

Multiparty protocol specification and endpoint implementation using Scribble

Raymond Hu

Imperial College London

<http://www.doc.ic.ac.uk/~rhu/betty16a.pdf>

Aims

- ▶ Scribble
 - ▶ Implementation and application of MPST to current practices
 - ▶ Specify real-world protocols
 - ▶ Implement fully interoperable endpoints in mainstream languages

Hello, world: HTTP (GET)

- ▶ Hypertext Transfer Protocol
 - ▶ HTTP/1.1 RFCs 7230–7235 [\[HTTP\]](#)
 - ▶ Client-server request-response “methods”
 - ▶ <https://tools.ietf.org/html/rfc7230#section-2.1>
 - ▶ (e.g. Web browser fetching a page from Web server)

[\[HTTP1.1\]](#) <https://tools.ietf.org/html/rfc7230>, ...

► Protocol specification = messages + interactions

- <https://github.com/rhu1/scribble-java/tree/rhu1-research/modules/core/src/test/scrib/demo/betty16/lec1/httpshort>

```
// Message types
```

```
sig <java> "demo.betty16.lec1.httpshort.message.client.Request"  
  from "demo/betty16/httpshort/message/Request.java"  
  as Request;
```

```
sig <java> "demo.betty16.lec1.httpshort.message.server.Response"  
  from "demo/betty16/shortvers/message/Response.java"  
  as Response;
```

```
global protocol Http(role C, role S) {  
  // Interaction structure  
  Request from C to S;  
  Response from S to C;  
}
```

Client implementation in Java

- ▶ For now, assume a basic fluent (call-chaining) Java API over TCP sockets

```
String host = "www.doc.ic.ac.uk"; int port = 80;  
Buf<Response> buf = new Buf<>();
```

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf); // Received message read into buf
```

```
c.send(S, new Response("1.1", "..body.."))  
  .receive(S, Response, buf);
```

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf);
```

- ▶ ..so is that it? For a good implementation


Client implementation in Java

- ▶ For now, assume a basic fluent (call-chaining) Java API over TCP sockets

```
String host = "www.doc.ic.ac.uk"; int port = 80;  
Buf<Response> buf = new Buf<>();
```

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf); // Received message read into buf
```

```
c.send(S, new Response("1.1", "..body.."))  
  .receive(S, Response, buf);
```

 The method send(S, Request) ... for the arguments (S, Response)

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf);
```

- ▶ ..so is that it? For a good implementation

Client implementation in Java


- ▶ For now, assume a basic fluent (call-chaining) Java API over TCP sockets

```
String host = "www.doc.ic.ac.uk"; int port = 80;  
Buf<Response> buf = new Buf<>();
```

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf); // Received message read into buf
```

```
c.send(S, new Response("1.1", "..body.."))  
  .receive(S, Response, buf);
```

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf);
```

 The method send(S, Request) is undefined for the type Http_C_2

- ▶ ..so is that it? For a good implementation

Client implementation in Java

- ▶ For now, assume a basic fluent (call-chaining) Java API over TCP sockets

```
String host = "www.doc.ic.ac.uk"; int port = 80;  
Buf<Response> buf = new Buf<>();
```

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf); // Received message read into buf
```

```
c.send(S, new Response("1.1", "..body.."))  
  .receive(S, Response, buf);
```

```
c.send(S, new Request("/~rhu/", "1.1", host))  
  .send(S, new Request("/~rhu/", "1.1", host))  
  .receive(S, Response, buf);
```

- ▶ ..so is that it? For a good implementation

Message types vs. interaction structure

- ▶ Simple interaction structure..
 - ▶ ..means more work is done in message serialization/deserialization
 - ▶ <https://tools.ietf.org/html/rfc7230#section-3>
 - ▶ The call-response pattern and top-level data types are checked..
how about serialization/deserializatou?
- ▶ Specification interplay between data types and interaction structure
 - ▶ Can leverage session types to expose message formatting details

HTTP client-server conversation

▶ telnet www.doc.ic.ac.uk 80

```
GET /~rhu/ HTTP/1.1
```

```
Host: www.doc.ic.ac.uk
```

```
User-Agent: User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

```
Accept-Language: en-GB,en;q=0.5
```

```
Accept-Encoding: gzip, deflate
```

```
DNT: 1
```

```
Connection: keep-alive
```

```
:
```

HTTP client-server conversation

▶ telnet www.doc.ic.ac.uk 80

:

HTTP/1.1 200 OK

Date: Mon, 13 Jun 2016 19:42:34 GMT

Server: Apache

Strict-Transport-Security: max-age=31536000; preload; includeSubDomains

Strict-Transport-Security: max-age=31536000; preload; includeSubDomains

Last-Modified: Thu, 14 Apr 2016 12:46:24 GMT

ETag: "74a-53071482f6e0f"

Accept-Ranges: bytes

Content-Length: 1866

Vary: Accept-Encoding

Content-Type: text/html

Via: 1.1 www.doc.ic.ac.uk

Decomposing message structures..

- ▶ <https://github.com/rhu1/scribble-java/tree/rhu1-research/modules/core/src/test/scrib/demo/betty16/lec1/httpplong>

- ▶ Client messages

```
sig <java> "...message.client.RequestLine" from "...message/RequestLine.java"
  as RequestLine; // GET /~rhu/ HTTP/1.1
sig <java> "...message.client.Host" from "...message/Host.java"
  as Host; // host: www.doc.ic.ac.uk
sig <java> "...message.client.UserAgent" from "...message/UserAgent.java"
  as UserAgent; // User-Agent: Mozilla/5.0 ... Firefox/38.0
...
```

- ▶ Server messages

```
sig <java> "...message.server.HttpVersion" from "...message/HttpVersion.java"
  as HTTPV; // HTTP/1.1
sig <java> "...message.server._200" from "...message/_200.java"
  as 200; // 200 OK
sig <java> "...message.server._404" from "...message/_404.java"
  as 404; // 404 Not found
...
```

..promotes more fine-grained interaction structures

```
global protocol Http(role C, role S) {
  do Request(C, S);
  do Response(C, S);
}

global protocol Request(role C, role S) {
  RequestLine from C to S;  // GET /~rhu/ HTTP/1.1
  rec X {
    choice at C {
      Host from C to S;      // Host: www.doc.ic.ac.uk
      continue X;
    } or {
      UserAgent from C to S; // User-Agent: Mozilla/5.0 ...
      continue X;
    } or {
      ...
    } or {
      Body from C to S;
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Http(role C, role S) {
  do Request(C, S);
  do Response(C, S);
}

global protocol Request(role C, role S) {
  RequestLine from C to S; // GET /~rhu/ HTTP/1.1
  rec X {
    choice at C {
      Host from C to S; // Host: www.doc.ic.ac.uk
      continue X;
    } or {
      UserAgent from C to S; // User-Agent: Mozilla/5.0 ...
      continue X;
    } or {
      ...
    } or {
      Body from C to S;
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Http(role C, role S) {
  do Request(C, S);
  do Response(C, S);
}

global protocol Request(role C, role S) {
  RequestLine from C to S; // GET /~rhu/ HTTP/1.1
  rec X {
    choice at C {
      Host from C to S; // Host: www.doc.ic.ac.uk
      continue X;
    } or {
      UserAgent from C to S; // User-Agent: Mozilla/5.0 ...
      continue X;
    } or {
      ...
    } or {
      Body from C to S;
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Http(role C, role S) {
  do Request(C, S);
  do Response(C, S);
}

global protocol Request(role C, role S) {
  RequestLine from C to S; // GET /~rhu/ HTTP/1.1
  rec X {
    choice at C {
      Host from C to S; // Host: www.doc.ic.ac.uk
      continue X;
    } or {
      UserAgent from C to S; // User-Agent: Mozilla/5.0 ...
      continue X;
    } or {
      ...
    } or {
      Body from C to S;
    }
  }
}
```


..promotes more fine-grained interaction structures

```
global protocol Http(role C, role S) {
  do Request(C, S);
  do Response(C, S);
}

global protocol Request(role C, role S) {
  RequestLine from C to S; // GET /~rhu/ HTTP/1.1
  rec X {
    choice at C {
      Host from C to S; // Host: www.doc.ic.ac.uk
      continue X;
    } or {
      UserAgent from C to S; // User-Agent: Mozilla/5.0 ...
      continue X;
    } or {
      ...
    } or {
      Body from C to S;
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Http(role C, role S) {
  do Request(C, S);
  do Response(C, S);
}

global protocol Request(role C, role S) {
  RequestLine from C to S; // GET /~rhu/ HTTP/1.1
  rec X {
    choice at C {
      Host from C to S; // Host: www.doc.ic.ac.uk
      continue X;
    } or {
      UserAgent from C to S; // User-Agent: Mozilla/5.0 ...
      continue X;
    } or {
      ...
    } or {
      Body from C to S;
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Http(role C, role S) {
  do Request(C, S);
  do Response(C, S);
}

global protocol Request(role C, role S) {
  RequestLine from C to S; // GET /~rhu/ HTTP/1.1
  rec X {
    choice at C {
      Host from C to S; // Host: www.doc.ic.ac.uk
      continue X;
    } or {
      UserAgent from C to S; // User-Agent: Mozilla/5.0 ...
      continue X;
    } or {
      ...
    } or {
      Body from C to S;
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Reponse(role C, role S) {
  HttpVers from S to C; // HTTP/1.1
  choice at S {
    200 from S to C; // 200 OK
  } or {
    404 from S to C; // 404 Not found
  } or {
    ...
  }

  rec Y {
    choice at S {
      Date from S to C; // Date: Sun, 24 May 2015 21:04:36 GMT
      continue Y;
    } or {
      Server from S to C; // Server: Apache
      continue Y;
    } or {
      ...
    } or {
      Body from S to C; // <html>...</html>
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Reponse(role C, role S) {
  HttpVers from S to C; // HTTP/1.1
  choice at S {
    200 from S to C; // 200 OK
  } or {
    404 from S to C; // 404 Not found
  } or {
    ...
  }

  rec Y {
    choice at S {
      Date from S to C; // Date: Sun, 24 May 2015 21:04:36 GMT
      continue Y;
    } or {
      Server from S to C; // Server: Apache
      continue Y;
    } or {
      ...
    } or {
      Body from S to C; // <html>...</html>
    }
  }
}
```

..promotes more fine-grained interaction structures

```
global protocol Reponse(role C, role S) {
  HttpVers from S to C; // HTTP/1.1
  choice at S {
    200 from S to C; // 200 OK
  } or {
    404 from S to C; // 404 Not found
  } or {
    ...
  }

  rec Y {
    choice at S {
      Date from S to C; // Date: Sun, 24 May 2015 21:04:36 GMT
      continue Y;
    } or {
      Server from S to C; // Server: Apache
      continue Y;
    } or {
      ...
    } or {
      Body from S to C; // <html>...</html>
    }
  }
}
```

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
Http_C_3 request(Http_C_1 c1, String host) throws ... {  
    return  
        c1.send(S, new RequestLine("/~rhu/", "1.1"))  
            .send(S, new Host(host))  
            .send(S, new Body(""));  
}
```

- ✓ Formatting of request message (request line, fields) is now checked

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
Http_C_3 request(Http_C_1 c1, String host) throws ... {  
    return  
        c1.send(S, new RequestLine("/~rhu/", "1.1"))  
            .send(S, new Host(host))  
            .send(S, new Body(""));  
}
```

- ✓ Formatting of request message (request line, fields) is now checked

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
Http_C_3 request(Http_C_1 c1, String host) throws ... {  
    return  
        c1.send(S, new RequestLine("/~rhu/", "1.1"))  
          .send(S, new Host(host))  
          .send(S, new Body(""));  
}
```

- ✓ Formatting of request message (request line, fields) is now checked

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
void response(Http_C_3 c3) throws ... {  
    Http_C_4_Cases status = c3.async(S, HTTPV).branch(S);  
    switch (status.op) {  
        case _200: responseAux(status.receive(_200)); break;  
        case _404: responseAux(status.receive(_404)); break;  
        default: throw new RuntimeException("[TODO]: " + status.op);  
    } }  
}
```

```
void responseAux(Http_C_5 c5) throws ... {  
    Http_C_5_Cases cases = c5.branch(S);  
    switch (cases.op) {  
        case DATE: responseAux(cases.receive(DATE)); break;  
        case SERVER: responseAux(cases.receive(SERVER)); break;  
        ...  
        case BODY: { Buf<Body> buf_body = new Buf<>();  
                    cases.receive(BODY, buf_body);  
                    System.out.println(buf_body.val.getBody());  
                    return; }  
        default: throw new RuntimeException("[TODO]: " + cases.op);  
    } }  
}
```

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
void response(Http_C_3 c3) throws ... {
    Http_C_4_Cases status = c3.async(S, HTTPV).branch(S);
    switch (status.op) {
        case _200: responseAux(status.receive(_200)); break;
        case _404: responseAux(status.receive(_404)); break;
        default: throw new RuntimeException("[TODO]: " + status.op);
    } }
}
```

```
void responseAux(Http_C_5 c5) throws ... {
    Http_C_5_Cases cases = c5.branch(S);
    switch (cases.op) {
        case DATE: responseAux(cases.receive(DATE)); break;
        case SERVER: responseAux(cases.receive(SERVER)); break;
        ...
        case BODY: { Buf<Body> buf_body = new Buf<>();
                    cases.receive(BODY, buf_body);
                    System.out.println(buf_body.val.getBody());
                    return; }
        default: throw new RuntimeException("[TODO]: " + cases.op);
    } }
}
```

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
void response(Http_C_3 c3) throws ... {
    Http_C_4_Cases status = c3.async(S, HTTPV).branch(S);
    switch (status.op) {
        case _200: responseAux(status.receive(_200)); break;
        case _404: responseAux(status.receive(_404)); break;
        default: throw new RuntimeException("[TODO]: " + status.op);
    } }
}
```

```
void responseAux(Http_C_5 c5) throws ... {
    Http_C_5_Cases cases = c5.branch(S);
    switch (cases.op) {
        case DATE: responseAux(cases.receive(DATE)); break;
        case SERVER: responseAux(cases.receive(SERVER)); break;
        ...
        case BODY: { Buf<Body> buf_body = new Buf<>();
                    cases.receive(BODY, buf_body);
                    System.out.println(buf_body.val.getBody());
                    return; }
        default: throw new RuntimeException("[TODO]: " + cases.op);
    } }
}
```

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
void response(Http_C_3 c3) throws ... {
    Http_C_4_Cases status = c3.async(S, HTTPV).branch(S);
    switch (status.op) {
        case _200: responseAux(status.receive(_200)); break;
        case _404: responseAux(status.receive(_404)); break;
        default: throw new RuntimeException("[TODO]: " + status.op);
    } }
}
```

```
void responseAux(Http_C_5 c5) throws ... {
    Http_C_5_Cases cases = c5.branch(S);
    switch (cases.op) {
        case DATE: responseAux(cases.receive(DATE)); break;
        case SERVER: responseAux(cases.receive(SERVER)); break;
        ...
        case BODY: { Buf<Body> buf_body = new Buf<>();
                    cases.receive(BODY, buf_body);
                    System.out.println(buf_body.val.getBody());
                    return; }
        default: throw new RuntimeException("[TODO]: " + cases.op);
    } }
}
```

Revised client code

```
response(request(new Http_C_1(client), "www.doc.ic.ac.uk"));
```

```
void response(Http_C_3 c3) throws ... {  
    Http_C_4_Cases status = c3.async(S, HTTPV).branch(S);  
    switch (status.op) {  
        case _200: responseAux(status.receive(_200)); break;  
        case _404: responseAux(status.receive(_404)); break;  
        default: throw new RuntimeException("[TODO]: " + status.op);  
    } }  
}
```

```
void responseAux(Http_C_5 c5) throws ... {  
    Http_C_5_Cases cases = c5.branch(S);  
    switch (cases.op) {  
        case DATE: responseAux(cases.receive(DATE)); break;  
        case SERVER: responseAux(cases.receive(SERVER)); break;  
        ...  
        case BODY: { Buf<Body> buf_body = new Buf<>();  
                    cases.receive(BODY, buf_body);  
                    System.out.println(buf_body.val.getBody());  
                    return; }  
        default: throw new RuntimeException("[TODO]: " + cases.op);  
    } }  
}
```

Hello, world: HTTP (GET)

- ▶ Rigorous specification and verification of protocols is important (Even for a “simple” binary call-return)
- ▶ Further alternative specifications?
 - ▶ Most simplified:
 - ▶ Most detailed:
- ▶ Similarly for the server
 - ▶ All versions interoperable with (compliant) real-world implementations
 - ▶ And with each other

Hello, world: HTTP (GET)

- ▶ Rigorous specification and verification of protocols is important (Even for a “simple” binary call-return)
- ▶ Further alternative specifications?
 - ▶ Most simplified: call-return of ASCII strings
 - ▶ Most detailed:
- ▶ Similarly for the server
 - ▶ All versions interoperable with (compliant) real-world implementations
 - ▶ And with each other

Hello, world: HTTP (GET)

- ▶ Rigorous specification and verification of protocols is important (Even for a “simple” binary call-return)
- ▶ Further alternative specifications?
 - ▶ Most simplified: call-return of ASCII strings
 - ▶ Most detailed: towards “character-perfect” I/O?
- ▶ Similarly for the server
 - ▶ All versions interoperable with (compliant) real-world implementations
 - ▶ And with each other

Outline

- ▶ Scribble toolchain implementation of MPST
 - ▶ Specify and check global protocol
 - ▶ Check endpoint implementations follow their role in the protocol

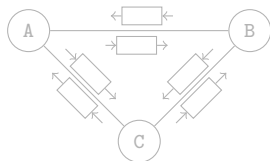
- ▶ Remainder of this session
 - ▶ Overview of the Scribble toolchain
 - ▶ Illustration of correspondence between MPST and communicating FSMs
 - ▶ Good and bad asynchronous multiparty protocols by example

- ▶ Next session
 - ▶ Session programming in Java
 - ▶ Hybrid session verification by Endpoint API generation

- ▶ (Implementation of distributed session delegation and asynchronous interrupt messages)

Scribble

- ▶ Adapts and extends formal MPST as a practical language for explicit specification of multiparty message passing protocols
 - ▶ Type syntax close to [MSCS15] Coppo, Dezani-Ciancaglini, Yoshida and Padovani
 - ▶ Key features build on correspondence to communicating FSM [ESOP12] Deniérou, Yoshida
 - ▶ Communication model: asynchronous, reliable, role-to-role ordering

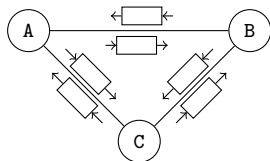


```
1() from A to B;  
2() from A to C;  
3() from C to B;
```

- ▶ Scribble applies to sessions conducted over transports that fit this model e.g. TCP, HTTP/TCP, ..., (AMQP), ..., shared memory, ...
- ▶ Scribble protocols should be fully explicit:
no implicit messages needed to conduct a session

Scribble

- ▶ Adapts and extends formal MPST as a practical language for explicit specification of multiparty message passing protocols
 - ▶ Type syntax close to [MSCS15] Coppo, Dezani-Ciancaglini, Yoshida and Padovani
 - ▶ Key features build on correspondence to communicating FSM [ESOP12] Deniélou, Yoshida
 - ▶ Communication model: asynchronous, reliable, role-to-role ordering

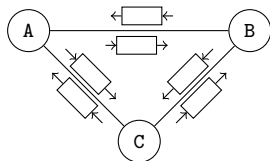


```
1() from A to B;  
2() from A to C;  
3() from C to B;
```

- ▶ Scribble applies to sessions conducted over transports that fit this model e.g. TCP, HTTP/TCP, ..., (AMQP), ..., shared memory, ...
- ▶ Scribble protocols should be fully explicit:
no implicit messages needed to conduct a session

Scribble

- ▶ Adapts and extends formal MPST as a practical language for explicit specification of multiparty message passing protocols
 - ▶ Type syntax close to [MSCS15] Coppo, Dezani-Ciancaglini, Yoshida and Padovani
 - ▶ Key features build on correspondence to communicating FSM [ESOP12] Deniérou, Yoshida
 - ▶ Communication model: asynchronous, reliable, role-to-role ordering

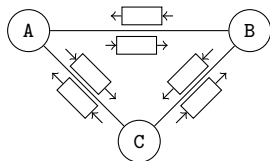


```
1() from A to B;  
2() from A to C;  
3() from C to B;
```

- ▶ Scribble applies to sessions conducted over transports that fit this model e.g. TCP, HTTP/TCP, ..., (AMQP), ..., shared memory, ...
- ▶ Scribble protocols should be fully explicit:
no implicit messages needed to conduct a session

Scribble

- ▶ Adapts and extends formal MPST as a practical language for explicit specification of multiparty message passing protocols
 - ▶ Type syntax close to [MSCS15] Coppo, Dezani-Ciancaglini, Yoshida and Padovani
 - ▶ Key features build on correspondence to communicating FSM [ESOP12] Deniérou, Yoshida
 - ▶ Communication model: asynchronous, reliable, role-to-role ordering



```
1() from A to B;  
2() from A to C;  
3() from C to B;
```

- ▶ Scribble applies to sessions conducted over transports that fit this model e.g. TCP, HTTP/TCP, ..., (AMQP), ..., shared memory, ...
- ▶ Scribble protocols should be fully explicit:
no implicit messages needed to conduct a session

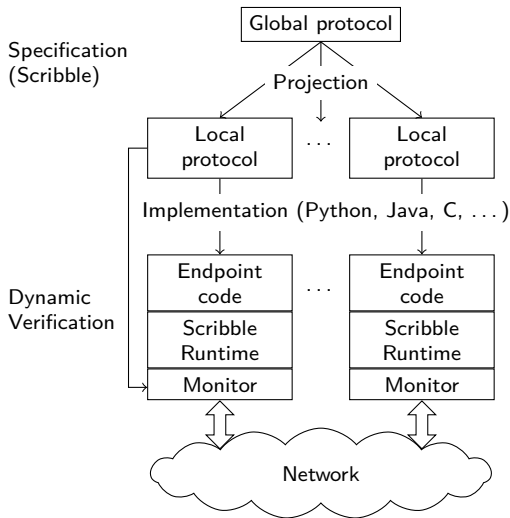
Scribble collaborations

- ▶ JBoss Savara (Red Hat): tool support for Testable Architecture
 - ▶ <http://www.jboss.org/savara>
 - ▶ Cognizant ZDLC: tools for governance and reverse engineering workflows
 - ▶ Uses Savara for internal modelling
 - ▶ <http://www.cognizantzdlc.com/>
- ▶ Ocean Observatories Initiative
 - ▶ Python-based endpoints on an AMQP-based network
 - ▶ <http://oceanobservatories.org/>
 - ▶ <https://confluence.oceanobservatories.org/display/syseng/CIAD+COI+TV+Conversations+and+Session+Types>
- ▶ Scribble resources
 - ▶ <http://www.scribble.org/>
(Some of the pre-built tools are based on older Scribble versions)
 - ▶ Master: <https://github.com/scribble/scribble-java>
 - ▶ Research: (used in these lectures; additional features but less stable)
<https://github.com/rhu1/scribble-java/tree/rhu1-research>

[FASE16] *Hybrid session verification through Endpoint API generation.* Hu and Yoshida.

[TGC13] *The Scribble Protocol Language.* Yoshida, Hu, Neykova and Ng.

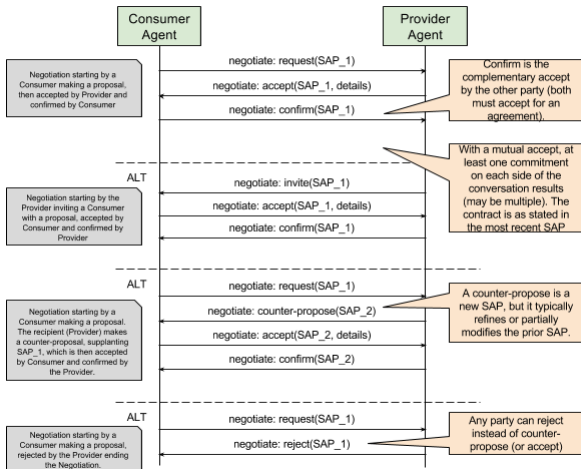
Scribble: MPST adapted for run-time monitoring



- ▶ Global protocol
 - ▶ Protocol validation
- ▶ Local protocols
 - ▶ FSM translation (endpoint monitor generation)
- ▶ (Heterogeneous) endpoint programs
 - ▶ Scribble session I/O API
 - ▶ (Interoperable) distributed session runtime

OOI Agent negotiation: user description

- ▶ <https://confluence.oceanobservatories.org/display/syseng/CIAD+COI+OV+Negotiate+Protocol>
- ▶ <https://github.com/rhu1/scribble-java/blob/rhu1-research/modules/core/src/test/scrib/demo/betty16/lec1/nego/Negotiate.scr>



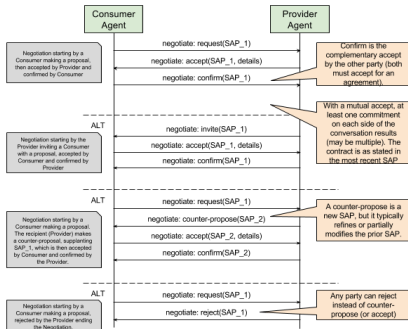
OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;  
  rec X {  
    choice at P {  
      acctpt() 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 {  
        acctpt() from C to P;  
        confirm() from P to C;  
      } or {  
        reject() from C to P;  
      } or {  
        propose(SAP) from C to P;  
        continue X;  
      }  
    }  
  }  
}
```

```
} } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

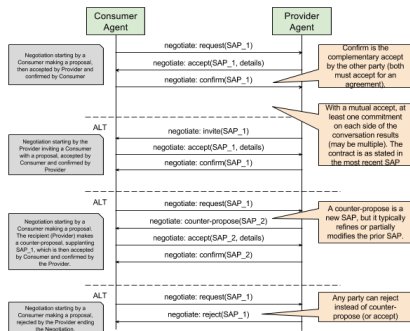
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
  } } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

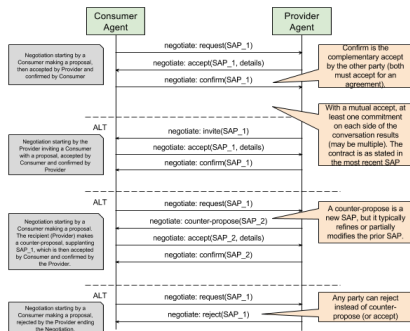
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
  } } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

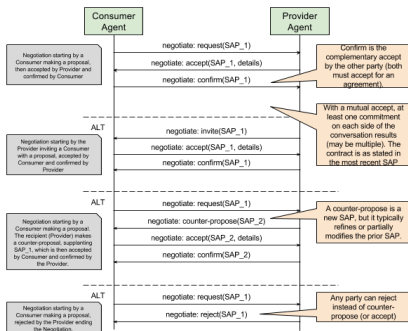
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
  } } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

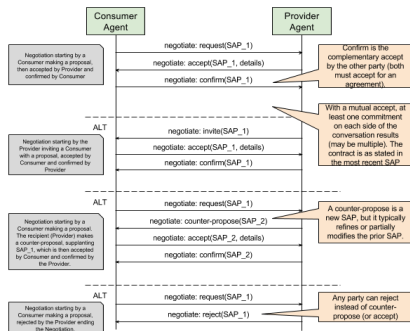
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
  } } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

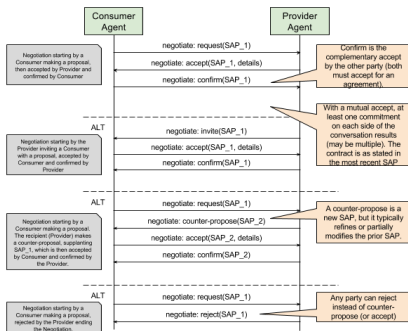
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
    } } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

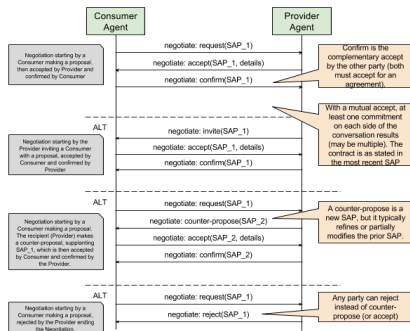
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
    } } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

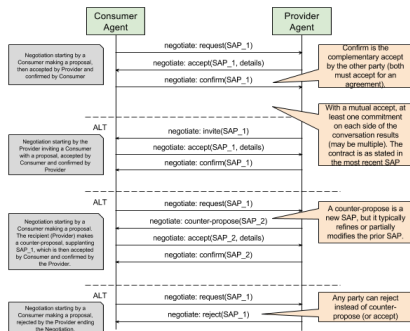
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
  } } } }
```



OOI Agent negotiation: global protocol

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

```
global protocol Negotiate(role C, role P) {
```

```
  propose(SAP) from C to P;
```

```
  rec X {
```

```
    choice at P {
```

```
      acctpt() 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 {
```

```
        acctpt() from C to P;
```

```
        confirm() from P to C;
```

```
      } or {
```

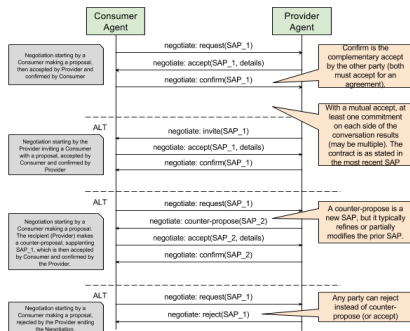
```
        reject() from C to P;
```

```
      } or {
```

```
        propose(SAP) from C to P;
```

```
        continue X;
```

```
    } } } }
```

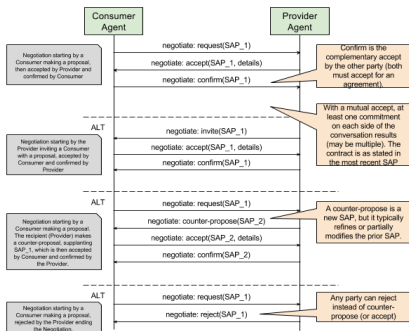


OOI Agent negotiation: local projection for C

```

propose(SAP) to P;
rec X {
  choice at P {
    acctpt() to C;
    confirm() to P;
  } or {
    reject() from P;
  } or {
    propose(SAP) from P;
    choice at C {
      acctpt() to P;
      confirm() from P;
    } or {
      reject() to P;
    } or {
      propose(SAP) to P;
      continue X;
    }
  } } }

```

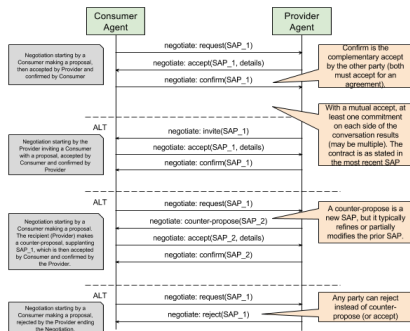


OOI Agent negotiation: local projection for C

```

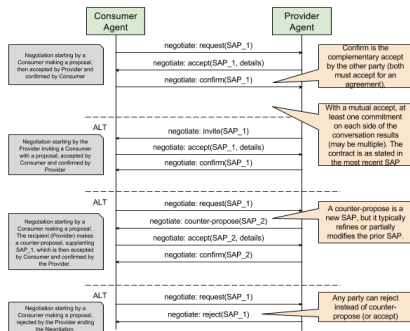
propose(SAP) to P;
rec X {
  choice at P {
    acppt() to C;
    confirm() to P;
  } or {
    reject() from P;
  } or {
    propose(SAP) from P;
    choice at C {
      acppt() to P;
      confirm() from P;
    } or {
      reject() to P;
    } or {
      propose(SAP) to P;
      continue X;
    } } }

```



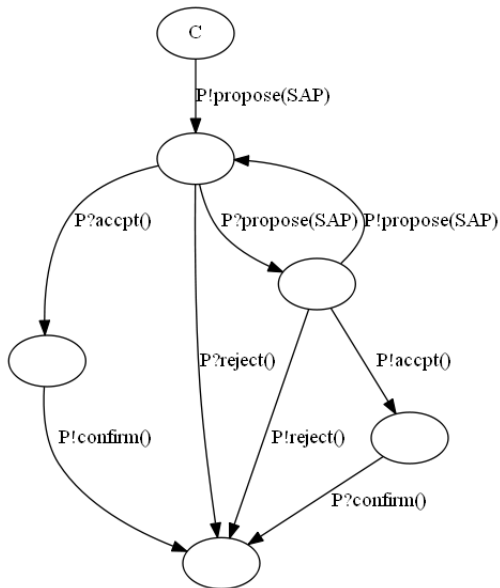
OOI Agent negotiation: local projection for C

```
propose(SAP) to P;  
rec X {  
  choice at P {  
    acctpt() to C;  
    confirm() to P;  
  } or {  
    reject() from P;  
  } or {  
    propose(SAP) from P;  
    choice at C {  
      acctpt() to P;  
      confirm() from P;  
    } or {  
      reject() to P;  
    } or {  
      propose(SAP) to P;  
      continue X;  
    } } }  
}
```



OOI Agent negotiation: FSM translation for C

```
propose(SAP) to P;
rec X {
  choice at P {
    acpt() to C;
    confirm() to P;
  } or {
    reject() from P;
  } or {
    propose(SAP) from P;
    choice at C {
      acpt() to P;
      confirm() from P;
    } or {
      reject() to P;
    } or {
      propose(SAP) to P;
      continue X;
    } } }
}
```



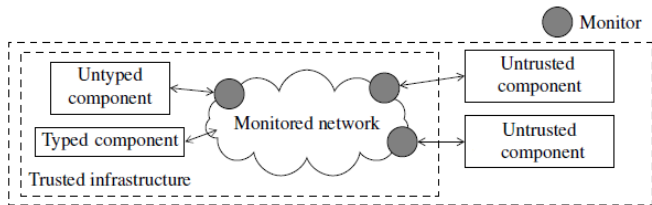
Python Conversation API

```
class UserApp(BaseApp):
    def start(self):
        conv = Conversation.create('Negotiate', 'config.yml')
        with conv.join(C, 'consumer') as c:
            c.send(P, 'propose', sap)
            aux(c, sap)

    def aux(self, c, sap):
        msg = c.recv(P) # Monitor ensures accept/propose/reject
        if msg.label == 'accept':
            c.send(P, 'confirm')
        elif msg.label == 'propose':
            if isAcceptable(msg.arg[0]):
                c.send(P, 'accept')
                c.receive(P, 'confirm')
            elif isNegotiable(msg.arg[0]):
                sap2 = revise(msg.arg[0])
                c.send(P, 'propose', sap1)
                aux(c, sap1)
        else:
            c.send(P, 'reject')
```

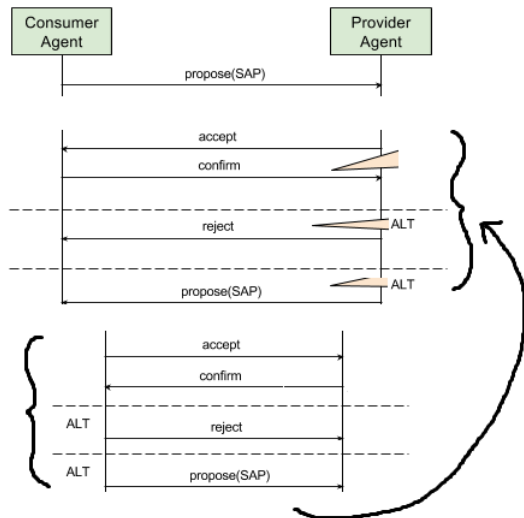
- ▶ Endpoints implemented using Scribble-Python API
- ▶ Inline (“synchronous”) vs. outline (“asynchronous”) monitoring

MPST-based distributed protocol monitoring



- ▶ Dynamic verification of MPST communication safety
 - ▶ Session fidelity: correspondence between system of monitored endpoints and the original global specification
 - ▶ Local transparency: a monitored process has equivalent behaviour to an unmonitored but statically verified process
- ▶ Interoperability
 - [FMOODS13] *Monitoring networks through multiparty session types*. Bocchi, Chen, Demangeon, Honda and Yoshida.
 - [RV13] *Practical Interruptible Conversations*. Hu, Neykova, Yoshida, Demangeon and Honda.
 - [TGC11] *Asynchronous Distributed Monitoring for Multiparty Session Enforcement*. Chen, Bocchi, Deniérou, Honda and Yoshida.

Exercise: refactor Negotiate



```
// Protocol decl  
global protocol Proto  
    (role R1, role R2) {  
    ...  
}
```

```
// Message passing  
123(T) from R1 to R2;
```

```
// Located choice  
choice at R {  
    ...  
} or {  
    ...  
}
```

```
// "Subprotocol"  
do Proto(R1, R2);
```

Exercise: refactor Negotiate

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

```
// Protocol decl
global protocol Proto
  (role R1, role R2) {
  ...
}

// Message passing
123(T) from R1 to R2;

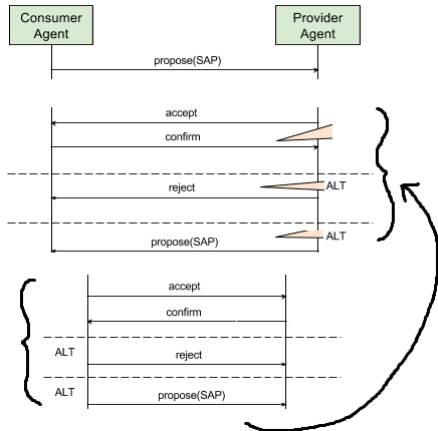
// Located choice
choice at R {
  ...
} or {
  ...
}

// "Subprotocol"
do Proto(R1, R2);
```

Exercise: refactor Negotiate

```
global protocol Negotiate(role C, role P) {  
  propose(SAP) from C to P;  
  do Aux(P, C);  
}
```

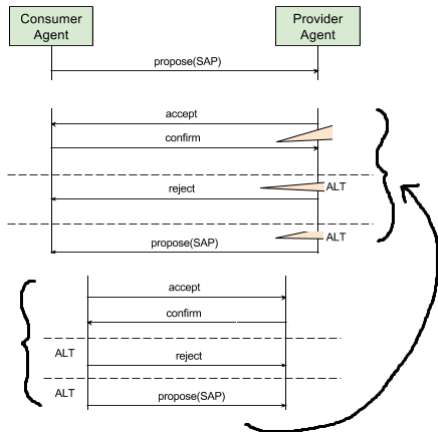
```
global protocol Aux(role A, role B) {  
  choice at A {  
    accpt() from A to B;  
    confirm() from B to A;  
  } or {  
    reject() from A to B;  
  } or {  
    propose(SAP) from A to B;  
    do Aux(B, A);  
  }  
}
```



Exercise: refactor Negotiate

```
global protocol Negotiate(role C, role P) {  
  propose(SAP) from C to P;  
  do Aux(P, C);  
}
```

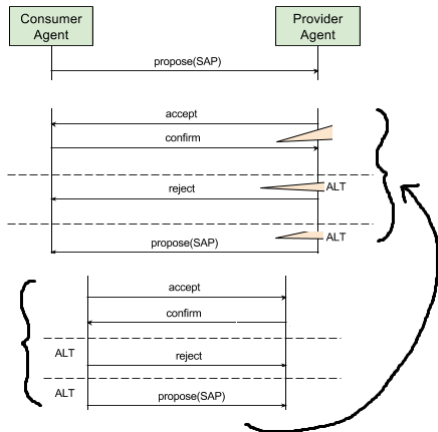
```
global protocol Aux(role A, role B) {  
  choice at A {  
    acpt() from A to B;  
    confirm() from B to A;  
  } or {  
    reject() from A to B;  
  } or {  
    propose(SAP) from A to B;  
    do Aux(B, A);  
  }  
}
```



Exercise: refactor Negotiate

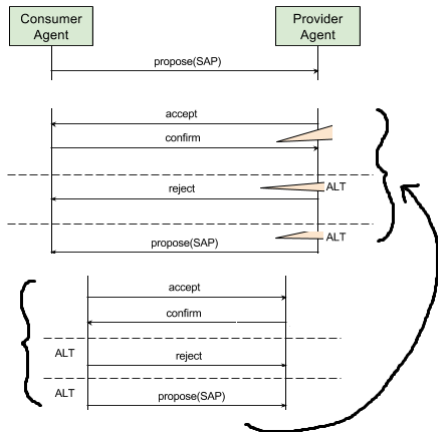
```
global protocol Negotiate(role C, role P) {  
  propose(SAP) from C to P;  
  do Aux(P, C);  
}
```

```
global protocol Aux(role A, role B) {  
  choice at A {  
    acppt() from A to B;  
    confirm() from B to A;  
  } or {  
    reject() from A to B;  
  } or {  
    propose(SAP) from A to B;  
    do Aux(B, A);  
  }  
}
```



Exercise: refactor Negotiate

```
global protocol Negotiate(role C, role P) {  
  propose(SAP) from C to P;  
  choice at P {  
    acpt() from P to C;  
    confirm() from C to P;  
  } or {  
    reject() from P to C;  
  } or {  
    do Negotiate(P, C);  
  }  
}
```



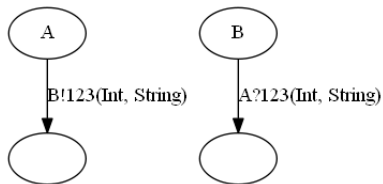
Good/bad MPST by example

- ▶ Core Scribble constructs (review)
 - ▶ Further illustration of endpoint FSMs
- ▶ MPST safety and liveness errors (informally)
 - ▶ What can go wrong in a “bad” session type?
 - ▶ How are they ruled out in formal MPST (syntactically)
- ▶ <https://github.com/rhu1/scribble-java/tree/rhu1-research/modules/core/src/test/scrib/demo/betty16/lec1/misc>

Role-to-role message passing

123(Int, String) from A to B;

- ▶ Message signature
 - ▶ Operator (header, label, ...)
 - ▶ Payload types



() from A to B;

- ▶ Empty operator and/or payload OK

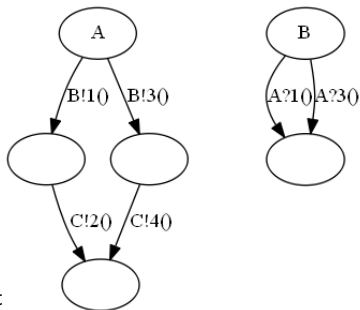
Choice

- ▶ “Located” multiparty choice

```
choice at A {  
  1() from A to B;  
  2() from A to C;  
} or {  
  3() from A to B;  
  4() from A to C;  
}
```

- ▶ Internal choice by global choice subject
- ▶ External choice for all other involved roles

- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role



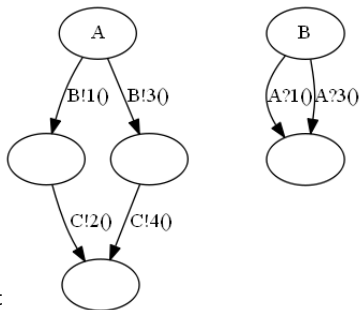
Choice

- ▶ “Located” multiparty choice

```
choice at A {  
  1() from A to B;  
  2() from A to C;  
} or {  
  3() from A to B;  
  4() from A to C;  
}
```

- ▶ Internal choice by global choice subject
- ▶ External choice for all other involved roles

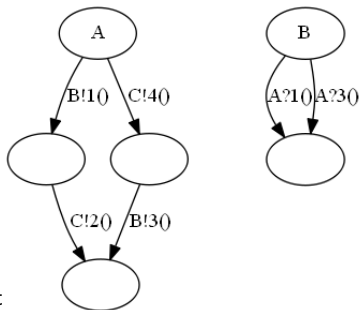
- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role



Choice

- ▶ “Located” multiparty choice

```
choice at A {  
  1() from A to B;  
  2() from A to C;  
} or {  
  4() from A to C;  
  3() from A to B;  
}
```



- ▶ Internal choice by global choice subject
- ▶ External choice for all other involved roles

- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role

“Located” choice

```
choice at A {  
  buyer1(Int) from A to B; // Total to pay  
  (Int) from B to A; // B will pay this much  
  buyer2(Int) from A to C; // C pays remainder  
} or {  
  buyer1(Int) from A to C; // Total to pay  
  (Int) from C to A; // C will pay this much  
  buyer2(Int) from A to B; // B pays remainder  
}
```

- ▶ More “flexible” than “directed” choice

$p \rightarrow q : \{l_i : G_i\}_{i \in I}$ Branching

- ▶ Branching via messages with identical payloads OK (cf. [POPL11])

```
choice at A { 1() from A to B; } or { 1(Int) from A to B; } ✗
```

“Located” choice

```
choice at A {  
  buyer1(Int) from A to B; // Total to pay  
  (Int) from B to A; // B will pay this much  
  buyer2(Int) from A to C; // C pays remainder  
} or {  
  buyer1(Int) from A to C; // Total to pay  
  (Int) from C to A; // C will pay this much  
  buyer2(Int) from A to B; // B pays remainder  
}
```

- ▶ More “flexible” than “directed” choice

$$p \rightarrow q : \{l_i : G_i\}_{i \in I} \quad \text{Branching}$$

- ▶ Branching via messages with identical payloads OK (cf. [POPL11])

```
choice at A { 1() from A to B; } or { 1(Int) from A to B; } X
```

Exercise: role enabling

- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role

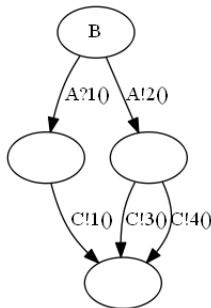
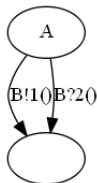
```
choice at A {  
  1() from A to B;  
  1() from B to C;  
  
} or {  
  2() from B to A;  
  choice at B {  
    2() from B to C;  
  } or {  
    3() from B to C;  
  }  
  
}
```

- ▶ Syntactic Scribble error?

Exercise: role enabling

- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role

```
choice at A {  
  1() from A to B;  
  1() from B to C;  
}  
or {  
  2() from B to A;  
  choice at B {  
    2() from B to C;  
  } or {  
    3() from B to C;  
  }  
}
```

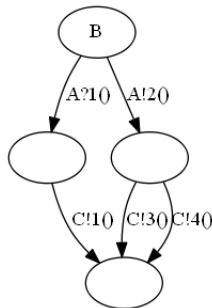
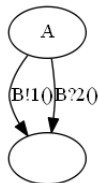


- ▶ Syntactic Scribble error? **B** not enabled (“mixed choice” protocol states)

Exercise: role enabling

- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role

```
choice at A {  
  1() from A to B;  
  1() from B to C;  
}  
or {  
  2() from B to A;  
  choice at B {  
    2() from B to C;  
  } or {  
    3() from B to C;  
  }  
}
```



- ▶ Syntactic Scribble error? **B** not enabled (“mixed choice” protocol states)
- ▶ What actually “goes wrong”?

Exercise: role enabling

- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role

```
choice at A {  
  1() from A to B;  
  1() from B to C;  
  1() from C to A;  
} or {  
  2() from B to A;  
  choice at B {  
    2() from B to C;  
  } or {  
    3() from B to C;  
  }  
  4() from C to A;  
}
```

- ▶ Syntactic Scribble error? **B** not enabled (“mixed choice” protocol states)
- ▶ What actually “goes wrong”?

Exercise: role enabling

- ▶ Only *enabled* roles can send messages in choice paths
 - ▶ Subject starts enabled; others disabled
 - ▶ A disabled role is enabled by receiving a message from an enabled role

```
choice at A {  
  1() from A to B;  
  1() from B to C;  
  1() from C to A;  
} or {  
  2() from B to A;  
  choice at B {  
    2() from B to C;  
  } or {  
    3() from B to C;  
  }  
  4() from C to A;  
}
```

- ▶ Syntactic Scribble error? **B** not enabled (“mixed choice” protocol states)
- ▶ What actually “goes wrong”?
 - ▶ MPST safety errors: *receptions errors, orphan messages, deadlock*

Is this choice OK? 1/4

```
choice at A {  
  1() from A to B;  
  3() from B to C;  
  4() from C to A;  
} or {  
  2() from A to B;  
  3() from B to C;  
  5() from C to A;  
}
```

Is this choice OK? 1/4

```
choice at A {  
  1() from A to B;  
  3() from B to C; X  
  4() from C to A;  
} or {  
  2() from A to B;  
  3() from B to C; X  
  5() from C to A;  
}
```

- ▶ “Ambiguous” choice to C
 - ▶ Should C send a 4 or a 5 to A?

Is this choice OK? 1/4

```
choice at A {  
  1() from A to B;  
  3() from B to C; X  
  4() from C to A;  
} or {  
  2() from A to B;  
  3() from B to C; X  
  5() from C to A;  
}
```

- ▶ “Ambiguous” choice to C
 - ▶ Should C send a 4 or a 5 to A?
 - ▶ Potential *reception errors* (4, 5), if interpreted non-deterministically

Is this choice OK? 1/4

```
choice at A {  
  1() from A to B;  
  3() from B to C; X  
  4() from C to A;  
} or {  
  2() from A to B;  
  3() from B to C; X  
  5() from C to A;  
}
```

- ▶ “Ambiguous” choice to C
 - ▶ Should C send a 4 or a 5 to A?
 - ▶ Potential *reception errors* (4, 5), if interpreted non-deterministically
- ▶ Non-det external choice at C inconsistent with original internal choice by A
 - ▶ Not “mergeable” in syntactic projection
(Need to merge continuations: undefined for distinct outputs)
 - ▶ Simple fix: distinguish the 3's (distinct external choice ops mergeable)

Is this choice OK? 2/4

```
choice at A {  
  1() from A to B;  
  3() from B to C;  
  4() from A to C;  
} or {  
  2() from A to B;  
  3() from B to C;  
  4() from A to C;  
}
```

Is this choice OK? 2/4

```
choice at A {  
  1() from A to B;  
  3() from B to C;  
  4() from A to C; ✓  
} or {  
  2() from A to B;  
  3() from B to C;  
  4() from A to C; ✓  
}
```


Is this choice OK? 2/4

```
choice at A {  
  1() from A to B;  
  3() from B to C;  
  do Merge(A, C);  
} or {  
  2() from A to B;  
  3() from B to C;  
  do Merge(A, C);  
}
```

```
global protocol Merge(role A, role C) {  
  4() from A to C;  
}
```

- ▶ Duplicate cases inherently mergeable, e.g. [\[POPL11\]](#)

Is this choice OK? 2/4

```
choice at A {  
  1() from A to B;  
  3() from B to C;  
  do Merge(A, C);  
} or {  
  2() from A to B;  
  3() from B to C;  
  do Merge(A, C);  
}
```

```
global protocol Merge(role A, role C) {  
  4() from C to A;  
}
```

- ▶ Duplicate cases inherently mergeable, e.g. [\[POPL11\]](#)

Is this choice OK? 2/4

```
choice at A {  
  1() from A to B;  
  3() from B to C;  
  do Merge(A, C);  
} or {  
  2() from A to B;  
  3() from B to C;  
  do Merge(A, C);  
}
```

```
global protocol Merge(role A, role C) {  
  choice at A {  
    4() from A to C;  
  } or {  
    5() from A to C;  
  }  
}
```

- ▶ Duplicate cases inherently mergeable, e.g. [POPL11]

Is this choice OK? 3/4

```
choice at A {  
  1a() from A to B;  
  2() from A to C;  
  3() from B to C;  
  4() from C to A;  
} or {  
  1b() from A to B;  
  3() from B to C;  
  4() from C to A;  
}
```

Is this choice OK? 3/4

```
choice at A {  
  1a() from A to B;  
  2() from A to C; X  
  3() from B to C;  
  4() from C to A;  
} or {  
  1b() from A to B;  
  3() from B to C; X  
  4() from C to A;  
}
```

- ▶ “Race condition” in choice to C due to asynchrony
 - ▶ What should C do after receiving a 3?

Is this choice OK? 3/4

```
choice at A {  
  1a() from A to B;  
  2() from A to C; X  
  3() from B to C;  
  4() from C to A;  
} or {  
  1b() from A to B;  
  3() from B to C; X  
  4() from C to A;  
}
```

- ▶ “Race condition” in choice to C due to asynchrony
 - ▶ What should C do after receiving a 3?
 - ▶ Potential *orphan message* (2), if interpreted as “multi-queue FIFO”

Is this choice OK? 3/4

```
choice at A {  
  1a() from A to B;  
  2() from A to C; X  
  3() from B to C;  
  4() from C to A;  
} or {  
  1b() from A to B;  
  3() from B to C; X  
  4() from C to A;  
}
```

- ▶ “Race condition” in choice to C due to asynchrony
 - ▶ What should C do after receiving a 3?
 - ▶ Potential *orphan message* (2), if interpreted as “multi-queue FIFO”
- ▶ Inconsistent external choice subjects
 - ▶ (Trivially non-mergeable in standard MPST)
 - ▶ A role must be enabled by the same role in all choice paths

Is this choice OK? 4/4

```
choice at A {  
  1() from A to B;  
  2() from A to C;  
} or {  
  3() from A to B;  
}
```


Is this choice OK? 4/4

```
choice at A {  
  1() from A to B;  
  2() from A to C; X  
} or {  
  3() from A to B;  
} X
```

- ▶ “Unrealisable” choice for C
 - ▶ No implicit messages can be assumed, e.g., end-of-session
 - ▶ How can C locally determine if no message is coming?

Is this choice OK? 4/4

```
choice at A {  
  1() from A to B;  
  2() from A to C; X  
} or {  
  3() from A to B;  
} X
```

- ▶ “Unrealisable” choice for c
 - ▶ No implicit messages can be assumed, e.g., end-of-session
 - ▶ How can C locally determine if no message is coming?
 - ▶ Potential *deadlock* (C waiting-for A), or potential *orphan* (2), depending on interpretation

Is this choice OK? 4/4

```
choice at A {  
  1() from A to B;  
  2() from A to C; X  
} or {  
  3() from A to B;  
} X
```

- ▶ “Unrealisable” choice for c
 - ▶ No implicit messages can be assumed, e.g., end-of-session
 - ▶ How can C locally determine if no message is coming?
 - ▶ Potential *deadlock* (C waiting-for A), or potential *orphan* (2), depending on interpretation
- ▶ Empty action option to terminal state
 - ▶ Cannot merge end type with anything else

Recursion

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
    2() from A to B;  
  } or {  
    3() from A to B;  
  }  
  4() from A to B;  
}  
5() from A to B;
```

- ▶ Tail recursion within recursive scopes

Recursion

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
    2() from A to B; X  
  } or {  
    3() from A to B;  
  }  
  4() from A to B;  
}  
5() from A to B;
```

- ▶ Tail recursion within recursive scopes
 - ▶ Reachability of protocol states (no “dead code”)

Recursion

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
    2() from A to B; ✗  
  } or {  
    3() from A to B;  
  }  
  4() from A to B; ✗  
}  
5() from A to B;
```

- ▶ Tail recursion within recursive scopes
 - ▶ Reachability of protocol states (no “dead code”)
 - ▶ Regular interaction structure at endpoints (CFSM model)

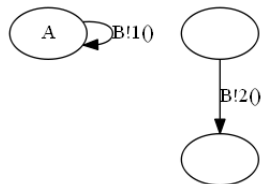
Recursion

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from A to B;
```

- ▶ Reachability of protocol states

Recursion

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from A to B; X
```

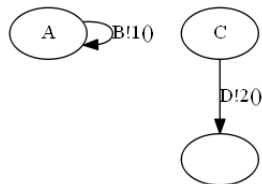


- ▶ Reachability of protocol states

Recursion

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from A to B; X
```

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from C to D;
```

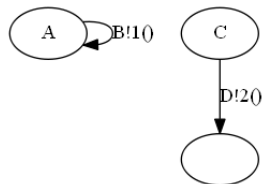


- Reachability of protocol states

Recursion

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from A to B; ✗
```

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from C to D; ✓
```

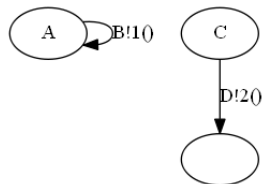


- Reachability of protocol states

Recursion

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from A to B; ✗
```

```
rec X {  
  1() from A to B;  
  continue X;  
}  
2() from C to D; ✓
```



- ▶ Reachability of protocol states checked via projections
 - ▶ (Reachability wrt. “per-role” protocol flow)

Is this protocol OK? 1/2

```
choice at A {  
  rec X {  
    1() from A to B;  
    1() from B to C;  
    continue X;  
  }  
} or {  
  2() from A to B;  
  2() from B to C;  
}
```

- ▶ Safety errors? (reception errors, orphan messages, deadlock)

Is this protocol OK? 1/2

```
choice at A {  
  rec X {  
    1() from A to B;  
    1() from B to C;  
    continue X;  
  }  
} or {  
  2() from A to B;  
  2() from B to C;  
}
```

- ▶ Safety errors? (reception errors, orphan messages, deadlock)
 - ▶ Endpoint FSM for A?

Is this protocol OK? 1/2

```
choice at A {  
  rec X {  
    1() from A to B;  
    //1() from B to C;  
    continue X;  
  }  
} or {  
  2() from A to B;  
  2() from B to C;  
}
```

- ▶ Safety errors? (reception errors, orphan messages, deadlock)
 - ▶ Endpoint FSM for A?
 - ▶ How about now?

Is this protocol OK? 1/2

```
choice at A {  
  rec X {  
    1() from A to B;  
    //1() from B to C;  
    continue X;  
  }  
} or {  
  2() from A to B;  
  2() from B to C;  
}
```

- ▶ Safety errors? (reception errors, orphan messages, deadlock)
 - ▶ Endpoint FSM for A?
 - ▶ How about now?
 - ▶ But is this a “good” protocol?

Is this protocol OK? 1/2

```
choice at A {  
  rec X {  
    1() from A to B;  
    //1() from B to C;  
    continue X;  
  }  
} or {  
  2() from A to B;  
  2() from B to C; X  
}
```

- ▶ Safety errors? (reception errors, orphan messages, deadlock) no
 - ▶ Endpoint FSM for A?
 - ▶ How about now?
 - ▶ But is this a “good” protocol?
- ▶ Liveness errors
 - ▶ Role progress

Is this protocol OK? 1/2

```
choice at A {  
  rec X {  
    1() from A to B;  
    //1() from B to C;  
    continue X;  
  }  
} or {  
  2() from A to B;  
  
}  
2() from C to B; X
```

- ▶ Safety errors? (reception errors, orphan messages, deadlock) no
 - ▶ Endpoint FSM for A?
 - ▶ How about now?
 - ▶ But is this a “good” protocol?
- ▶ Liveness errors
 - ▶ Role progress
 - ▶ Message liveness

Is this protocol OK? 2/2

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
  } or {  
    2() from A to B;  
    2() from B to C;  
  }  
}
```

- ▶ Is this a good protocol?

Is this protocol OK? 2/2

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
  } or {  
    2() from A to B;  
    2() from B to C; ?  
  }  
}
```

- ▶ Is this a good protocol?
 - ▶ Depends on...

Is this protocol OK? 2/2

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
  } or {  
    2() from A to B;  
    2() from B to C; ?  
  }  
}
```

- ▶ Is this a good protocol?
 - ▶ Depends on... *fairness* of output choice

Is this protocol OK? 2/2

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
  } or {  
    2() from A to B;  
    2() from B to C; ?  
  }  
}
```

- ▶ Is this a good protocol?
 - ▶ Depends on... *fairness* of output choice
- ▶ Session subtyping vs. fairness [MSCS16]

[MSCS16] *Fair subtyping for multi-party session types*. L. Padovani.

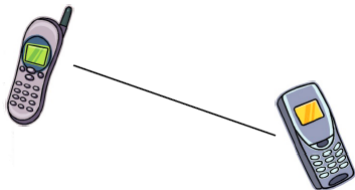
Homework

```
rec X {  
  choice at A {  
    1() from A to B;  
    2() from B to C;  
    3() from C to B;  
  } or {  
    4() from A to C;  
    5() from C to B;  
  }  
  continue X;  
}
```

- ▶ Why does Scribble not allow this protocol?
- ▶ What can “go wrong”?

Implementing session delegation

- ▶ Type safe connection dynamics
- ▶ Transparent to the “passive party”



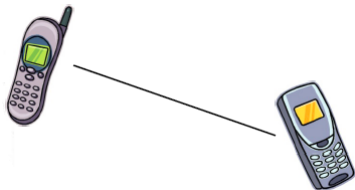
$$s[p]!\langle\langle q, s'[p'] \rangle\rangle.P \mid s : h \longrightarrow P \mid s : h \cdot (p, q, s'[p']) \quad [\text{Deleg}]$$

$$s[p]?((q, y)).P \mid s : (q, p, s'[p']) \cdot h \longrightarrow P\{s'[p']/y\} \mid s : h \quad [\text{SRcv}]$$

- ▶ Asynchrony modelled by decoupling input/output via (global) queue
 - ▶ All messages “rerouted” in transit

Implementing session delegation

- ▶ Type safe connection dynamics
- ▶ Transparent to the “passive party”



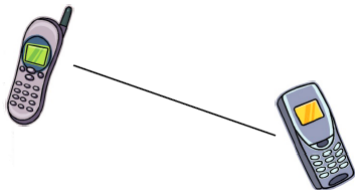
$$s[p]!\langle\langle q, s'[p'] \rangle\rangle.P \mid s : h \longrightarrow P \mid s : h \cdot (p, q, s'[p']) \quad [\text{Deleg}]$$

$$s[p]?((q, y)).P \mid s : (q, p, s'[p']) \cdot h \longrightarrow P\{s'[p']/y\} \mid s : h \quad [\text{SRcv}]$$

- ▶ Asynchrony modelled by decoupling input/output via (global) queue
 - ▶ All messages “rerouted” in transit

Implementing session delegation

- ▶ Type safe connection dynamics
- ▶ Transparent to the “passive party”



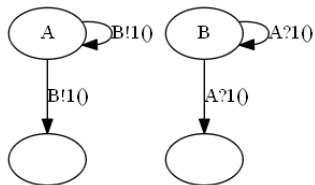
$$s[p]!\langle\langle q, s'[p'] \rangle\rangle.P \mid s : h \longrightarrow P \mid s : h \cdot (p, q, s'[p']) \quad [\text{Deleg}]$$

$$s[p]?((q, y)).P \mid s : (q, p, s'[p']) \cdot h \longrightarrow P\{s'[p']/y\} \mid s : h \quad [\text{SRcv}]$$

- ▶ Asynchrony modelled by decoupling input/output via (global) queue
 - ▶ All messages “rerouted” in transit

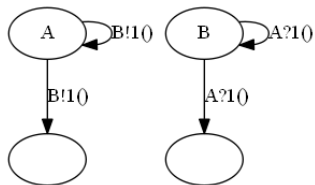
Is this protocol OK?

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
  } or {  
    1() from A to B;  
  }  
}
```



Is this protocol OK?

```
rec X {  
  choice at A {  
    1() from A to B;  
    continue X;  
  } or {  
    1() from A to B;  
  }  
}
```



- ▶ Potential deadlock or orphans
 - ▶ (This example is invalid branch/select syntax in standard MPST)