The OpenKnowledge project

People talk about it. What is it??

OpenKnowledge

Decentralised, open, lightweight P2P framework

- Decentralised approach?
  - aimed at **choreography**, not orchestration
- Open?
  - peers do not have to be pre-configured at design-time
- Lightweight?
  - concrete implementation, small footprint (<15Mb)
  - no need for a system administrator to install and run it
- Peer-to-Peer?
  - the choreographies treat all peers as equal

Choreographies

- Choreographies offer a global view of interactions
- Choreographies provide a contractual agreement between the parties

Choreography Languages

- Majority of research has focused on orchestration (centralised) coordination languages
- Few choreography languages:
  - WS-CDL – W3C working draft (virtually abandoned)
  - BPEL4Chor (extension to BPEL)
  - Let's Dance (simulation not enactment)
- Even fewer implementations!

LCC as choreography language

- LCC is an executable specification language
- Based on process calculus:
  - facilitates model checking etc.
  - protocols are declarative scripts written in a Prolog-like language
- It uses **roles** for agents and **constraints** on message sending to enforce social norms

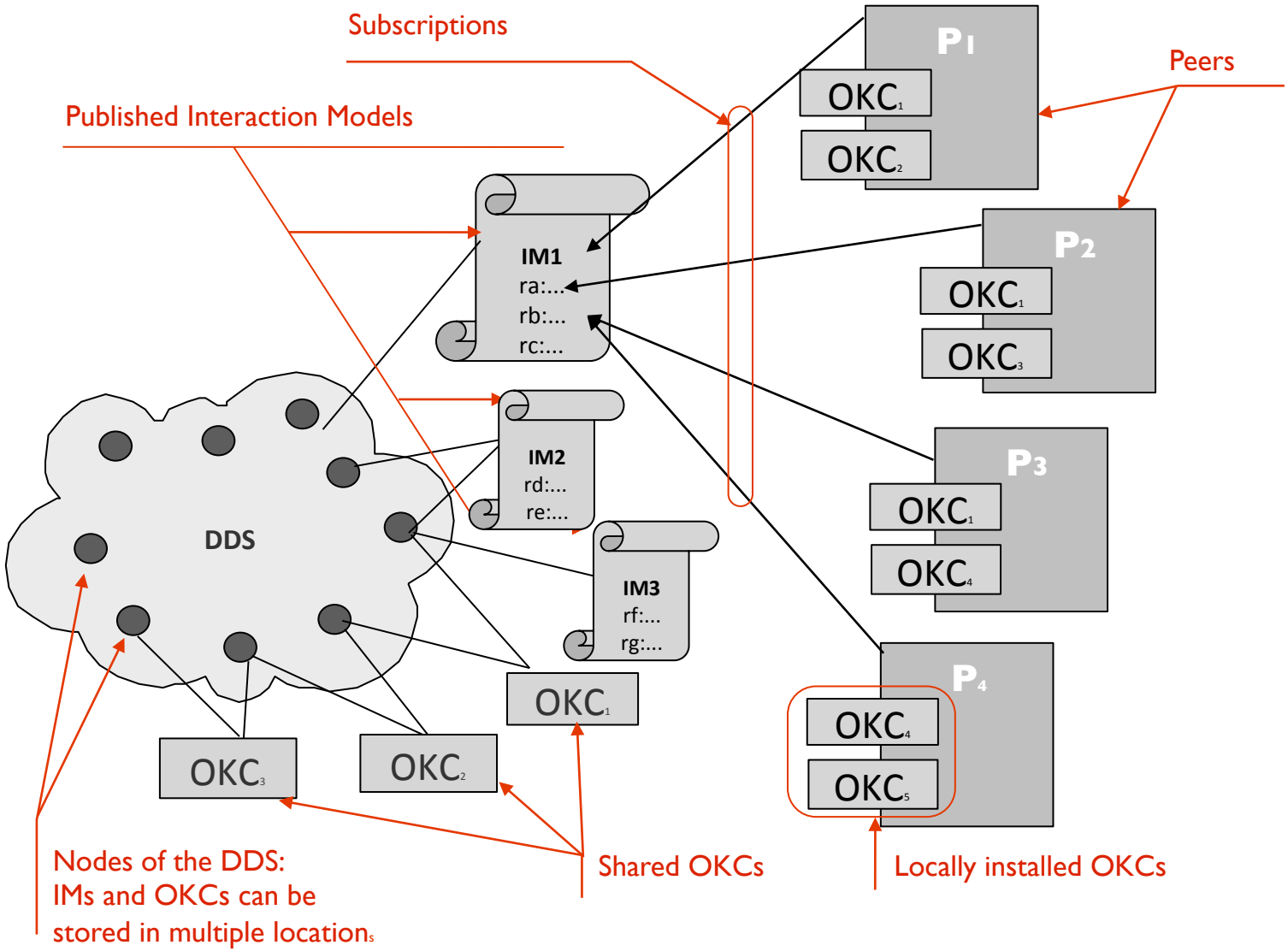
About LCC

- The basic behaviours are:
  - ⇒ for sending a message
  - ⇐ for receiving a message
  - no-op
- More complex behaviours can be expressed using connectives:
  - `then` to create sequences
  - `or` to create choices

Openk Framework structure

- The framework is composed by:
  - a distributed network of **peers** able to perform tasks through plug-in components (**OKCs**)
  - a distributed **Discovery Service** (**DS**)
- The tasks are specified by **Interaction Models** (**IMs**), written in *Lightweight Coordination Calculus*
  - Interaction models are choreographies

Framework architecture



Interaction lifecycle

1. IM selection and subscription
   a. Query to the DS
   b. Subscribe to the IM best fitting needs and capabilities
   c. Wait for all roles in the IM to be subscribed
2. Bootstrapping the interaction
   a. choice of the coordinator
   b. selection of peers
   c. commitment to participate
3. Interaction run

Choreography selection

- A peer selects choreographies for a task:
  - queries the DS with a task description (keywords)
  - DS matches choreography and task descriptions
  - sends back list of matching choreographies
- Peer matches constraints to its OKCs
  - uses ontological definition of arguments (wordnet)

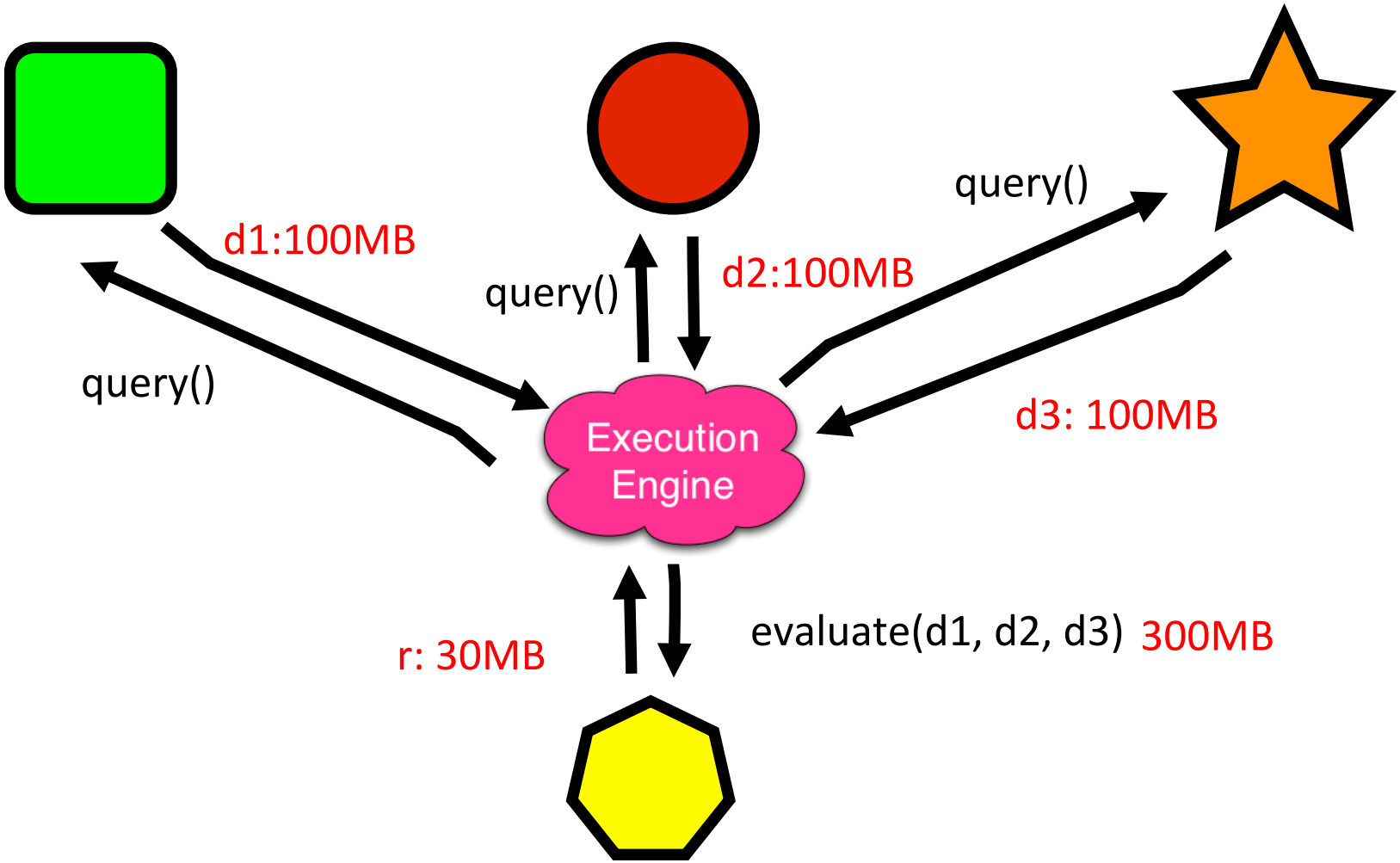
Link to e-science/DIR

- Peers own, control and process the data locally:
  - Processing elements (the OKC plug-ins) can be shared, downloaded by the peers and executed locally
  - IM describe their interactions, and what should be done on the data.
  - Data transfer can be minimised

Data flow efficiency

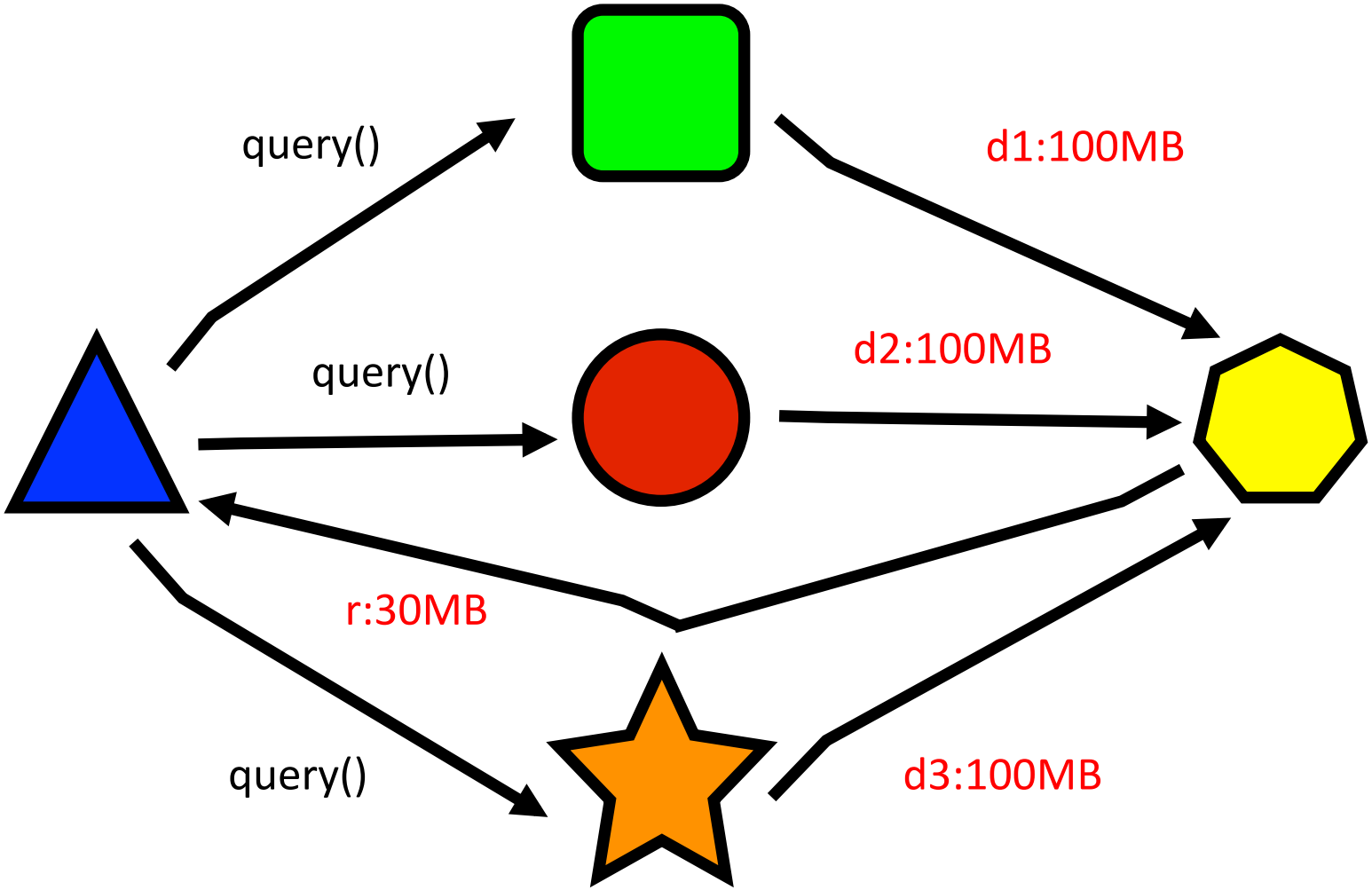
- Choreography improves efficiency of data flow:
  - no need of central engine as in orchestration
- Direct flow between processing nodes
- Involved participants and processing not embedded in the nodes, but flexible

Orchestration



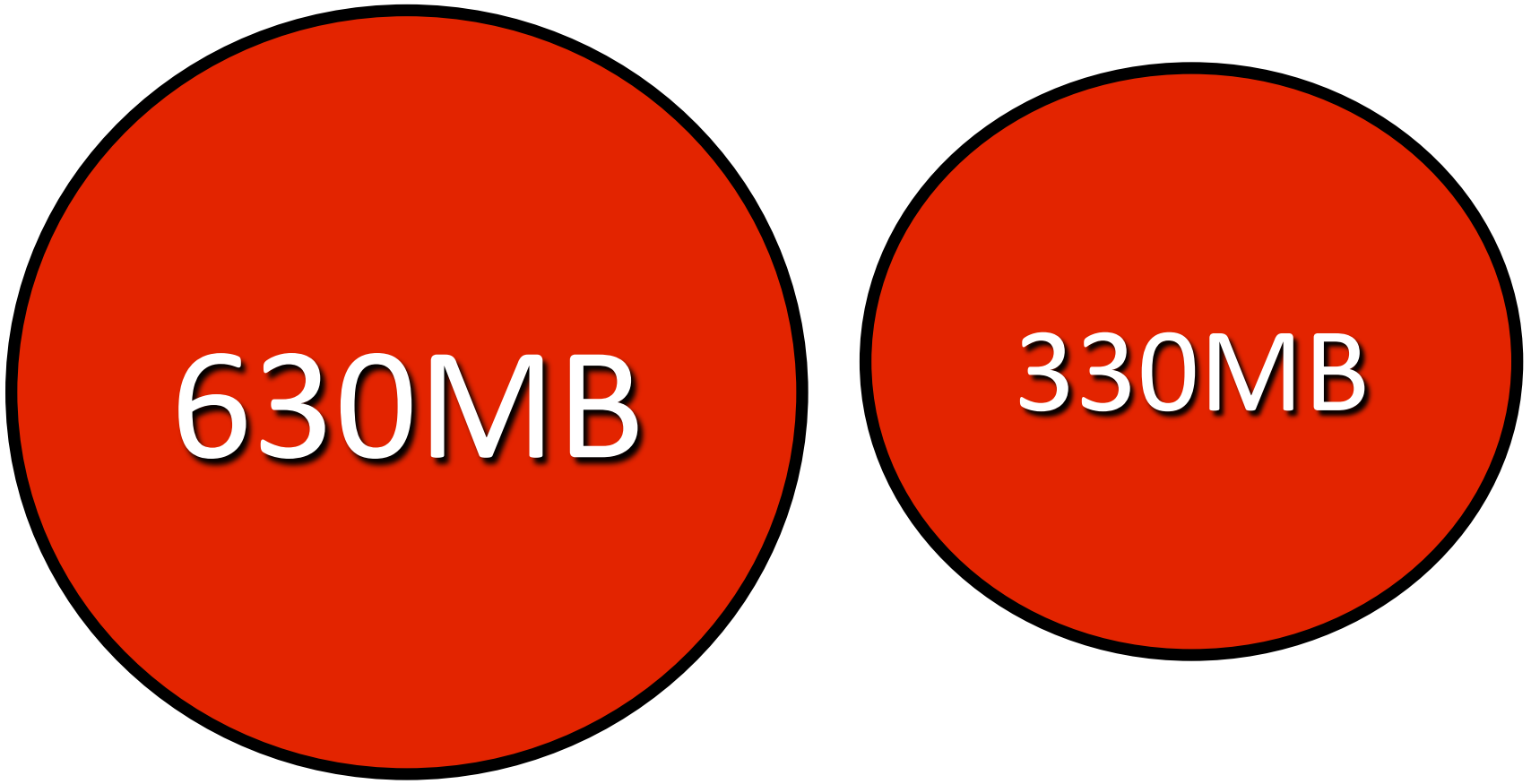
Thanks Barker

Choreography



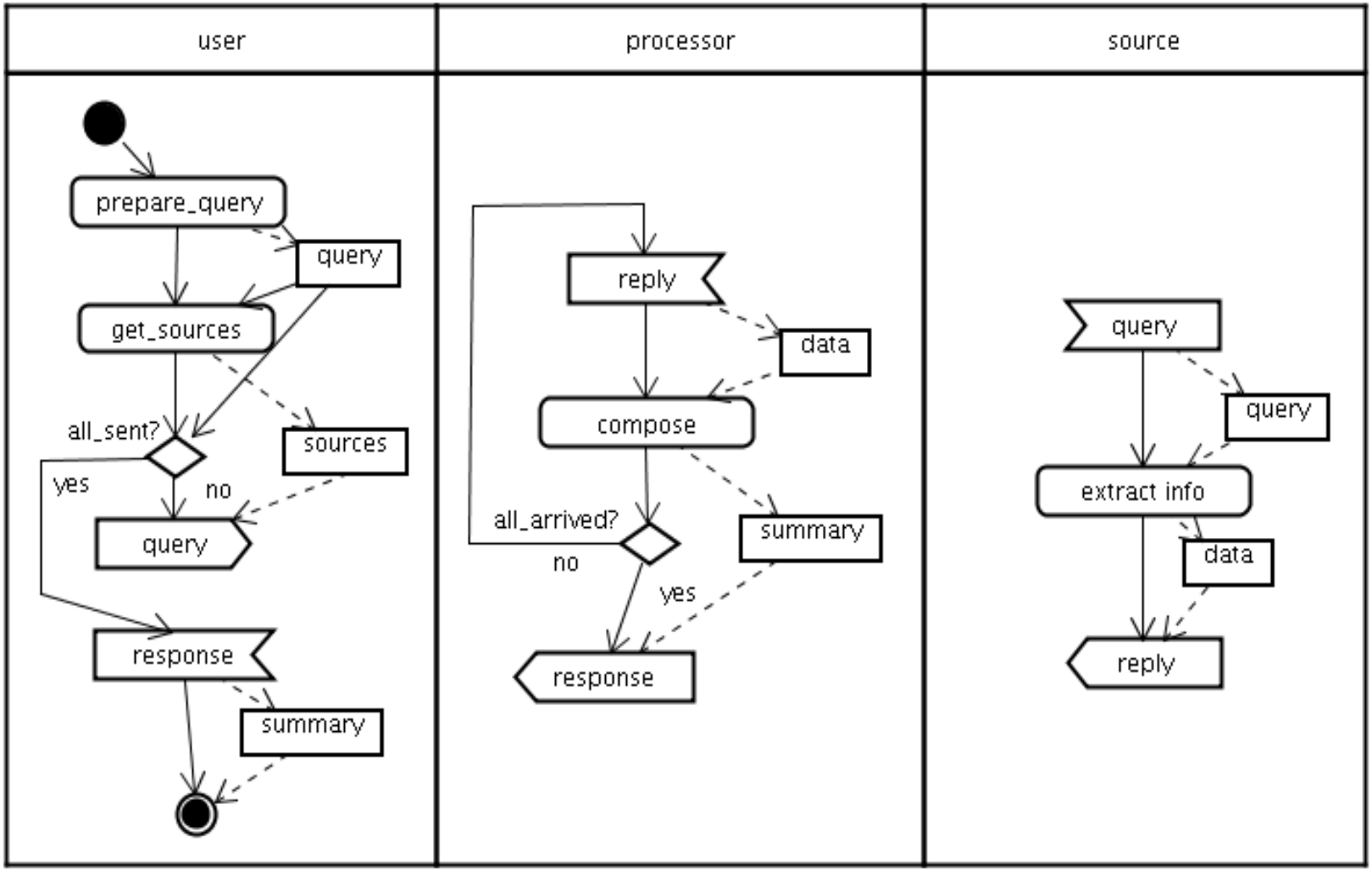
Thanks Barker

A quick calculation



Thanks Barker

Distributed fan-in



LCC fan-in

Message reception

Message sending

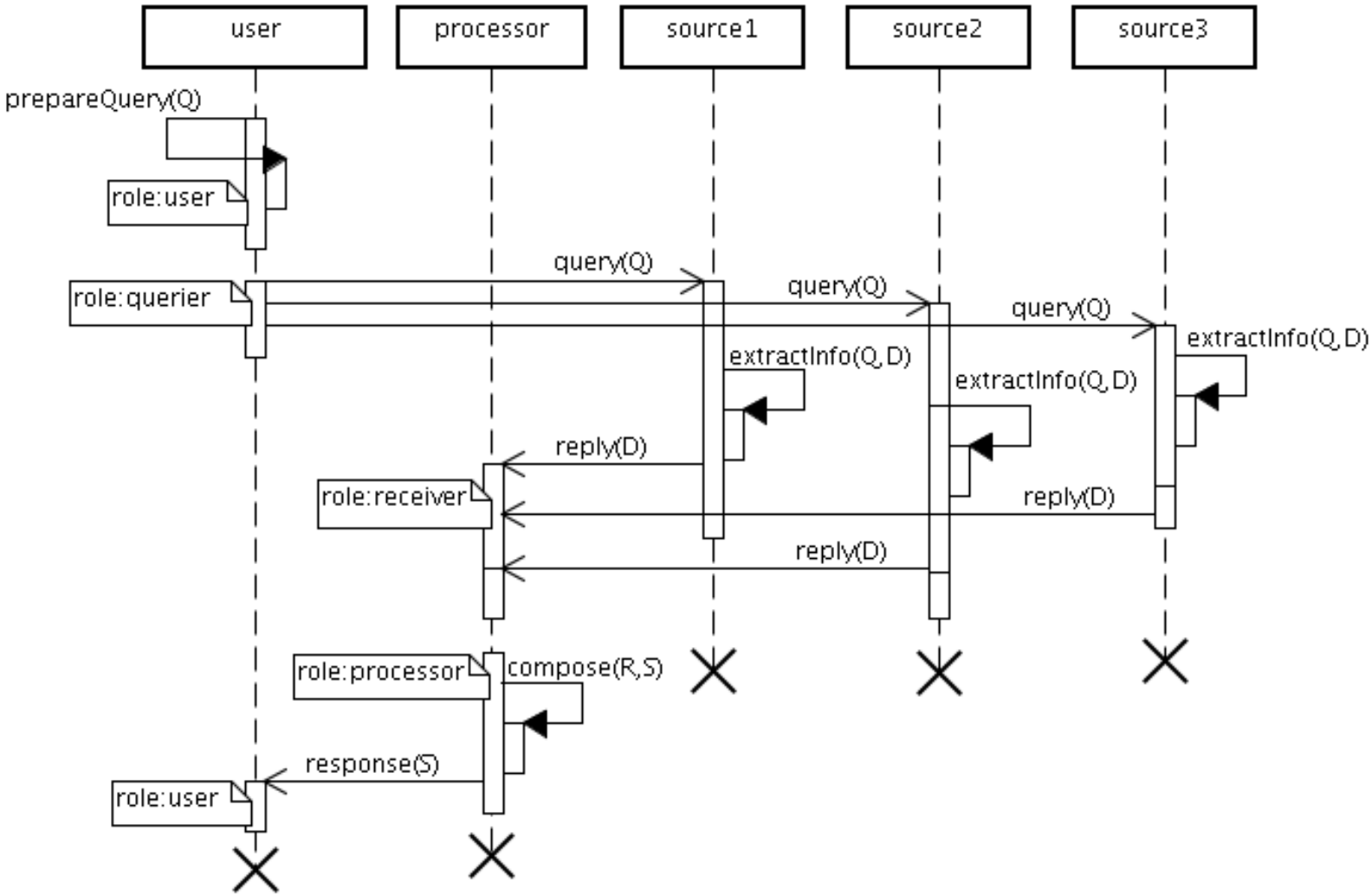
constraint

```
a(user,U)::
 null←prepare_query(Q)and getPeers("source",Ps)
 then a(querier(Q,Ps),U)
 then response(S)⇐a(processor,P)
```

```
a(querier(Q,Ps),U)::
 null←Ps=[]
 or
 query(Q)⇒a(source,P)←Ps=[P|Pt]
 then a(querier(Q,Pt),U)
```

```
a(source,S)::
 query(Q)⇐a(querier,U)
  then
 reply(D)⇒a(processor,P)←extract_info(Q,D)
```

Run example


user
processor
source1
source2
source3
prepareQuery(Q)
role:user
query(Q)
role:querier
query(Q)
query(Q)
extractInfo(Q,D)
extractInfo(Q,D)
extractInfo(Q,D)
reply(D)
role:receiver
reply(D)
reply(D)
role:processor
compose(R,S)
response(S)
role:user

Some concluding remarks

- Service choreography
  - Global view, contractual agreement, optimal data transfer
  - Important in data-centric workflows
- Robustness
  - No single point of failure
- OpenKnowledge and LCC
  - Compact, executable choreography language
  - Corresponding enactment framework