Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan

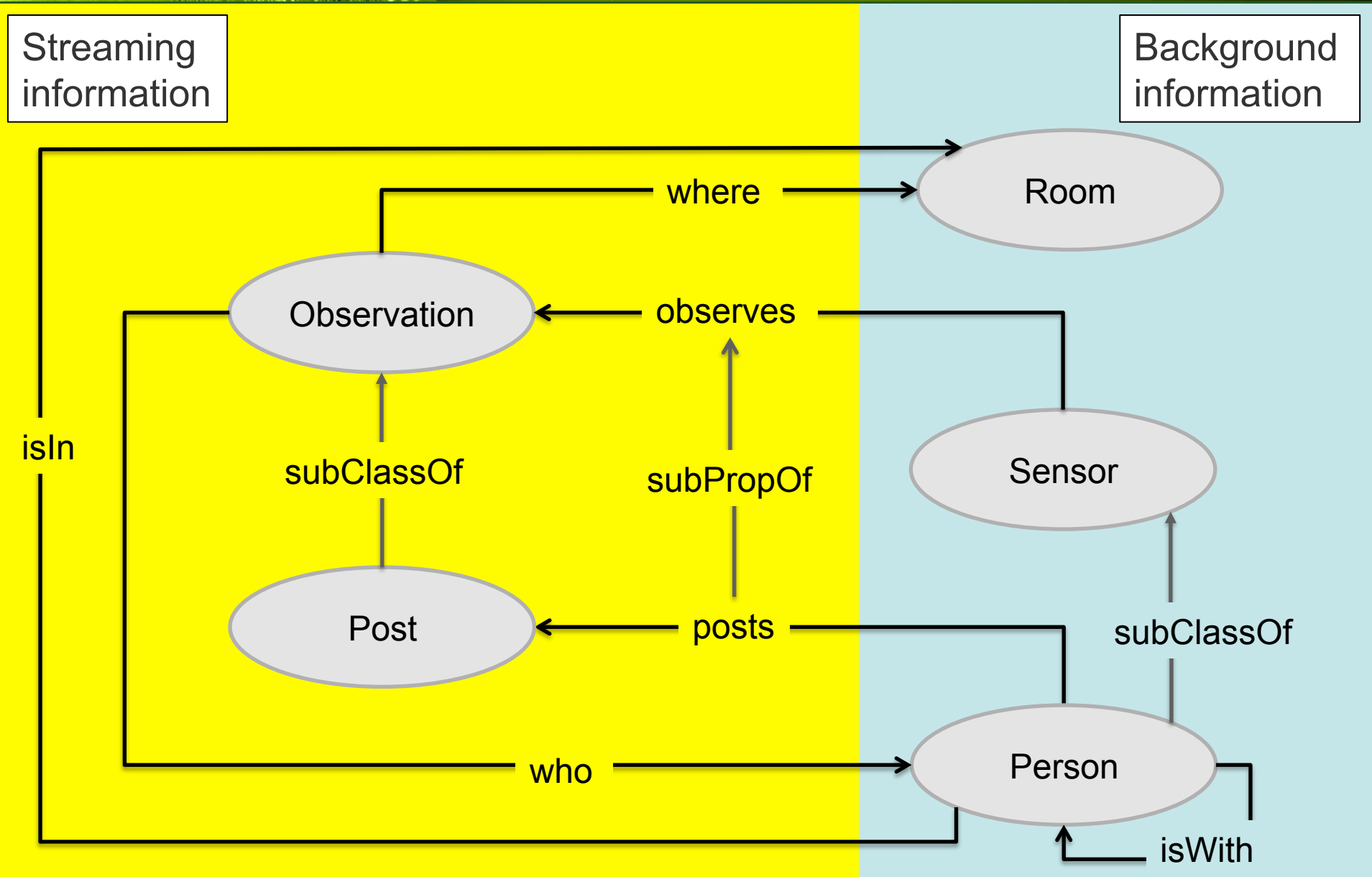
http://streamreasoning.org/sr4ld2013

C-SPARQL:Hands on Session

Marco Balduini
marco.balduini@polimi.it

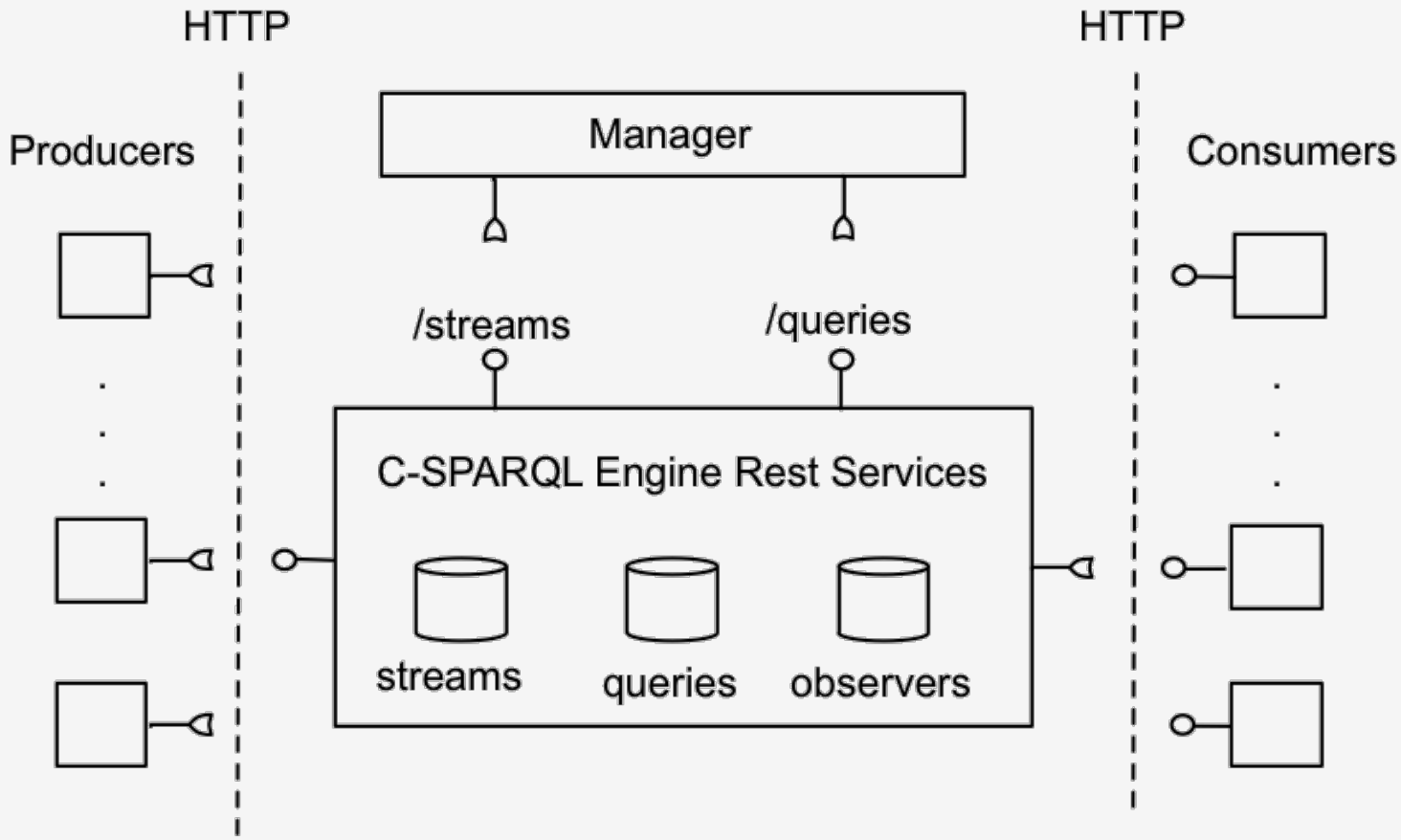
Share, Remix, Reuse — Legally

Schema Reminder



RDF Stream Processors Services

- Implementation of RSP Services protocol for C-SPARQL Engine



Package Content

- RSP_Services_CSPARQL folder
  - rsp-services-csparql-0.3.2.3.jar : executable jar containing the implemantation of rsp-services for the C-SPARQL Engine
    - To start it run rsp-services-csparql.sh (Unix) or rsp-services-csparql.bat (Windows)
- Http2Sysout
  - http2sysout-0.1.jar : executable jar containing a simple program to receive results over HTTP and print the received data on system out.
    - To start it run http2sysout.sh (Unix) or http2sysout.bat (Windows)
- SR4LD2013_Data.txt : a txt file containing all the exercises data (Descriptions, URLs, queries etc…)
- Java rest client
  - To start it type *: java –jar restclient-ui-3.2.1-jar-with-dependencies.jar*

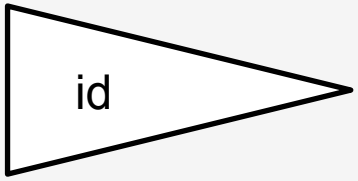
Program

1. Foursquare IsIn **example**
   - We will illustrate a guided example that checks who is in each room. The input RDF stream contains "Foursquare" check-in actions. The query we register produces an RDF stream.

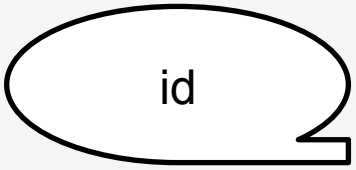
2. Facebook IsWith **exercise**
   - You will be asked to create a pipeline to check who is with whom. The input RDF stream contains "Facebook" posts. The query has to produce an RDF stream.

3. IsInWith **example**
   - We will illustrate a guided example that checks, for each room, who is with whom. As inputs, we will use the streams of the first two exercises. The query we register produces an RDF stream.

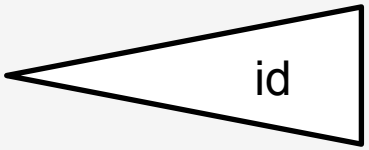
Legend



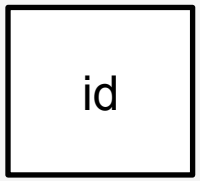
Stream: an RDF stream in the Engine. The id is specified by the user during the registration process



C-SPARQL Query: the id is specified by the user during registration process



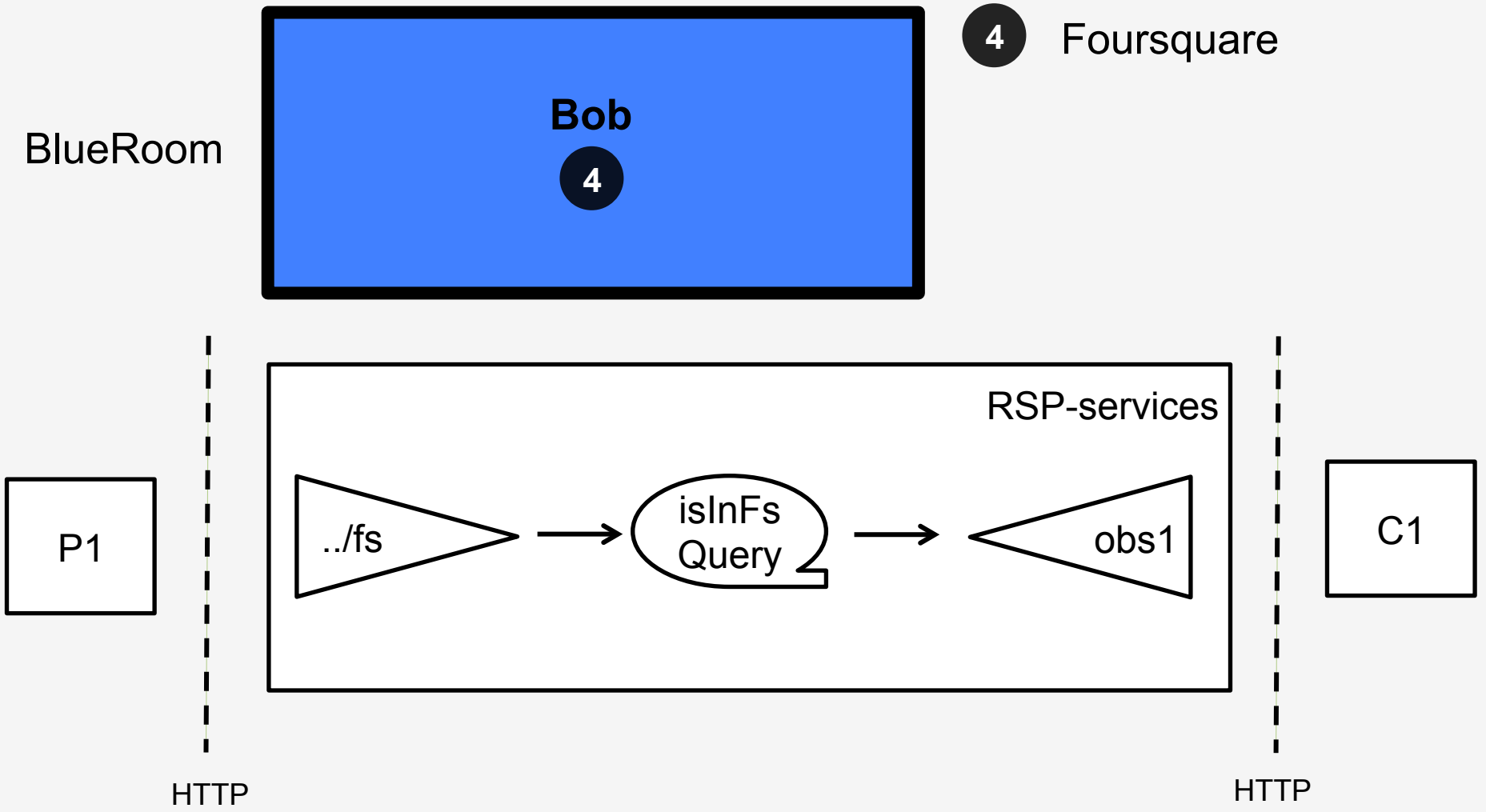
Observer: the query result from the Engine. The id is assigned by the engine during the registration process



Producer/Consumer: object that pushes data into specified stream (Producer) or receives data from observer (Consumer)

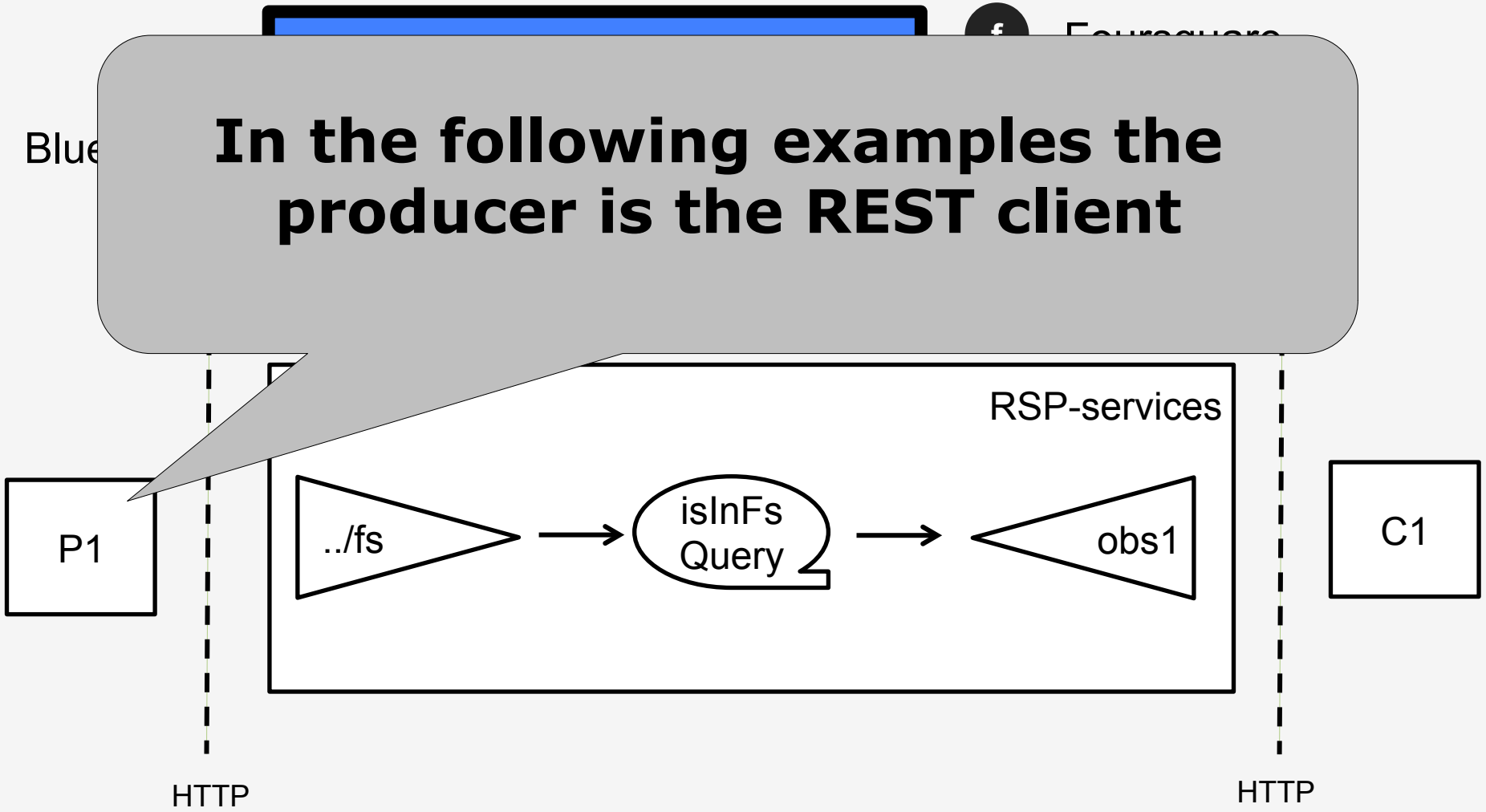
Foursquare IsIn example

Example that checks who is in each room using "Foursquare" check-in actions



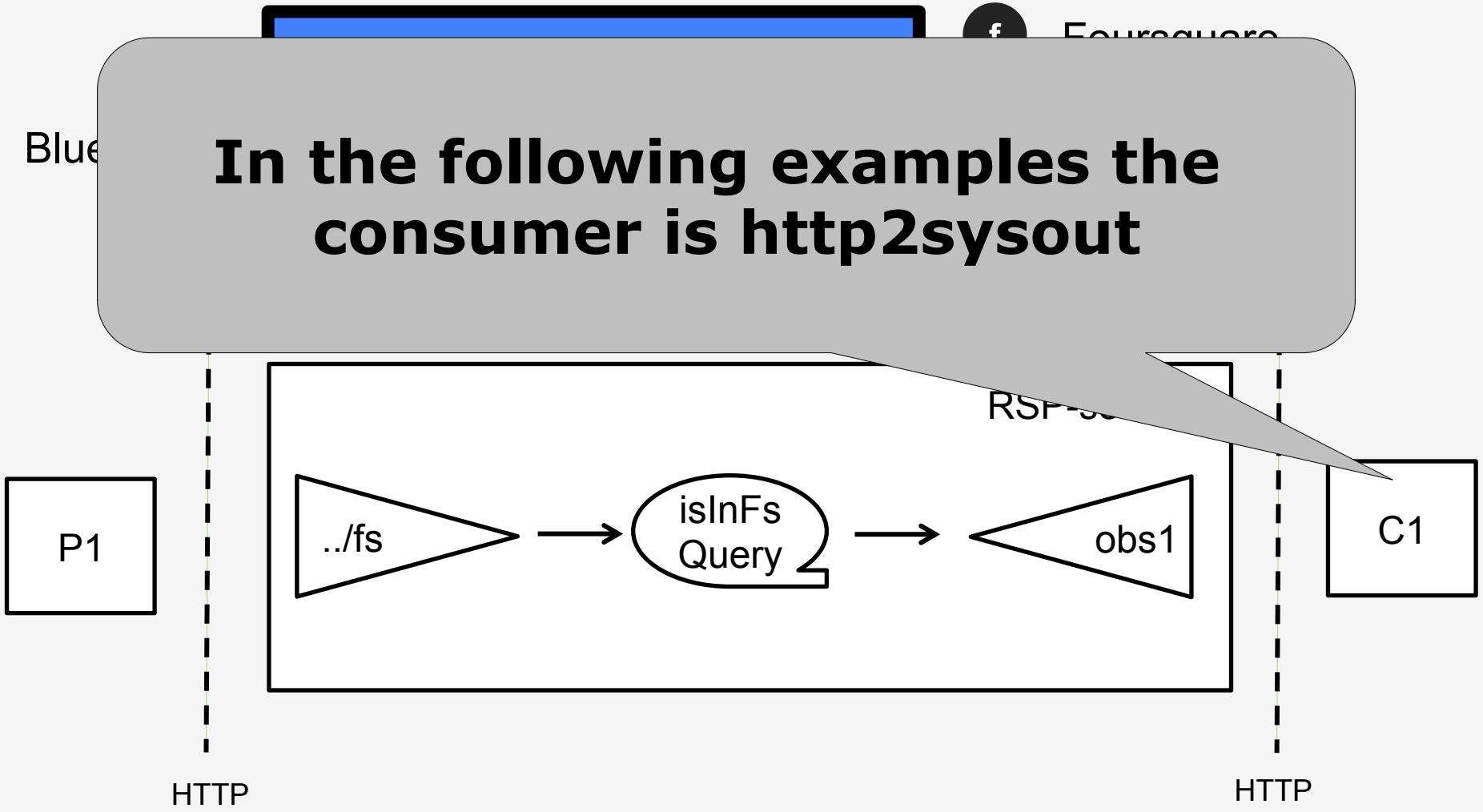
Foursquare IsIn example

Example that checks who is in each room using "Foursquare" check-in actions



Foursquare IsIn example

Example that checks who is in each room using "Foursquare" check-in actions



Foursquare IsIn example

1. Register the input RDF stream into the C-SPARQL engine

PUT
http://localhost:8175/streams/http%3A%2F%2Fex.org%2Ffs

- http%3A%2F%2Fex.org%2Ffs is the encoded version of the stream name (http://ex.org/fs). This is the unique id of the stream
- Every registered stream could be used in a query into the FROM STREAM clause

Foursquare IsIn example

2. Register a new query into the C-SPARQL engine

PUT http://localhost:8175/queries/IsInFs

```
REGISTER STREAM IsInFs AS
PREFIX : <http://.../sr4ld2013-onto#>
CONSTRUCT { ?person :isIn ?room }
FROM STREAM <http://ex.org/fs> [RANGE 1m STEP 10s]
WHERE { ?person :posts [ :who ?person ; :where ?
room ] .}
```

- *IsInFsQuery* represents the name of the query and its unique id
- The parameter of the HTTP PUT body must be the query
- The name given in the REGISTER STREAM clause must be the same one specified in the URL

Foursquare IsIn example

3. Register a new observer for the IsInFsQuery query

POST http://localhost:8175/queries/IsInFs

http://localhost:8176/http2sysout

- The parameter of the HTTP POST body must be the callback URL.

Foursquare IsIn example

4. Feed the input RDF stream

POST
http://localhost:8175/streams/http%3A%2F%2Fex.org%2Ffs

```
@prefix : <http://.../sr4ld2013-onto# > .
:Bob :posts [ :who :Bob ; :where :BlueRoom ] .
```

- The parameter of the HTTP POST body must be the data to be pushed into stream.

Foursquare IsIn example

- Now wait to see the result on the system out console (produced by the http2sysout printer)
  - Expected Result:

```
{
   "http://.../sr4ld2013-onto# Bob" : {
     "http://.../sr4ld2013-onto# isIn" : [ {
       "type" : "uri" ,
       "value" : "http://.../sr4ld2013-onto# BlueRoom"
     }
     ]
   }
}
```

What We Have Learnt

1. Register a new RDF stream into the C-SPARQL engine

   PUT http://localhost:8175/streams/<*stream_URI*>

2. Register a new query into the C-SPARQL engine

   PUT http://localhost:8175/queries/<query_name>

3. Add an Observer to a query to retrieve the results

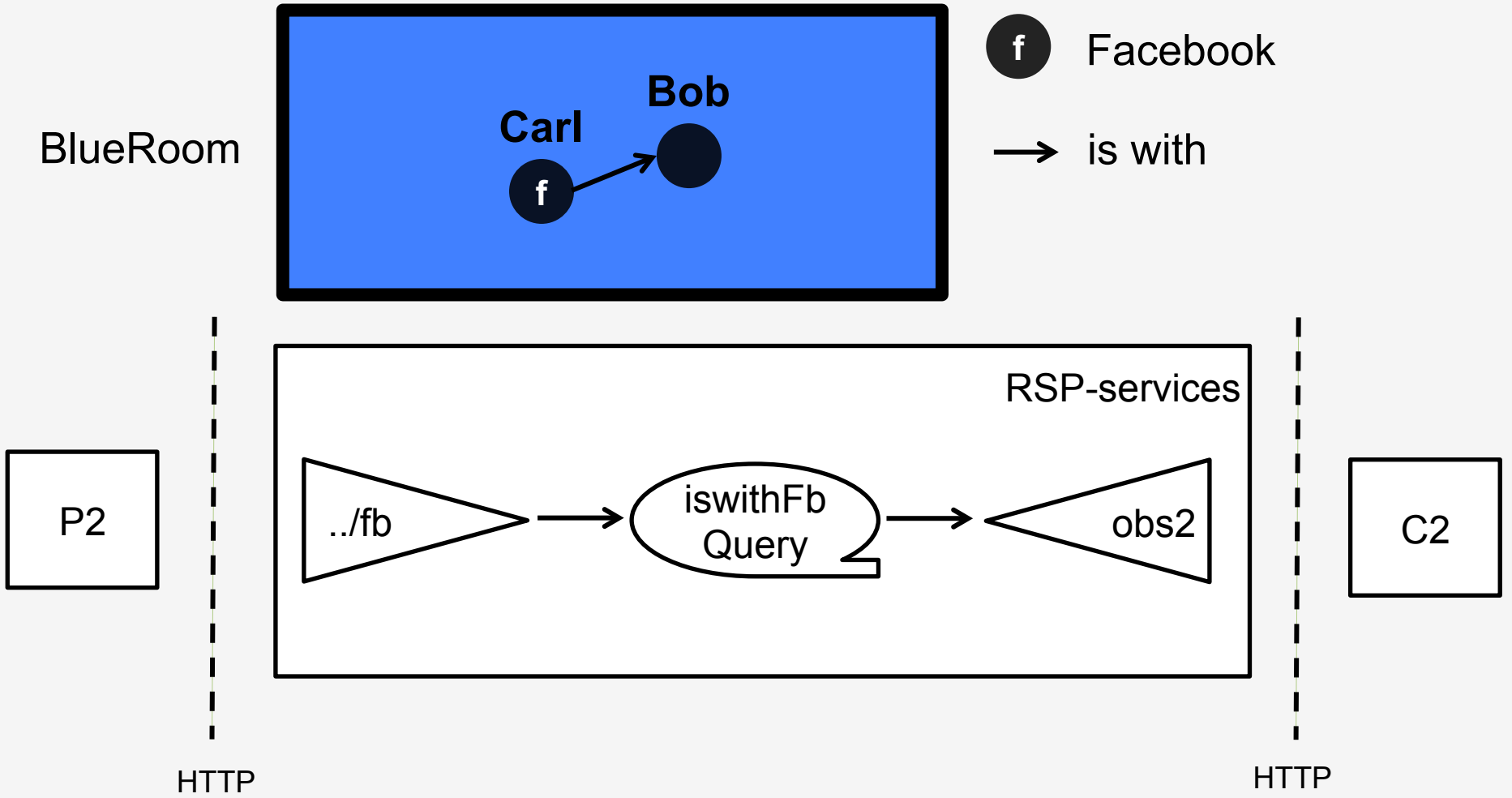
   POST http://localhost:8175/queries/<query_name>

4. Feed a registered RDF stream

   POST http://localhost:8175/streams/<*stream_URI*>

Facebook IsWith exercise

Create a pipeline that checks who is with whom using "Facebook" posts



Solution to Facebook IsWith exercise

1. Register the input RDF stream into the C-SPARQL engine

PUT

http://localhost:8175/streams/http%3A%2F%2Fex.org%2Ffb

- http%3A%2F%2Fex.org%2Ffb is the encoded version of the stream name (http://ex.org/fb). This is the unique id of the stream
- Every registered stream could be use in a query into FROM STREAM clause

Solution to Facebook IsWith exercise

2. Register a new query into the C-SPARQL engine

PUT http://localhost:8175/queries/isWithFb

```
REGISTER STREAM isWithFb AS
PREFIX : <http://.../sr4ld2013-onto#>
CONSTRUCT { ?person1 :isWith ?person }
FROM STREAM <http://ex.org/fb> [RANGE 1m STEP 10s]
WHERE { ?person1 :posts [ :who ?person1, ?person ].
FILTER (?person1!=?person)}
```

- *IsWithFbQuery* represents the name of the query and its unique id
- The parameter of the HTTP PUT body must be the query
- The name given in the REGISTER STREAM clause must be the same one specified in the URL

Solution to Facebook IsWith exercise

3. Register a new observer for the isWithFbQuery query

POST http://localhost:8175/queries/isWithFb

http://localhost:8176/http2sysout

- The parameter of the HTTP POST body must be the callback URL.

Solution to Facebook IsWith exercise

4. Feed the input RDF stream

POST
http://localhost:8175/streams/http%3A%2F%2Fex.org%2Ffb

```
@prefix : <http://.../sr4ld2013-onto# > .
:Carl :posts [ :who :Carl , :Bob ] .
```

- The parameter of the HTTP POST body must be the data to be pushed into stream.

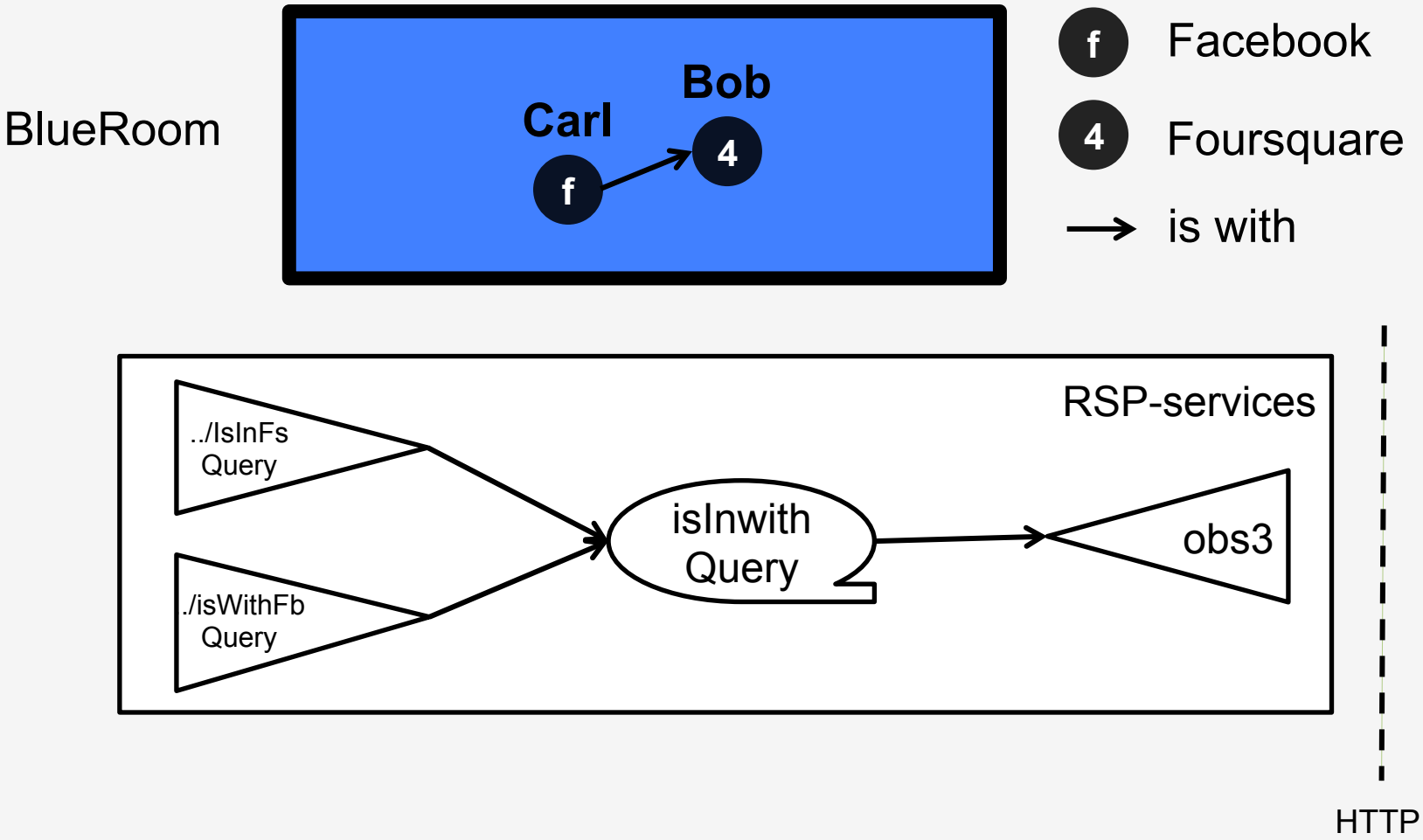
Solution to Facebook IsWith exercise

- Now wait to see the result on the system out console (produced by the http2sysout printer)
  - Expected Result

```
{
   "http://.../sr4ld2013-onto# Carl" : {
     "http://.../sr4ld2013-onto# isWith" : [ {
       "type" : "uri" ,
       "value" : “http://.../sr4ld2013-onto# Bob”
     }
     ]
   }
 }
```

IsInWith Example

Example that checks who are in each room together using the stream produced by previous queries



IsInWith Example

1. Register a new query into the C-SPARQL engine

PUT http://localhost:8175/queries/isInWith

```
REGISTER STREAM isInWith AS
PREFIX : <http://.../sr4ld2013-onto#>
CONSTRUCT { ?person :isIn ?room }
FROM STREAM <http://localhost:8175/streams/
IsInFsQuery> [RANGE 1m STEP 10s]
FROM STREAM <http://localhost:8175/streams/
isWithFbQuery> [RANGE 1m STEP 10s]
WHERE { { ?person :isWith ?person1 .
?person1 :isIn ?room .
FILTER (?person1!=?person)  }
}
```

IsInWith Example

2. Register a new observer for the isInWithQuery query

POST http://localhost:8175/queries/isInWith

http://localhost:8177/http2sysout

- The parameter of the HTTP POST body must be the callback URL.

IsInWith Example

3. Feed the input streams

POST
http://localhost:8175/streams/http%3A%2F%2Fex.org%2Ffs

```
@prefix : <http://.../sr4ld2013-onto# > .
:Bob :posts [ :who :Bob ; :where :BlueRoom ] .
```

POST
http://localhost:8175/streams/http%3A%2F%2Fex.org%2Ffb

```
@prefix : <http://.../sr4ld2013-onto# > .
:Carl :posts [ :who :Carl , :Bob ] .
```

IsInWith Example

- Now wait to see the result on the system out console (produced by the http2sysout printer)
  - Expected Result

```
{
  "http://…/sr4ld2013-onto#Carl" : {
    "http://…/sr4ld2013-onto#isIn" : [ {
      "type" : "uri" ,
      "value" : "http://…/sr4ld2013-onto#BlueRoom"
    }
     ]
  }
}
```

What We Have Learnt

1. Register a new RDF stream into the C-SPARQL engine

   PUT http://localhost:8175/streams/<*stream_URI*>

2. Register a new query into the C-SPARQL engine

   PUT http://localhost:8175/queries/<query_name>

3. Add an Observer to a query to retrieve the results

   POST http://localhost:8175/queries/<query_name>

4. Feed a registered RDF stream

   POST http://localhost:8175/streams/<*stream_URI*>

5. Perform a query chain
6. Explore base features of CSPARQL Engine

Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan

http://streamreasoning.org/sr4ld2013

C-SPARQL:Hands on Session

Marco Balduini
marco.balduini@polimi.it