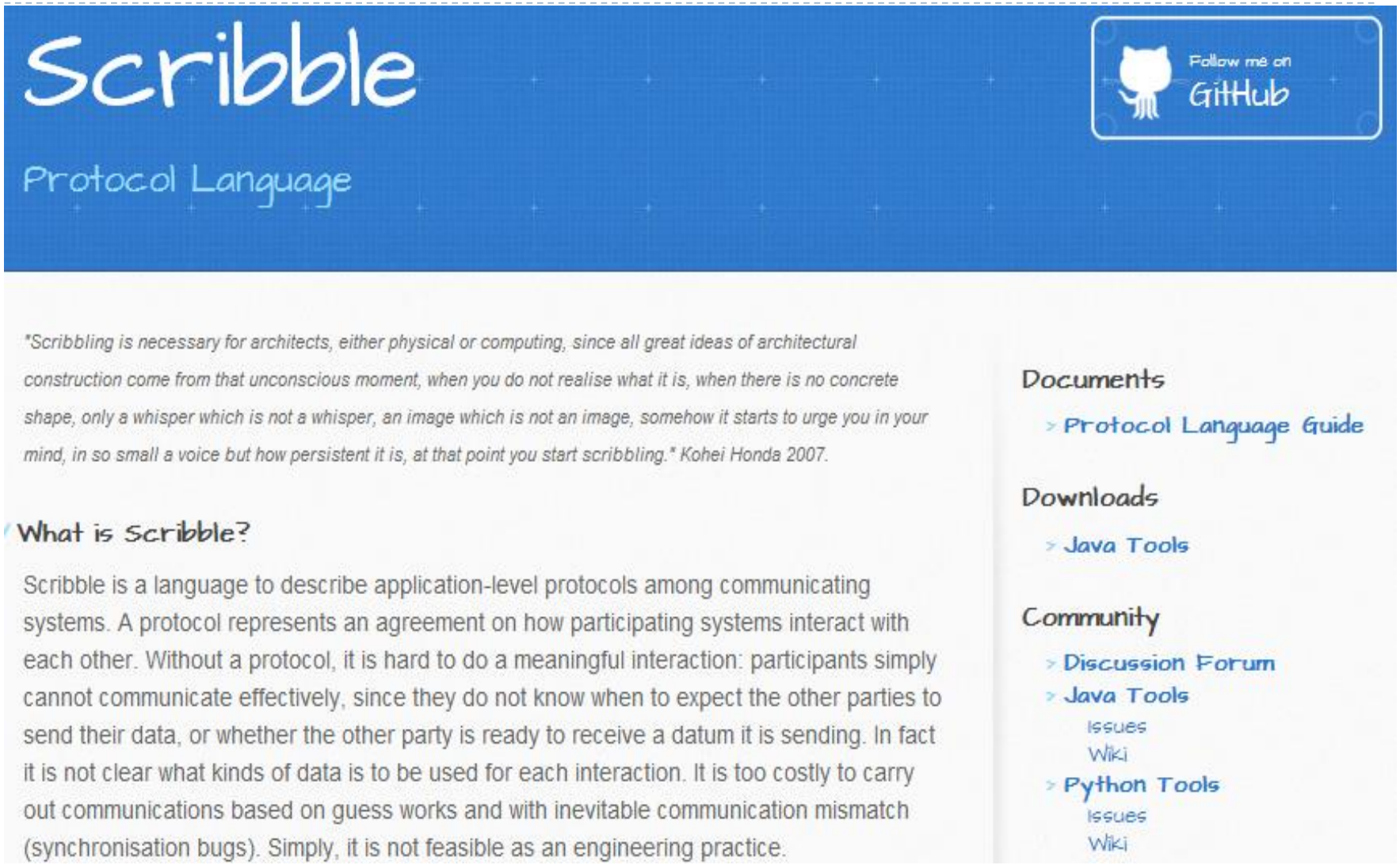
Content

1. Writing correct global protocols with **Scribble Compiler**

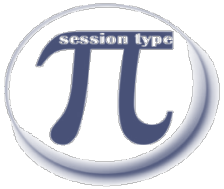
2. Verify programs via ***local monitors***

3. Build additional verification modules via ***annotations***

Meet Scribble



A Global Protocol



Import type ⟶

protocol def ⟶

send-receive ⟶

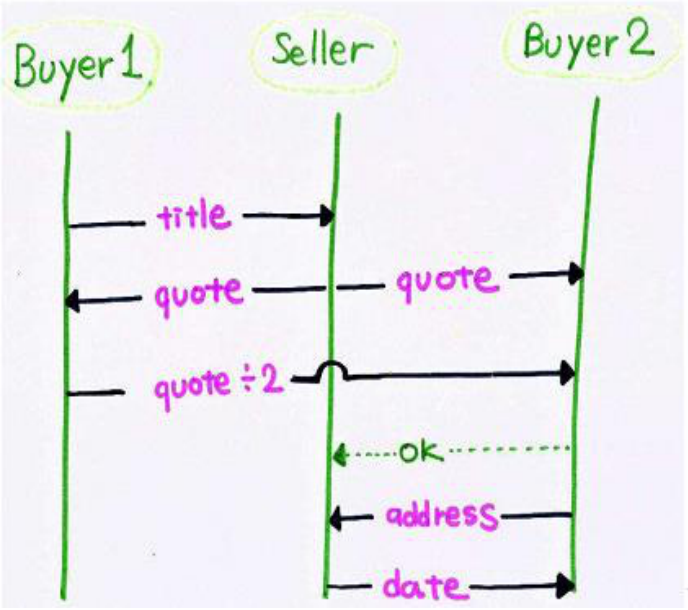
recursion ⟶

choice ⟶

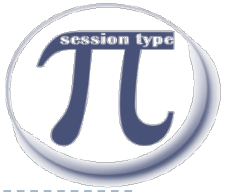
```
type <python> "StringType" from "Lib/types.py" as str;

global protocol Negotiation(role P, role R, role A) {
    offer(string) from P to R;
    offer(string) from R to A;
    (string) from A to R;
    rec START {
        choice at R {
            accept() from R to P;
            confirm() from P to R;
        } or {
            offer(string) from R to P;
            (conditions:string) from P to R;
            continue START;
        } or {
            reject() from R to P;
            confirm() from R to P;}}}
```

Two Buyer Protocol in Scribble

```
module Bookstore;

type <java> "java.lang.Integer" from "rt.jar" as Integer;
type <java> "java.lang.String" from "rt.jar" as String;

global protocol TwoBuyers(role A, role B, role S) {
    title(String) from A to S;
    quote(Integer) from S to A, B;
    rec LOOP {
        share(Integer) from A to B;
        choice at B {
            accept(address:String) from B to A, S;
            date(String) from S to B;
        } or {
            retry() from B to A, S;
            continue LOOP;
        } or {
            quit() from B to A, S;
}   }   }
```



Buyer: A local projection




```
module Bookstore_TwoBuyers_A;

type <java> "java.lang.Integer" from "rt.jar" as Integer;
type <java> "java.lang.String" from "rt.jar" as String;

local protocol TwoBuyers_A at A(role A, role B, role S) {
 title(String) to S;
 quote(Integer) from S;
 rec LOOP {
  share(Integer) to B;
  choice at B {
   accept(address:String) from B;
  } or {
   retry() from B;
   continue LOOP;
  } or {
   quit() from B;
} } }
```

Global protocol well-formedness 1/2

```
global protocol ChoiceAmbiguous(role A, role B, role C) {
  choice at A {
    m1() from A to B; // X
    m2() from B to C;
    m3() from C to A;
  } or {
    m1() from A to B; // X
    m5() from B to C;
    m6() from C to A;
} }
```

```
global protocol ChoiceNotCommunicated(role A, role B, role C) {
  choice at A {
    m1() from A to B;
    m2() from B to C; // X
  } or {
    m4() from A to B;
} }
```

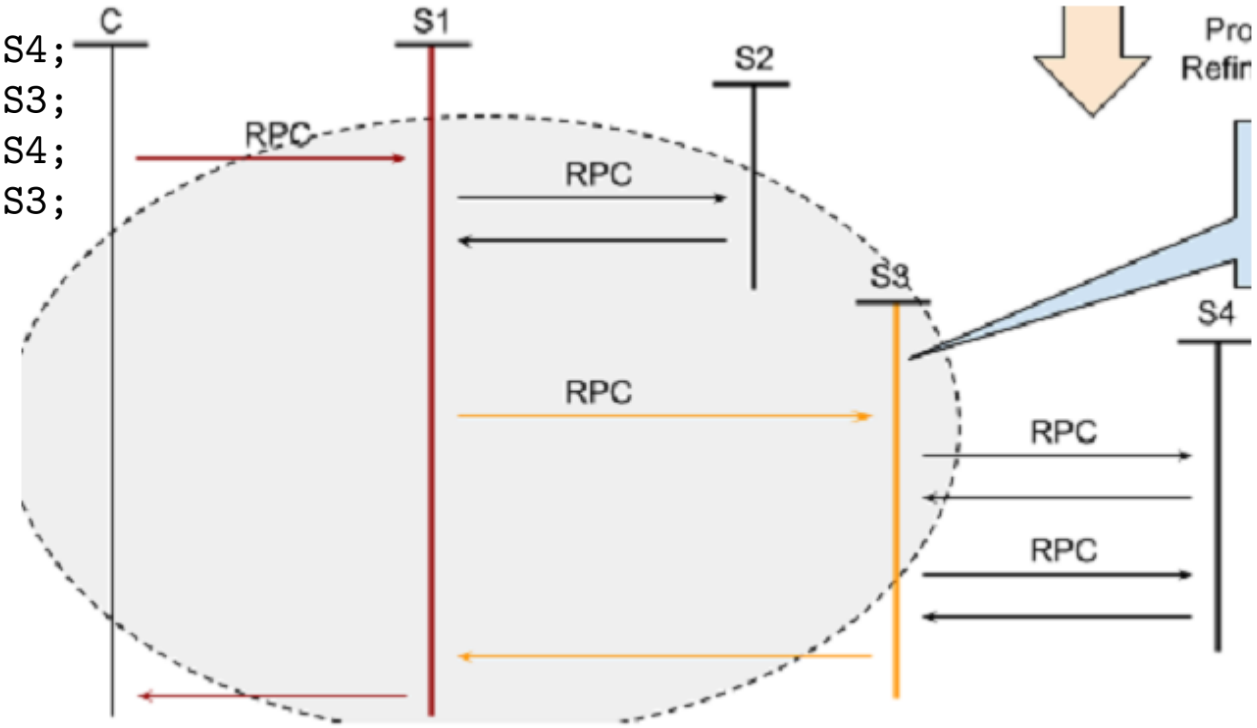
Global protocol well-formedness 2/2

```
global protocol ParallelNotLinear(role A, role B, role C) {
  par {
    m1() from A to B; // X
    m2() from B to C;
  } and {
    m1() from A to B; // X
    m4() from B to C;
} }
```

```
global protocol RecursionNoExit(role A, role B, role C, role D) {
  rec X {
    m1() from A to B;
    continue X;
  }
  m2() from A to B; // Unreachable for A, B
  m3() from C to D;
}
```

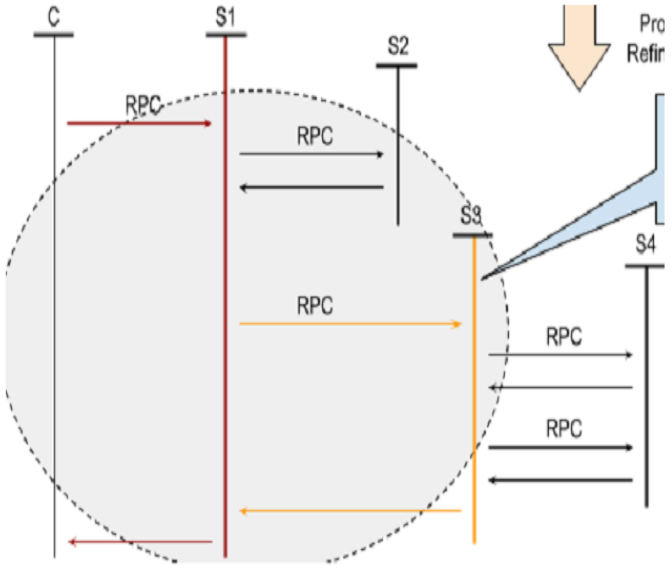
Application-level service call composition

```
// Direct specification
global protocol P3(role C, role S1, role S2, role S3, role S4)
{
    () from C to S1;
        () from S1 to S2;
        () from S2 to S1;
        () from S1 to S3;
            () from S3 to S4;
            () from S4 to S3;
            () from S3 to S4;
            () from S4 to S3;
        () from S3 to S1;
    () from S1 to C;
}
```



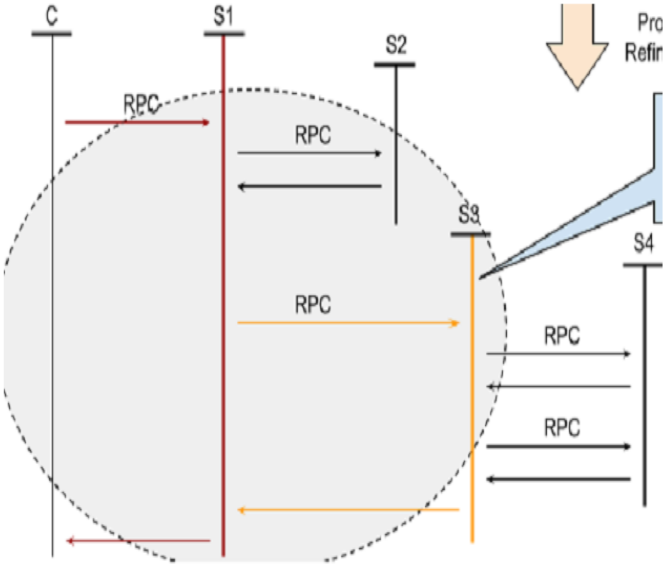
Scoping

```
global protocol ServiceCall(role Client, role Service) {
    () from Client to Server;
    () from Server to Client;
}

// By composing basic ServiceCalls
global protocol P2(role C, role S1, role S2, role S3, role S4)
{
    () from C to S1;
        do ServiceCall(S1 as Client, S2 as Server);
        () from S1 to S3;
            do ServiceCall(S3 as Client, S4 as Server);
            do ServiceCall(S3 as Client, S4 as Serve
        () from S3 to S1;
    () from S1 to C;
}
```

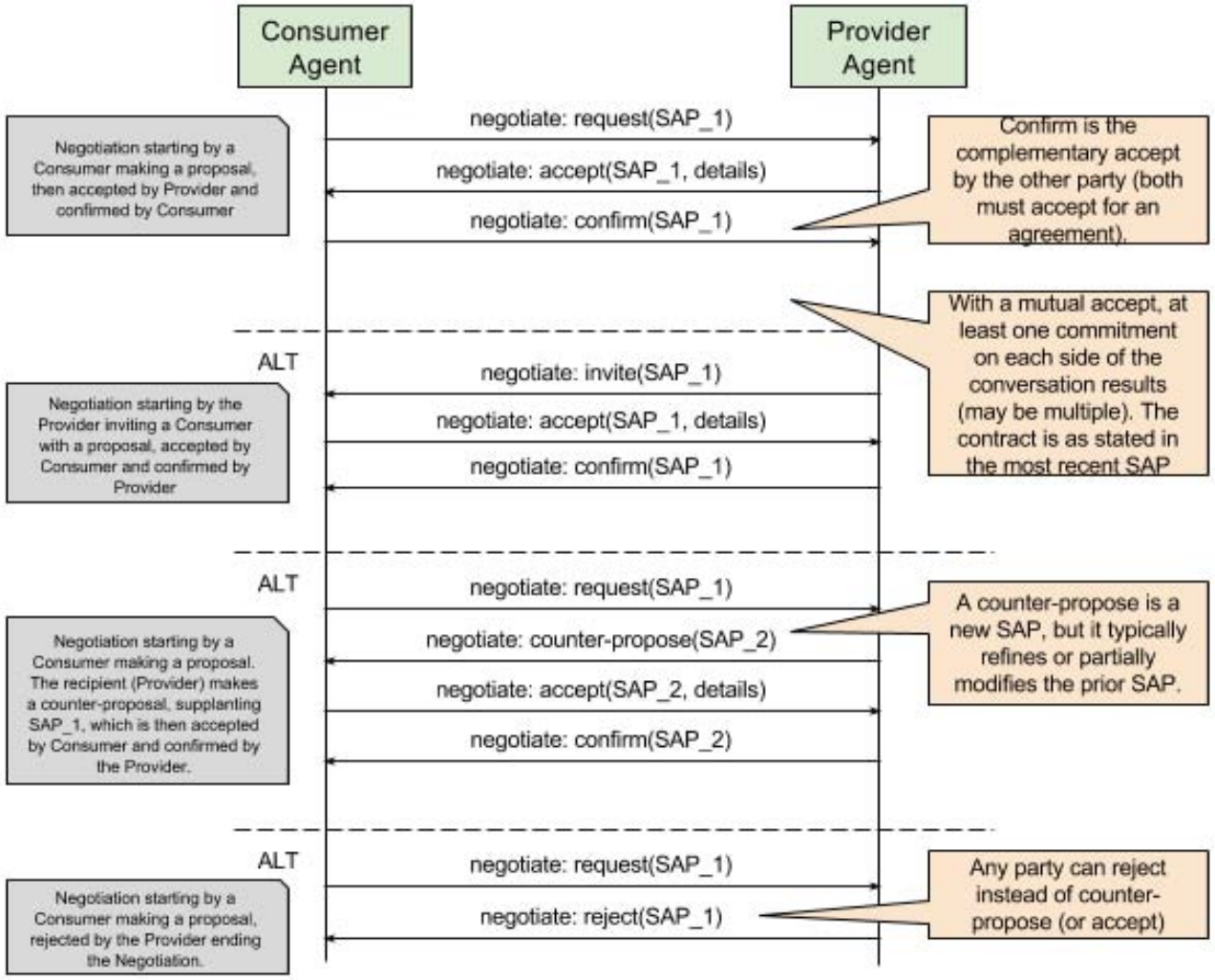


Scoping

```
// "Middleman" pattern
global protocol Middleman(
    role L, role M, role R, role S)
{
 () from L to M;
  do ServiceCall(M as Client, S as Server);
  do ServiceCall(M as Client, S as Server);
 () from M to R;
}
// By composing ServiceCall and Middleman patterns
global protocol P3(role C, role S1, role S2, role S3, role S4)
{
    () from C to S1;
        do ServiceCall(S1 as Client, S2 as Server);
        do Middleman(S1 as L, S3 as M, S4 as R);
    () from S1 to C;
}
```



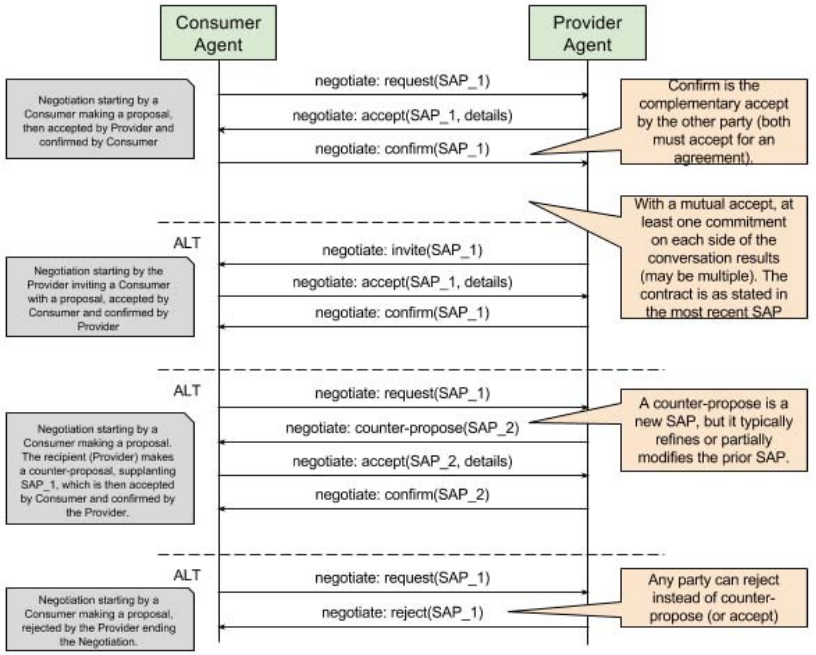
OOI agent negotiation 1/5



- `https://confluence.oceanobservatories.org/display/syseng/CIAD+COI+OV+Negotiate+Protocol`

OOI agent negotiation 2/5

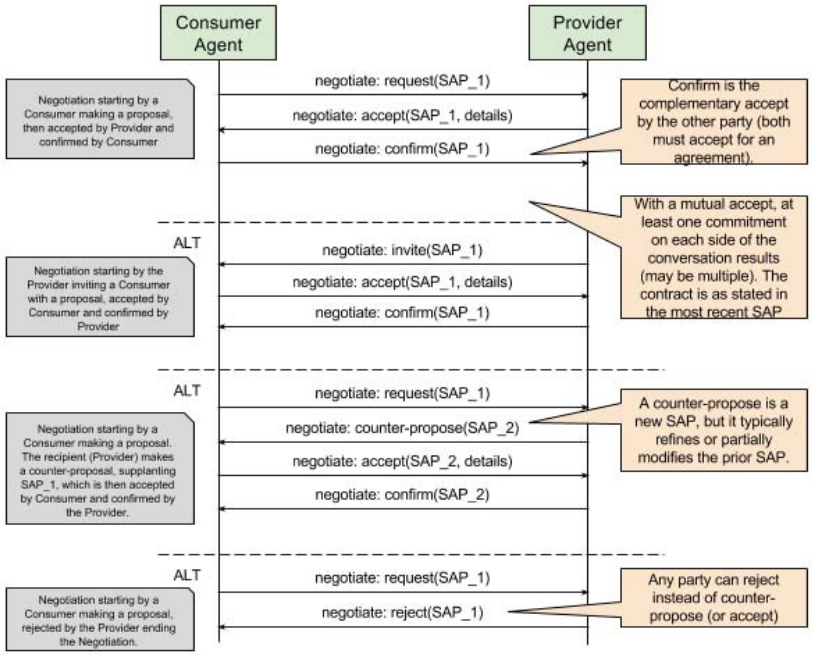
```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

global protocol Negotiate(role Consumer as C, role Producer as P) {
```




```
}
```

OOI agent negotiation 3/5 (choice)

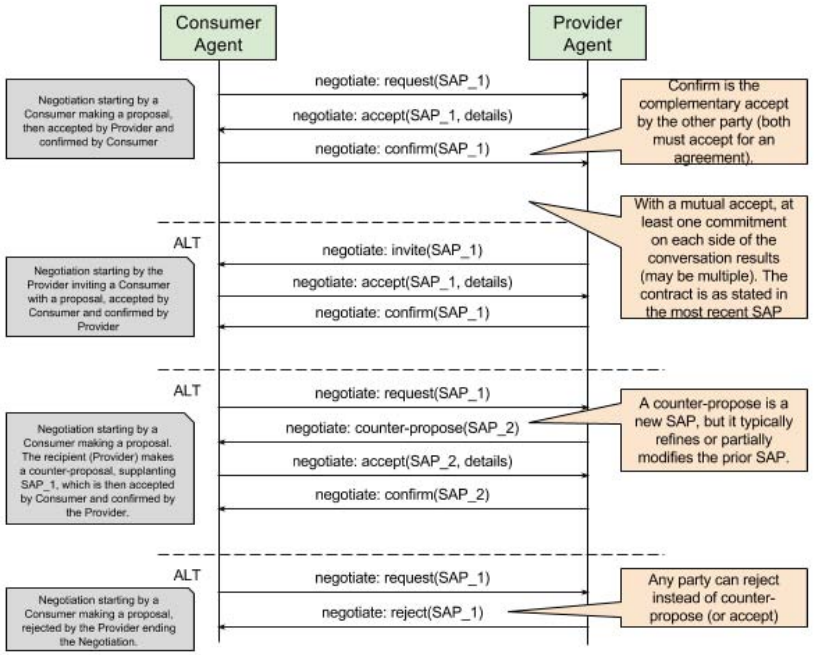
```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

global protocol Negotiate(role Consumer as C, role Producer as P) {
  propose(SAP) from C to P;

    choice at P {
      accept() from P to C;
      confirm() from C to P;
    } or {
      reject() from P to C;
    } or {
      propose(SAP) from P to C;
```




```
} }
```

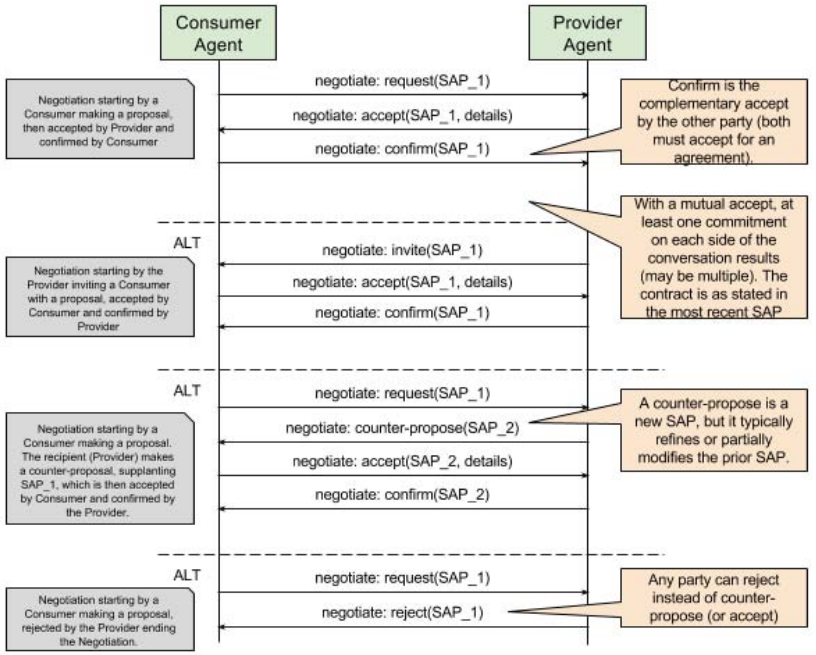
OOI agent negotiation 4/5

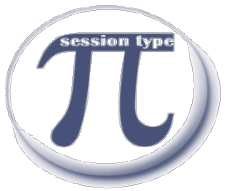
```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

global protocol Negotiate(role Consumer as C, role Producer as P) {
  propose(SAP) from C to P;

    choice at P {
      accept() from P to C;
      confirm() from C to P;
    } or {
      reject() from P to C;
    } or {
      propose(SAP) from P to C;
      choice at C {
        accept() from C to P;
        confirm() from P to C;
      } or {
        reject() from C to P;
      } or {
        propose(SAP) from C to P;

} } }
```



OOI agent negotiation 5/5 (recursion)

```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

global protocol Negotiate(role Consumer as C, role Producer as P) {
  propose(SAP) from C to P;
  rec X {
    choice at P {
      accept() from P to C;
      confirm() from C to P;
    } or {
      reject() from P to C;
    } or {
      propose(SAP) from P to C;
      choice at C {
        accept() from C to P;
        confirm() from P to C;
      } or {
        reject() from C to P;
      } or {
        propose(SAP) from C to P;
        continue X;
      }
    }
```



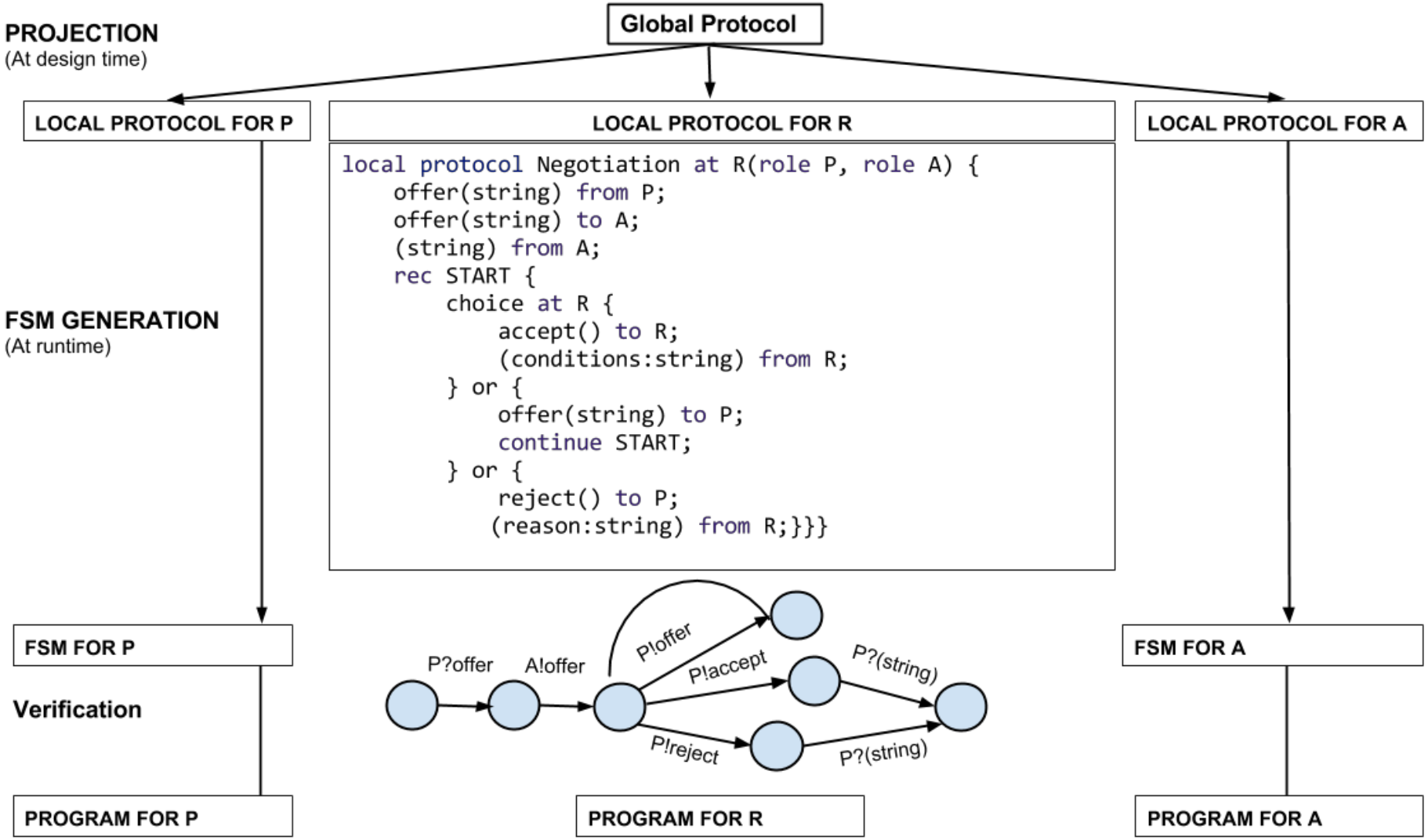


1. Writing correct global protocols with **Scribble Compiler**

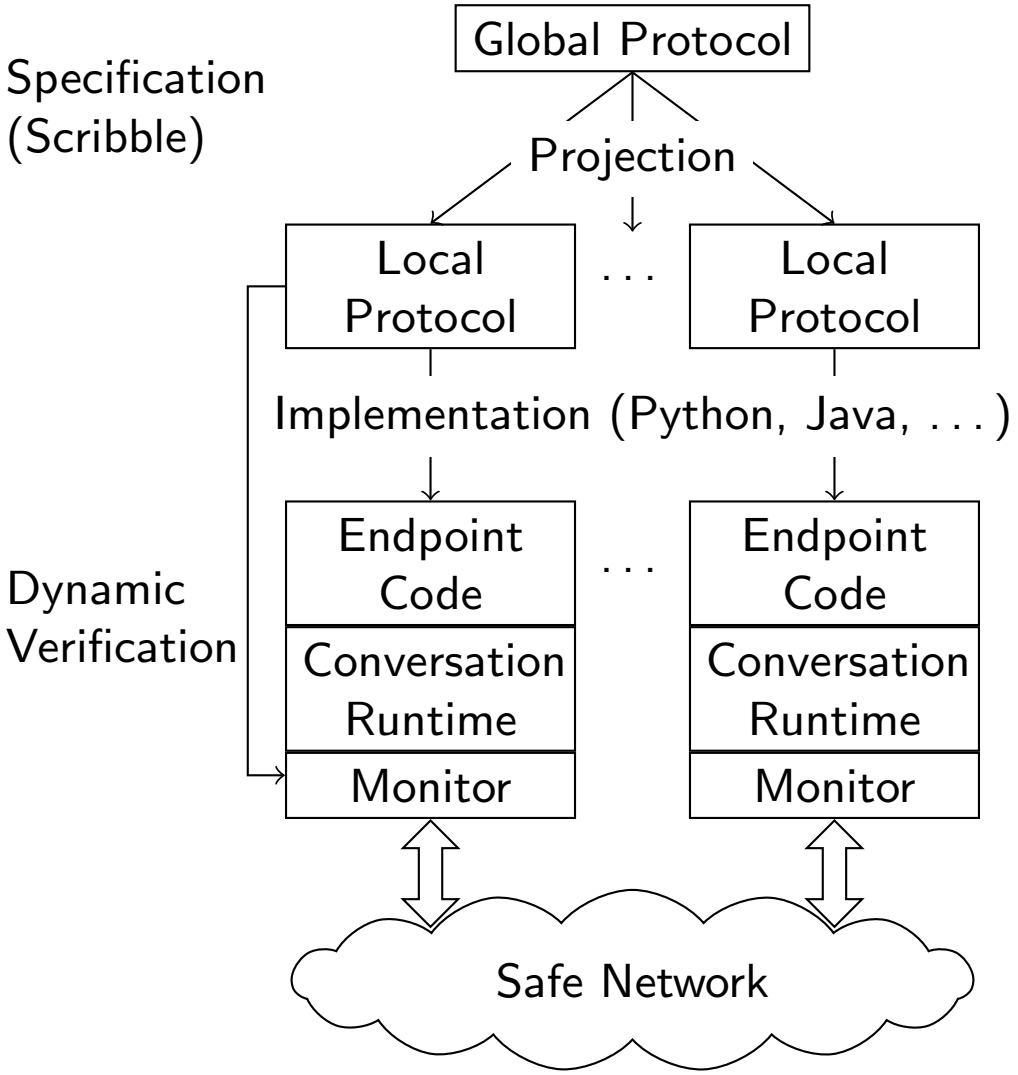
2. Verify programs via ***local monitors***

3. Build additional verification modules via ***annotations***

Local Protocol Conformance



The Scribble Framework



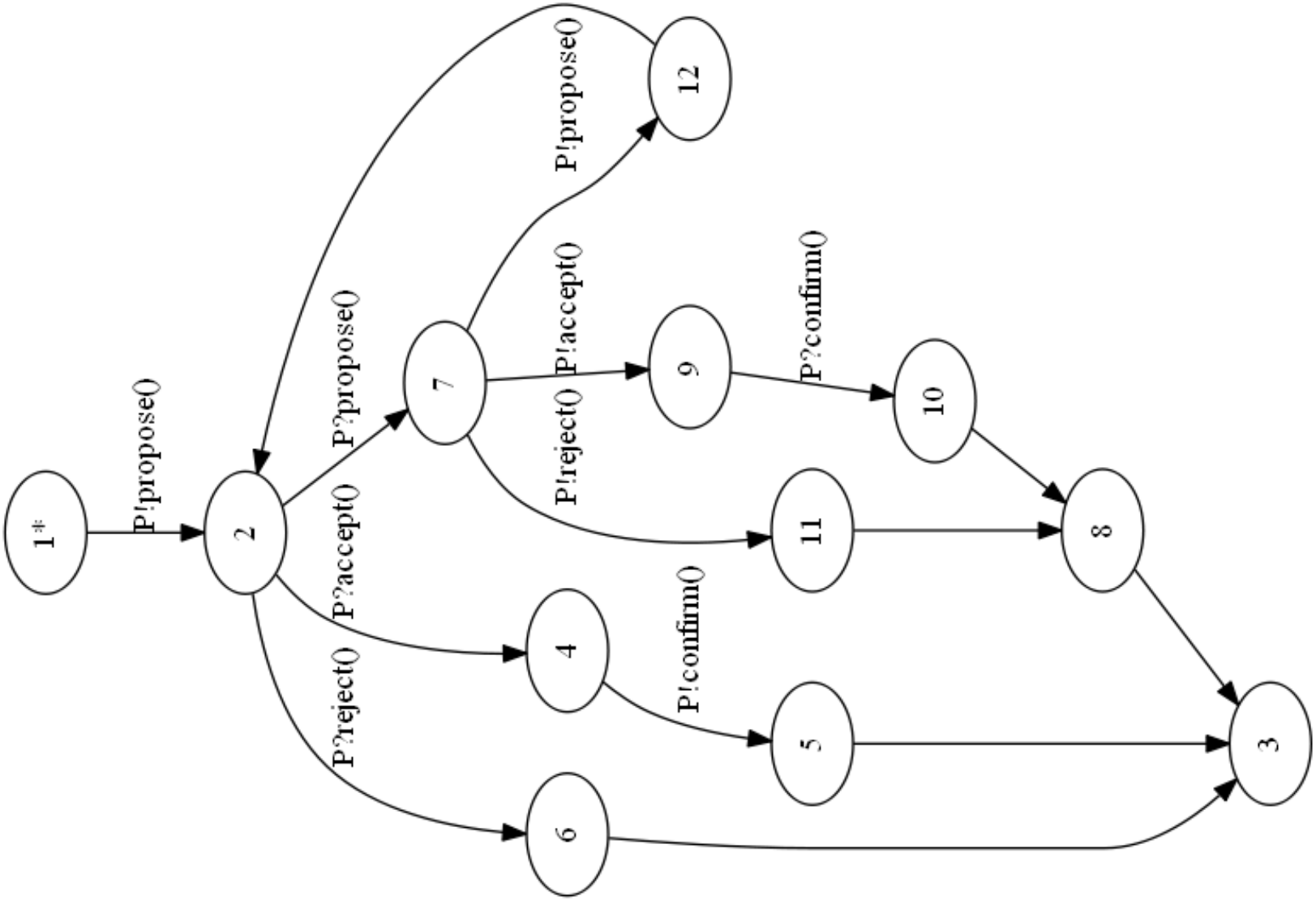
- Scribble global protocols
  - Well-formedness validation
- Scribble local protocols
  - FSM generation (for endpoint monitoring)
- (Heterogeneous) endpoint programs
  - Scribble Conversation API
  - (Interoperable) Distributed Conversation Runtime

Local protocol projection (Negotiation Consumer)

```
// Global
propose(SAP) from C to P;
rec START {
  choice at P {
    accept() from P to C;
    confirm() from C to P;
  } or {
    reject() from P to C;
  } or {
    propose(SAP) from P to C;
    choice at C {
      accept() from C to P;
      confirm() from P to C;
    } or {
      reject() from C to P;
    } or {
      propose(SAP) from C to P;
      continue START;
} } }
```

```
// Projection for Consumer
propose(SAP) to P;
rec START {
  choice at P {
    accept() from P;
    confirm() to P;
  } or {
    reject() from P;
  } or {
    propose(SAP) from P;
    choice at C {
      accept() to P;
      confirm() from P;
    } or {
      reject() to P;
    } or {
      propose(SAP) to P;
      continue START;
} } }
```

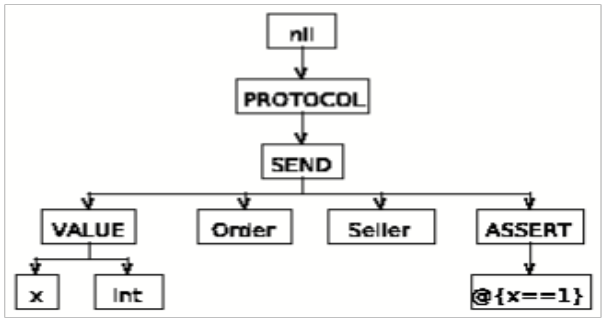
FSM generation (Negotiation Consumer)



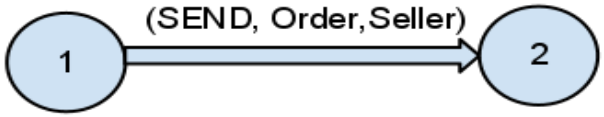
FSM Generator

Scribble: Order(x:int) to Seller @{x==1}

AST:

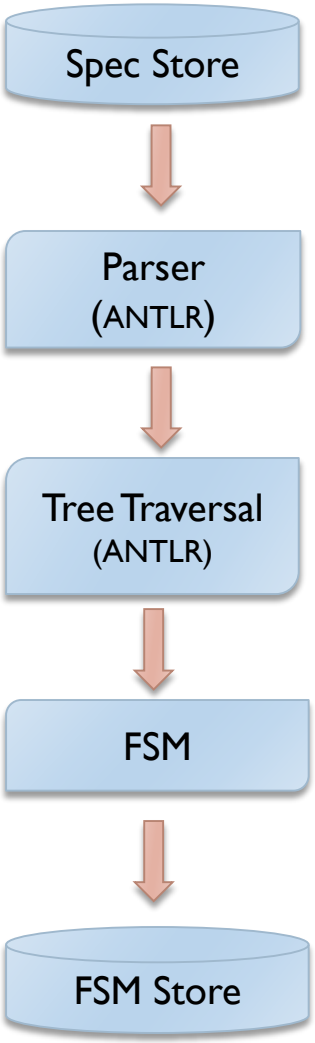


FSM:

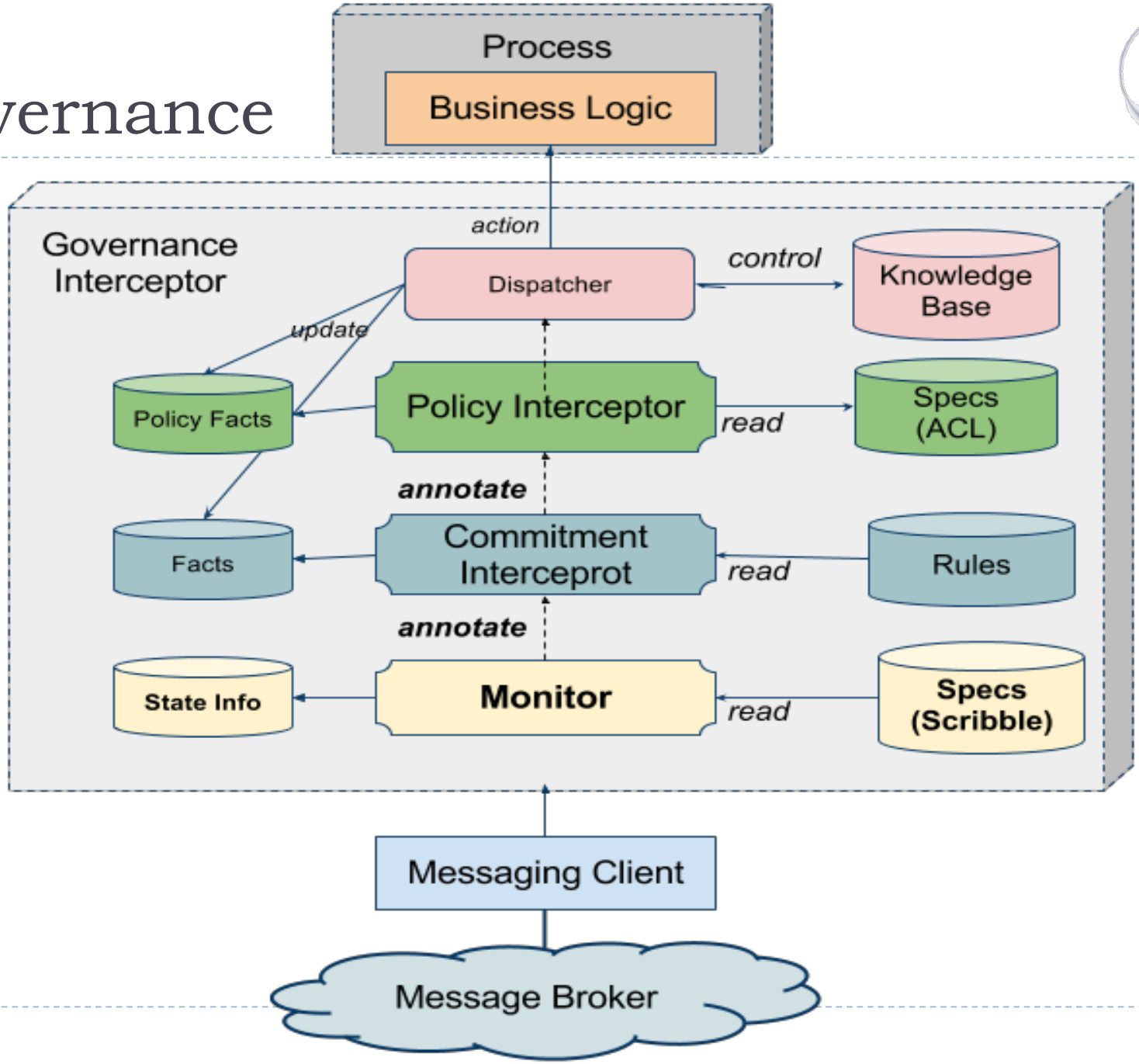


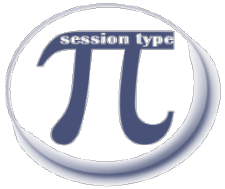
FSM transition_table:

```
(1, (send, order, seller) ->
    (2, assertion_object, {'x':'int'})
```



Governance


Process
Business Logic
Governance Interceptor
action
Dispatcher
control
Knowledge Base
update
Policy Facts
Policy Interceptor
read
Specs (ACL)
annotate
Facts
Commitment Interceprot
read
Rules
annotate
State Info
Monitor
read
Specs (Scribble)
Messaging Client
Message Broker



1. Writing correct global protocols with **Scribble Compiler**

2. Verify programs via ***local monitors***

3. Build additional verification modules via ***annotations***

Validation via Annotations

Annotations = @{Logic} Scribble Construct

```
…
@{assert: payment + overdraft>=1000}
offer(payment: int) from C to I;
…
```

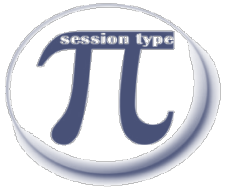
```
…
@{deadline: 5s}
offer(conditions string) from C to I;
…
```

```
…
rec Loop {
@{guard: repeat<10}
propose(string) from C to I;
…
```

- The monitor passes {'type':param, …}

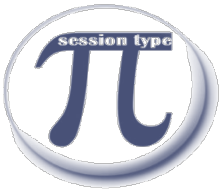
  to the upper layers
- Upper layers recognize and process the annotation type or discard it
- Statefull assertion

Scribble Community



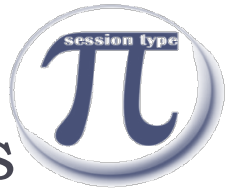
- Webpage:
  - www.scribble.org
- GitHub:
  - https://github.com/scribble
- Tutorial:
  - www.doc.ic.ac.uk/~rhu/scribble/tutorial.html
- Specification (0.3)
  - www.doc.ic.ac.uk/~rhu/scribble/langref.html

A theory for network monitoring



- Formalise MPST-monitoring and asynchronous networks.
- Introduce monitors as first-class objects in the theory
- Justify monitoring by soundness theorems.
  - Safety
    - monitors enforces specification conformance.
  - Transparency
    - monitors does not affect correct behaviours.
  - Fidelity
    - correspondence to global types is maintained.

Multiparty Sessions for Runtime Monitors



$$
\begin{array}{lll}
A & ::= & \mathtt{tt} \mid \mathtt{ff} \mid e_1 = e_2 \mid e_1 < e_2 \mid \neg A \mid A_1 \wedge A_2 \mid A_1 \vee A_2 \\
e & ::= & v \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid e_1 \ \mathtt{mod}\ e_2 \qquad S ::= \mathsf{bool} \mid \mathsf{int} \mid \mathsf{string} \\
G & ::= & \mathtt{r}_1 \to \mathtt{r}_2 : \{l_i(x_i : S_i)\{A_i\}.G_i\}_{i \in I} \mid G_1 \mid G_2 \mid G_1; G_2 \mid \mu \mathtt{t}.G \mid \mathtt{t} \mid \epsilon \mid \mathsf{end} \\
T & ::= & \mathtt{r}!\{l_i(x_i : S_i)\{A_i\}.T_i\}_{i \in I} \mid \mathtt{r}?\{l_i(x_i : S_i)\{A_i\}.T_i\}_{i \in I} \mid T_1 \mid T_2 \mid T_1; T_2 \mid \\
 & & \mu \mathtt{t}.T \mid \mathtt{t} \mid \epsilon \mid \mathsf{end} \\
P & ::= & \overline{a}\langle s[\mathtt{r}] : T \rangle \mid a(y[\mathtt{r}] : T).P \mid k[\mathtt{r}_1, \mathtt{r}_2]!l\langle e \rangle \mid k[\mathtt{r}_1, \mathtt{r}_2]?\{l_i(x_i).P_i\}_{i \in I} \mid \\
 & & \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid P \mid Q \mid \mathbf{0} \mid \mu X.P \mid X \mid P; Q \mid (\nu a)\ P \mid (\nu s)P \\
N & ::= & [P]_\alpha \mid N_1 | N_2 \mid \mathbf{0} \mid (\nu a)N \mid (\nu s)N \mid \langle r\ ;\ h \rangle \\
r & ::= & a \mapsto \alpha \mid s[\mathtt{r}] \mapsto \alpha \qquad h ::= m \cdot h \mid \emptyset \qquad m ::= \overline{a}\langle s[\mathtt{r}] : T \rangle \mid s\langle \mathtt{r}_1, \mathtt{r}_2, l\langle v \rangle \rangle
\end{array}
$$

Formal Semantics

$$[\overline{a}\langle s[\mathrm{r}] : T\rangle]_\alpha \mid \langle r \;;\; h\rangle \longrightarrow [\mathbf{0}]_\alpha \mid \langle r \;;\; h \cdot \overline{a}\langle s[\mathrm{r}] : T\rangle\rangle$$

$$[a(y[\mathrm{r}] : T).P]_\alpha \mid \langle r \;;\; \overline{a}\langle s[\mathrm{r}] : T\rangle \cdot h\rangle \longrightarrow [P[s/y]]_\alpha \mid \langle r{\cdot}s[\mathrm{r}] \mapsto \alpha \;;\; h\rangle \;^{\dagger}$$

$$[s[\mathrm{r}_1, \mathrm{r}_2]!l_j\langle v\rangle]_\alpha \mid \langle r \;;\; h\rangle \longrightarrow [\mathbf{0}]_\alpha \mid \langle r \;;\; h \cdot s\langle \mathrm{r}_1, \mathrm{r}_2, l_j\langle v\rangle\rangle\rangle \;^{\dagger\dagger}$$

$$[s[\mathrm{r}_1, \mathrm{r}_2]?\{l_i(x_i).P_i\}_i]_\alpha \mid \langle r \;;\; s\langle \mathrm{r}_1, \mathrm{r}_2, l_j\langle v\rangle\rangle \cdot h\rangle \longrightarrow [P_j[v/x_j]]_\alpha \mid \langle r \;;\; h\rangle \;^{\dagger\dagger\dagger}$$

$$\dagger : r(a) = \alpha \qquad \dagger\dagger : r(s[\mathrm{r}_2]) \neq \alpha \qquad \dagger\,\dagger\dagger : r(s[\mathrm{r}_2]) = \alpha$$

- processes P located at principals α
  - Abstracts local applications
- router r
  - abstracts network routing information updated on-the-fly

Formalism: Monitor

- Specifications

$$\Sigma ::= \emptyset \mid \Sigma, \alpha : \langle \Gamma ; \Delta \rangle,$$
$$\Gamma ::= \emptyset \mid \Gamma, a :?(T[\mathtt{r}]) \mid \Gamma, a :!(T[\mathtt{r}]) \quad \Delta ::= \emptyset \mid \Delta, s[\mathtt{r}] : T,$$

Σ: spec., Δ: session env, Γ: shared env.

- Monitors

$$\mathsf{M} = \alpha : \langle \Gamma ; \Delta \rangle$$

  - Monitors are introduced as component of monitored networks

$$\frac{\mathsf{M} \xrightarrow{s[\mathtt{r}_1, \mathtt{r}_2]!l\langle v \rangle} \mathsf{M}' \quad r(s[\mathtt{r}_2]) \neq \alpha}{[s[\mathtt{r}_1, \mathtt{r}_2]!l\langle v \rangle]_\alpha \mid \mathsf{M} \mid \langle r \; ; \; h \rangle \longrightarrow [\mathbf{0}]_\alpha \mid \mathsf{M}' \mid \langle r \; ; \; h \cdot s\langle \mathtt{r}_1, \mathtt{r}_2, l\langle v \rangle \rangle \rangle}$$

$$\frac{\mathsf{M} \not\xrightarrow{s[\mathtt{r}_1, \mathtt{r}_2]!l\langle v \rangle}}{[s[\mathtt{r}_1, \mathtt{r}_2]!l\langle v \rangle]_\alpha \mid \mathsf{M} \mid \langle r \; ; \; h \rangle \longrightarrow [\mathbf{0}]_\alpha \mid \mathsf{M} \mid \langle r \; ; \; h \rangle}$$

Satisfaction

The satisfaction relation $\models N : \Sigma$ relates networks and specification:

- if Σ expects an input, N should be able to process it.
- if N performs an output, Σ should be expecting it.
- still holds after reduction (coinductive definition).

Satisfaction equivalence

If $N_1 \cong N_2$ and $\models N_1 : \Sigma$ then $\models N_2 : \Sigma$.

Results (Safety)

Local Safety

$\models [P]_\alpha \mid \mathsf{M} : \alpha : \langle \Gamma; \Delta \rangle$ with $\mathsf{M} = \alpha : \langle \Gamma; \Delta \rangle$.

- A monitored process satisfies its specification.

Global Safety

If N is fully monitored w.r.t. Σ, then $\models \mathsf{N} : \Sigma$.

- monitored networks behave as expected.

Results (Transparency)

Local Transparency

If $\models [P]_\alpha : \alpha : \langle \Gamma; \Delta \rangle$, then $[P]_\alpha \approx ([P]_\alpha \mid \mathrm{M})$ with $\mathrm{M} = \alpha : \langle \Gamma; \Delta \rangle$.

- unmonitored correct processes are undistinguishable from their monitored counterparts.
- allows one to mix monitored and typechecked processes.

Global Transparency

Assume N and N have the same global transport $\langle r ; h \rangle$.
Assume:

1. N is fully monitored w.r.t. Σ and
2. $N = M \mid \langle r ; h \rangle$ is unmonitored but $\models M : \Sigma$.

We have $\mathrm{N} \cong N$.

- monitors does not alterate behaviors of correct networks.
- monitor actions are not observable on correct components.

Results (Fidelity)

- a configuration is consistent: when it corresponds to a well-formed array of global types $(G_1, \ldots, G_n)$ through projection.
- conformance is satisfaction + receivability (queue can be emptied).

Session Fidelity

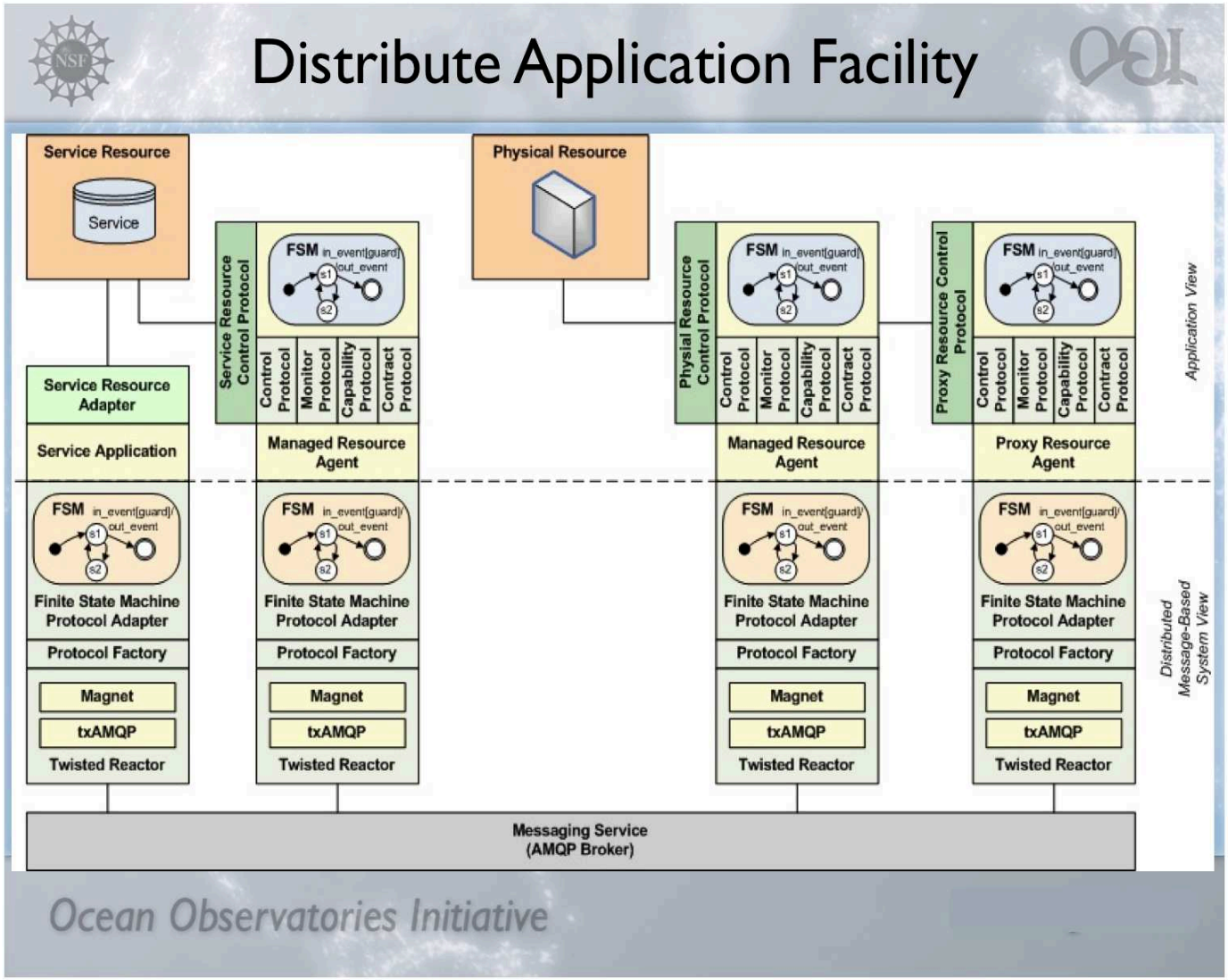
Assume:

1. configuration $\Sigma; \langle r \ ; \ h \rangle$ is consistent,
2. network $N \equiv M | \langle r \ ; \ h \rangle$ conforms to configuration $\Sigma; \langle r \ ; \ h \rangle$.

For any ℓ, whenever we have $N \xrightarrow{\ell}_{\mathrm{g}} N'$ s.t. $\Sigma; \langle r \ ; \ h \rangle \xrightarrow{\ell}_{\mathrm{g}} \Sigma'; \langle r' \ ; \ h' \rangle$, it holds that $\Sigma'; \langle r' \ ; \ h' \rangle$ is consistent and N' conforms to $\Sigma'; \langle r' \ ; \ h' \rangle$.

- consistence is preserved by reduction,
- at any time, the network correspond to a well-formed specification.

Distribute Application Facility



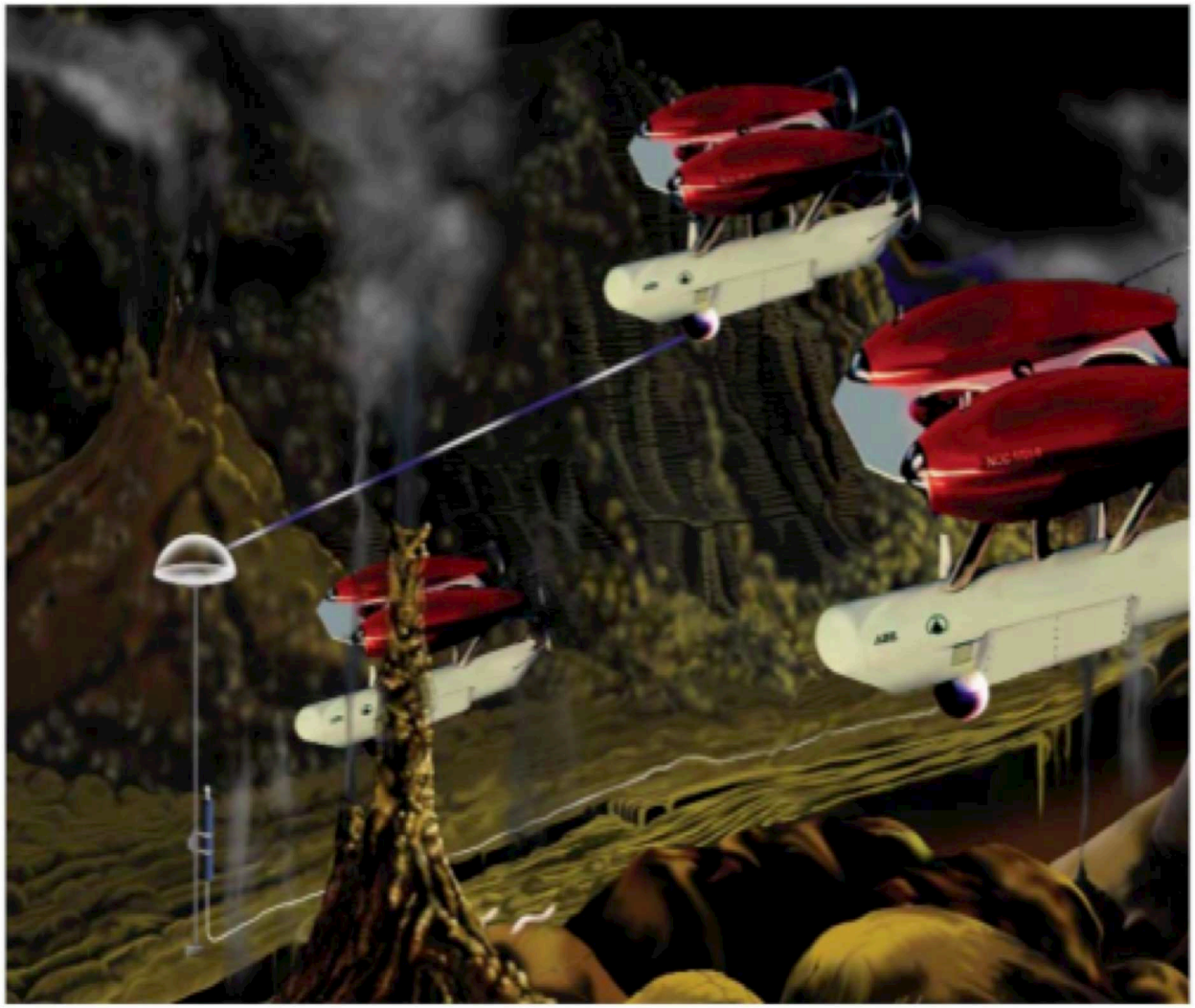


Figure 5: A coordinated set of autonomous underwater vehicles

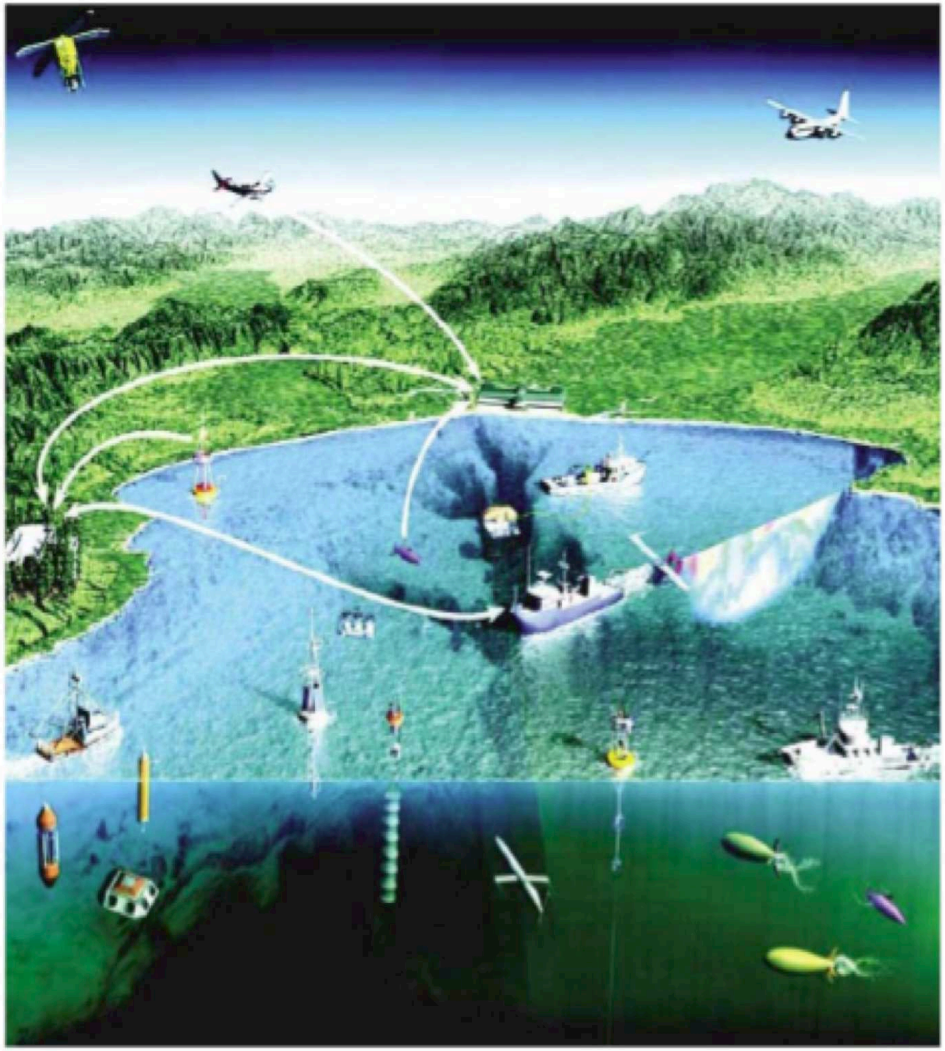


Figure 3: Observatory comprised of ships, aircraft and autonomous vehicles linked to assimilation modeling capabilities on shore

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.

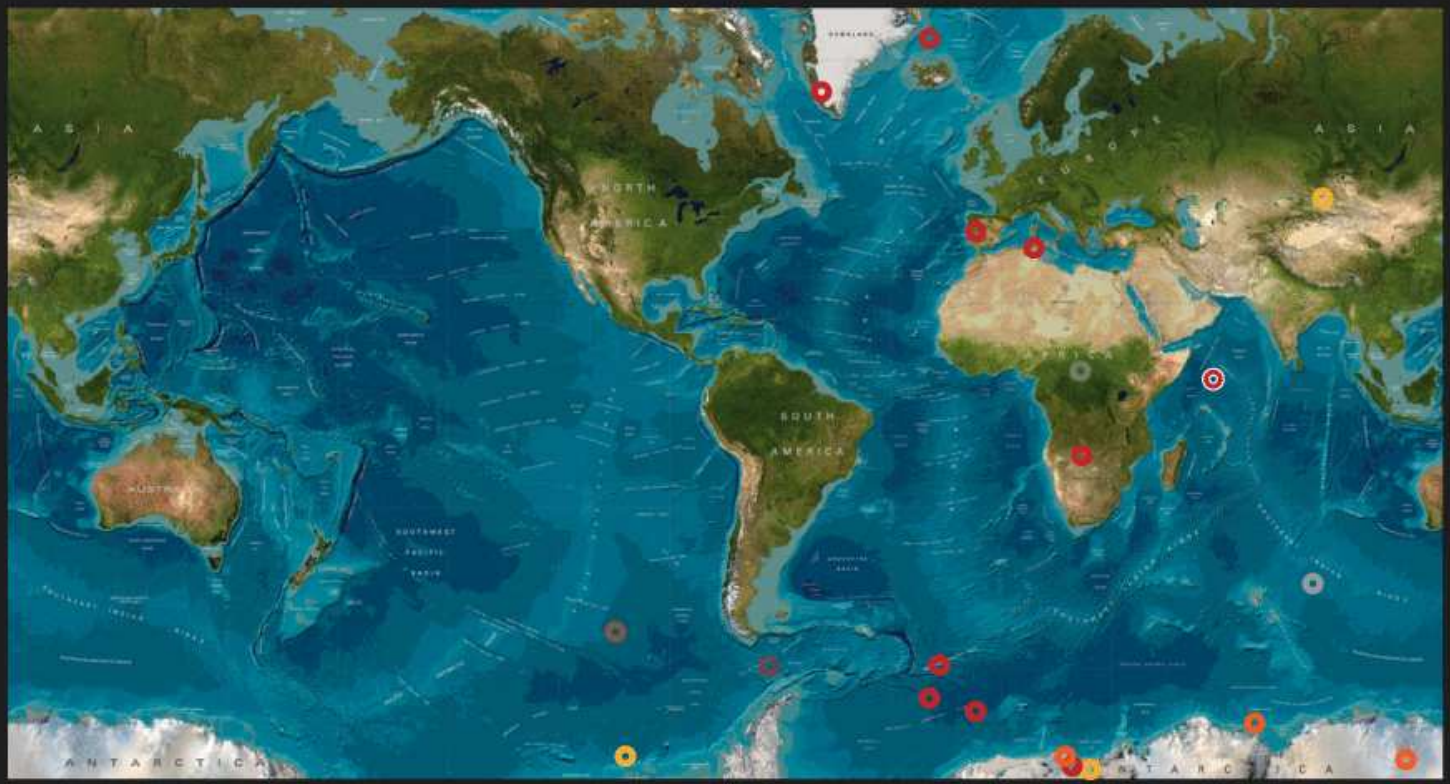
National Science Foundation working with *Consortium for Ocean Leadership*

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

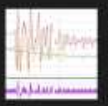
POPULAR RESOURCES

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

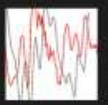
Marine caption
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

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

Multiparty Session Type Theory

- Multiparty Asynchronous Session Types [POPL'08]
- Progress
  - Global Progress in Dynamically Interleaved Multiparty Sessions [CONCUR'08], [Math. Struct. Comp. Sci.]
  - Inference of Progress Typing [Coordination'13]
- Asynchronous Optimisations and Resource Analysis
  - Global Principal Typing in Partially Commutative Asynchronous Sessions [ESOP'09]
  - Higher-Order Pi-Calculus [TLCA'07,TLCA'09]
  - Buffered Communication Analysis in Distributed Multiparty Sessions [CONCUR'10]

- Logics
  - Design-by-Contract for Distributed Multiparty Interactions [CONCUR'10]
  - Specifying Stateful Asynchronous Properties for Distributed Programs [CONCUR'12]
  - Multiparty, Multi-session Logic [TGC'12]
- Extensions of Multiparty Session Types
  - Multiparty Symmetric Sum Types [Express'10]
  - Parameterised Multiparty Session Types [FoSSaCs'10, LMCS]
  - Global Escape in Multiparty Sessions [FSTTCS'10] [Math. Struct. Comp. Sci.]
  - Dynamic Multirole Session Types [POPL'11]
  - Nested Multiparty Sessions [CONCUR'12]

- Dynamic Monitoring
  - Asynchronous Distributed Monitoring for Multiparty Session Enforcement [TGC'11]
  - Monitoring Networks through Multiparty Sessions [FORTE'13]
- Automata Theories
  - Multiparty Session Automata [ESOP'12]
  - Synthesis in Communicating Automata [ICALP'13]
- Typed Behavioural Theories
  - On Asynchronous Eventful Session Semantics [FORTE'11] [Math. Struct. Comp. Sci.]
  - Governed Session Semantics [CONCUR'13]
- Choreography Languages
  - Compositional Choreographies [CONCUR'13]

Language and Implementations

- Carrying out large-scale experiences with OOI, Pivotal, Red Hat, Congnizant, UNIFI, TrustCare
  - JBoss SCRIBBLE [ICDCIT'10, COB'12] and SAVARA projects
- High-performance computing

  Session Java [ECOOP'08,ECOOP'10,Coordination'11]

  $\Longrightarrow$ Multiparty Session C [TOOLS'12][Hearts'12][EuroMPI'12][PDP'14]
- Multiparty session languages Ocaml, Java, C, Python, Scala, Jolie
  - Trustworthy Pervasive Healthcare Services via Multiparty Session Types [FHIES'12]
  - SPY: Local Verification of Global Protocols [RV'13]
  - Practical interruptible conversations: Distributed dynamic verification with session types and Python [RV'13]

Session Type Projects

- COST Action *Behavioural Types for Reliable Large-Scale Software Systems*, over 60 academic members in 17 countries
- SADEA EPSRC *Exploiting Parallelism through Type Transformations for Hybrid Manycore Systems*, with Vanderbauwhede, Scholz, Gay and Luk
- Programme Grant *From Data Types to Session Types: A Basis for Concurrency and Distribution*, with Wadler and Gay
- EPSRC *Conversation-Based Governance for Distributed Systems by Multiparty Session Types*
- NSF Ocean Observatories Initiative
- Pivotal Dynamic Assurance based on Multiparty Session Types
- Cognizant/Qualit-e EPSRC Knowledge Transfer Secondments

Session Type Reading List

- [ESOP'98] Honda, Vasconcelos and Kubo, Language Primitives and Type Disciplines for Structured Communication-based Programming,
- [SecRet'06] Yoshida and Vasconcelos, Language Primitives and Type Disciplines for Structured Communication-based Programming Revisited, ENTCS.
- [ECOOP'08] Hu, Yoshida and Honda, Session-Based Distributed Programming in Java
- [POPL'08] Carbone, Yoshida and Honda, Multiparty Asynchronous Session Types
- [WS-FM'09] Dezani-Ciancaglini and de'Liguoro, Sessions and Session Types
- [TOOLS'12] Ng, Yoshida and Honda, Multiparty Session C
- [CONCUR'10] Caires and Pfenning, Session Types as Intuitionistic Linear Propositions; [ICFP'12] Walker, as Classical Linear Propositions.
- [OOI] Video by John Orcutt, Professor of Geophysics, UCSD, Ocean Observing: Oceanography in the 21st Century

A rare cluster of qualities

From the team of OOI CI:

Kohei has lead us deep into the nature of communication and processing. His esthetics, precision and enthusiasm for our mutual pursuit of formal Session (Conversation) Types and specifically for our OOI collaboration to realize this vision in very concrete terms were, as penned by Henry James, lessons in seeing the nuances of both beauty and craft, through a rare cluster of qualities - curiosity, patience and perception; all at the perfect pitch of passion and expression.