Let's Scribble

Rumyana Neykova, Imperial College London

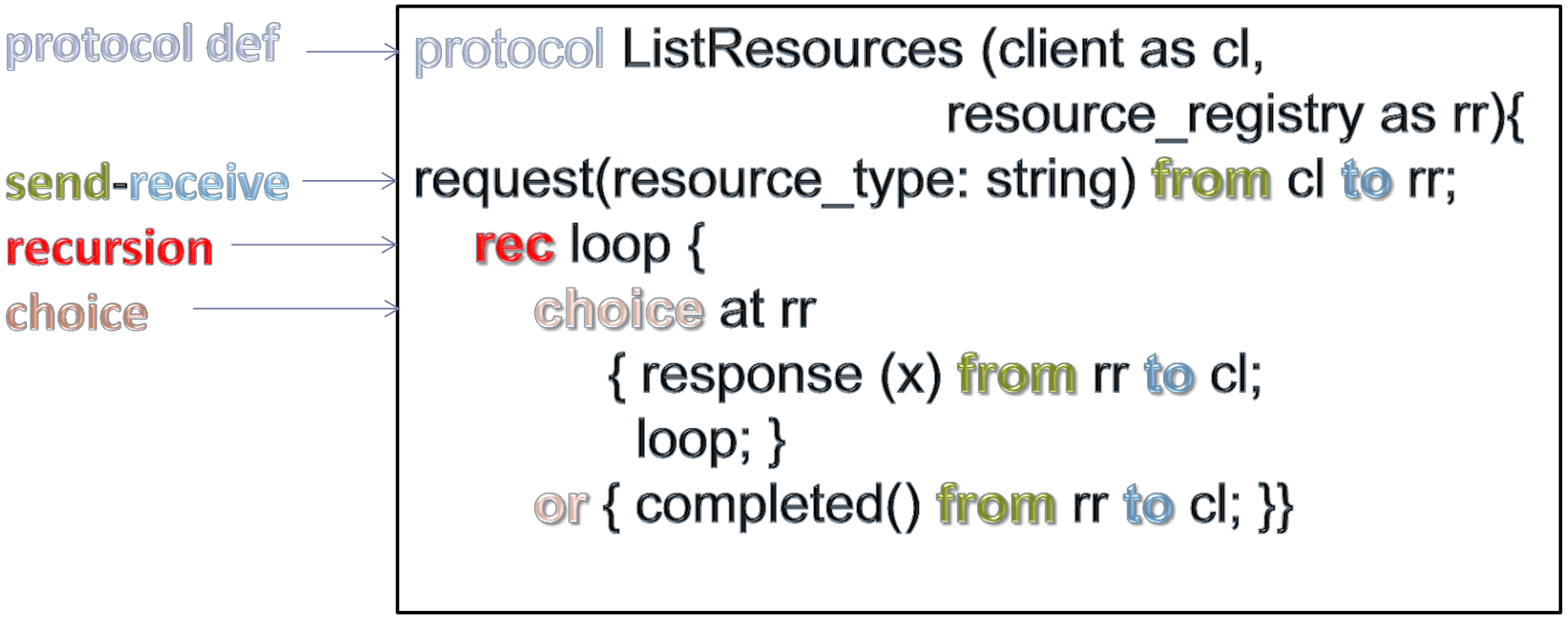
K.Honda, N.Yoshida,
Raymond Hu, Pierre-Malo Deni'elou, Romain Demangeon, Tzu-Chen Chun

Let's Scribble

```
protocol Hello(me, you)
{
    Hello from me to you;
    Hello from you to me;
}
```

Scribble is a language to describe application-level protocols among communicating systems. A protocol represents an agreement on how participating systems interact with each other.

Scribble: basic construct



Scribble: More constructs

interrupt →
parallel →
do →

```
import ListingFormat.xsd as lf;
protocol ListMultipleResources (client as cl,
                                resource_registry as rr1,
                                resource_registry as rr2){
Interruptible {
    parallel {
        do ListResource(cl, rr1)}
    and{
        do ListResource(cl, rr2)}
} by cl with  timeout();}}}
```

Tentative constructs

Logic assertions = Scribble Construct @{Logic}

```
...
request(resource: string) from cl to rr
        @{resource in available_resources};
...
```

Subsessions = spawn Scribble Protocol

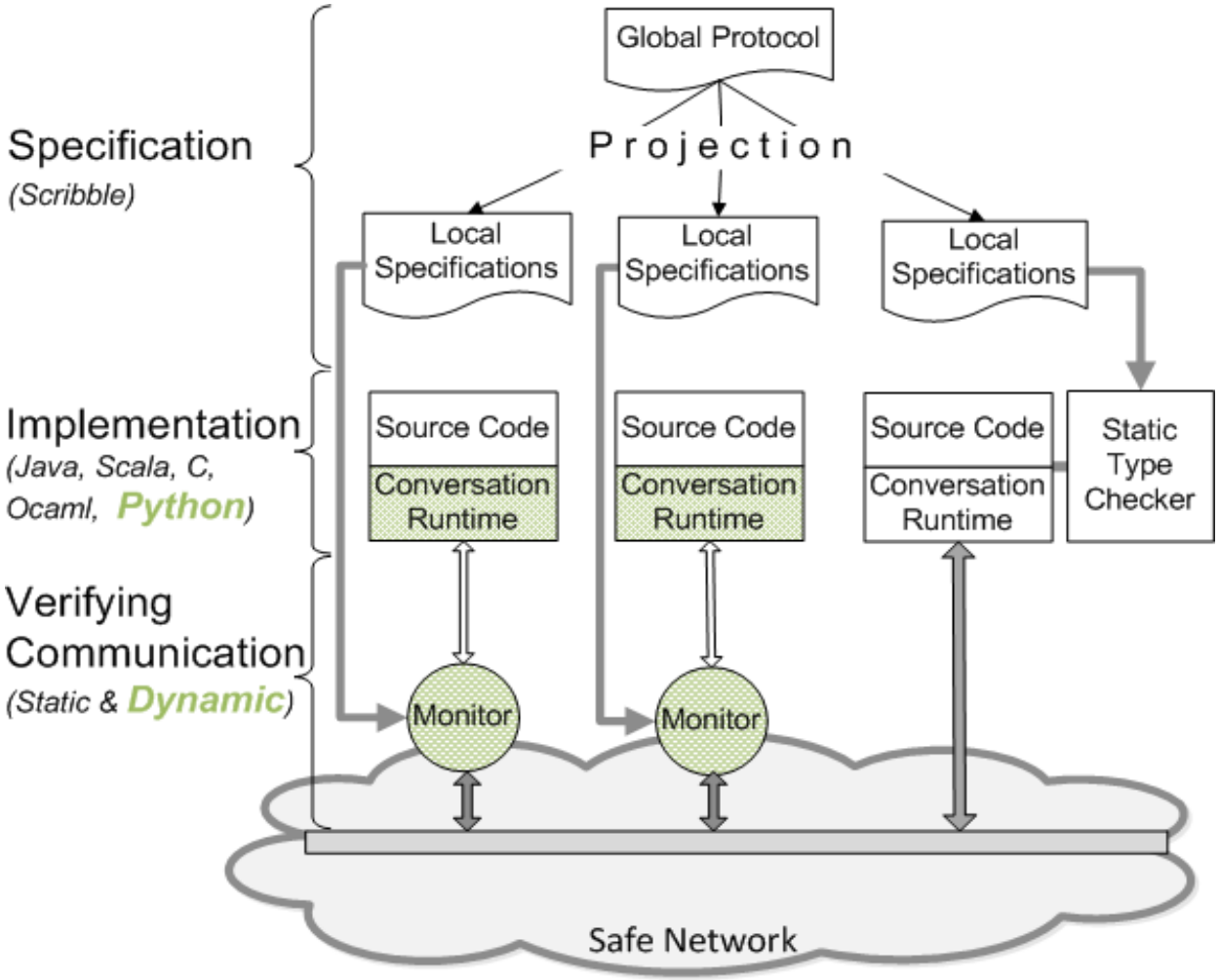
```
...
spawn ListResource(cl, rr ) at rr;
 ...
```

Interrupts – anywhere in the protocol

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

Applications: hybrid framework



How it can be used?

How to use Scribble

Global Conversation (In Scribble)

```
protocol ListResources(client, registry){
  Request(resource_kind) from client to registry;
  Response from registry to client; }
```

Projection

Local Type (In Scribble)

```
protocol ListResources at registry{
  Request(resource_kind) from client;
  Response to client;}
```

```
protocol ListResources at client{
  Request(resource_kind) to register;
  Response from register; }
```

Verification

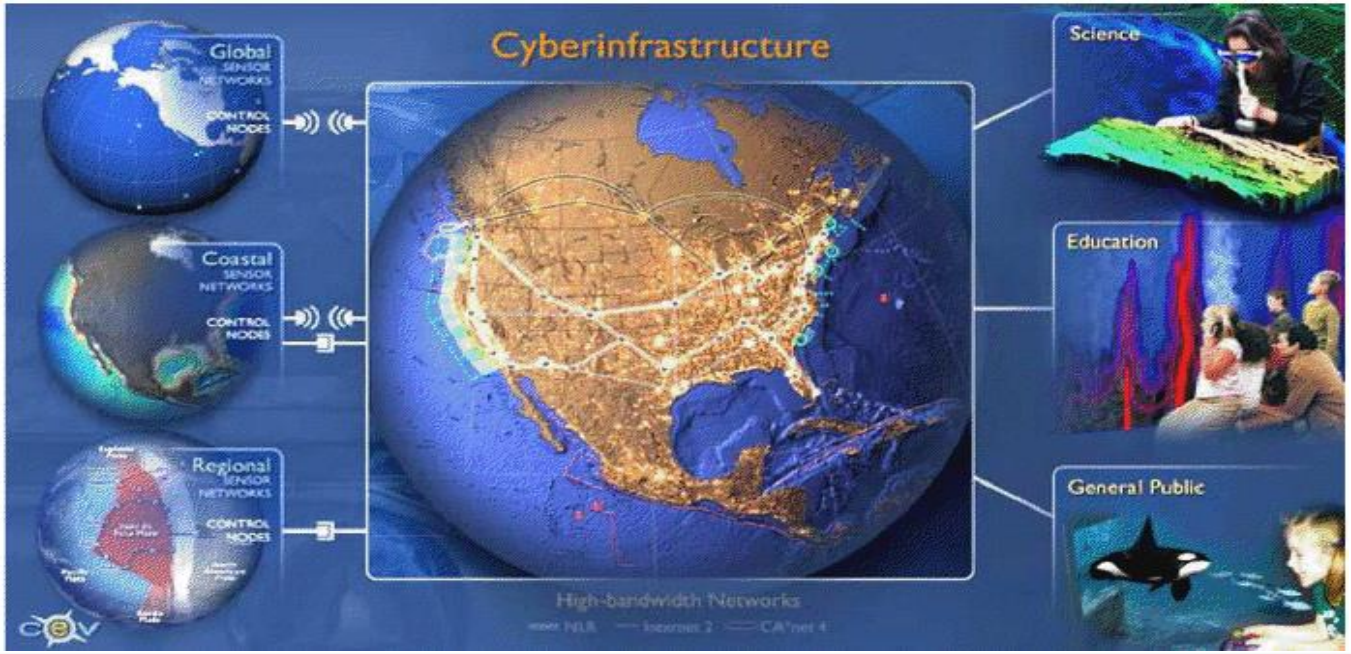
Programs (In Python)

```
p = Participant(name)
c = p.accept_invitation('ListResources',
                        'registry')
resource_kind = c.recv('client')
resource = get_resources(resource_kind)
c.send('client', resource)
```

```
p = Participant(name)
c =
p.create_conversation('ListResources',
                      'client')
msg = 'Resource:asdf'
c.send('registry', msg)
resource = c.recv('registry')
```

Case Study: OOI

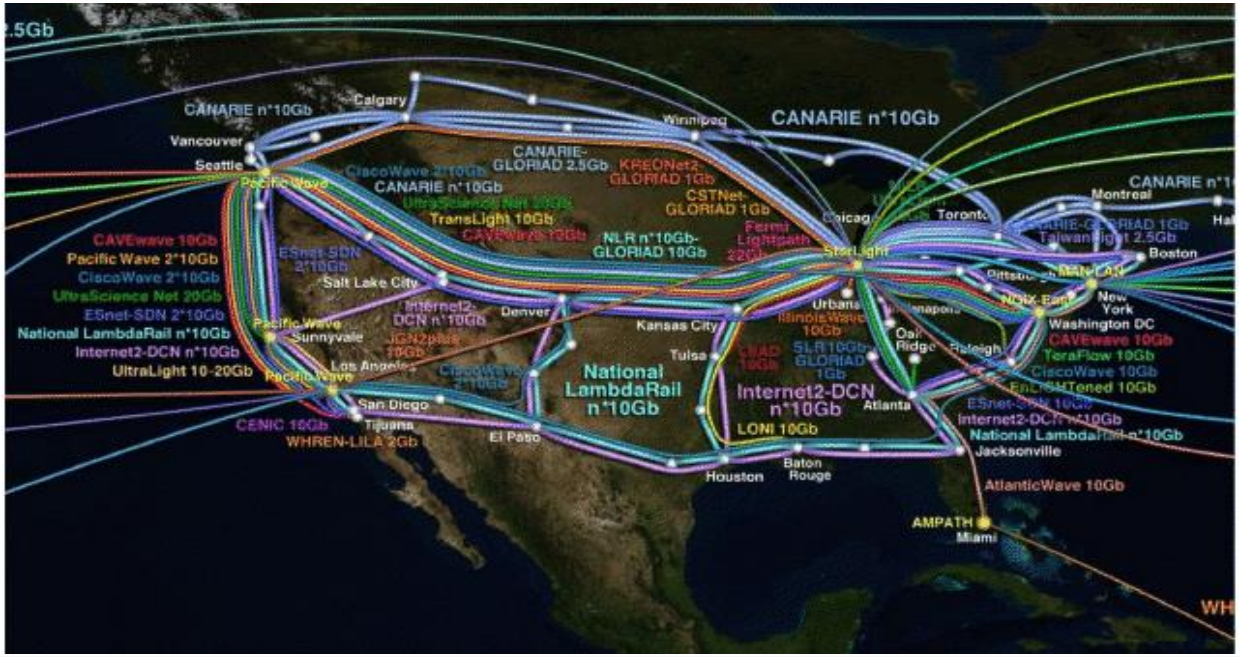
OOI aims: to deploy an infrastructure (global network) to expand the scientists' ability to remotely study the ocean



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

OOI Collaboration

Collaboration: build conversation layer on top of OOI network to guarantee global safety



Challenge: It is all RPC ☹, how and why to transform it into conversation => Conversation over RPC Project

Future work, interesting directions

- Governance on top of Scribble
  - Security, Policy
- Language
  - Generic protocols
  - flexible conversation flow (interruptible)
- How to increase adoption? Applications?
  - Tools for Scribble (plugin supports)
  - Specification of API, VM configuration/communication
- Imports and checks on the communicates data
  - Defining primary\base types
  - use of a common message formats such as yml
  - check integrity constraints, marshaling

It is your turn …

```
protocol Q&A(you, me)
{
    rec Loop
    {
        Questions from you to me;
        Answers from me to you;
        Loop;
    }
}
```