Multi-agent and Semantic Web Systems: Coordination

Fiona McNeill

School of Informatics

21st March 2013

Overview

1. Agents & MAS

2. Architectures
   - Network Architectures
   - Middle Agents

3. From services to agents

4. Interaction Models

5. Summary

Multi-Agents Systems (MAS)

- Distributed system which incorporates independent agents.
- Collective action ⇒ solve problems outside capacities of individuals.
- Focus is on properties that emerge from cooperation (vs. capabilities of individual agents)
- (Some aspects of) coordination achieved dynamically at run-time

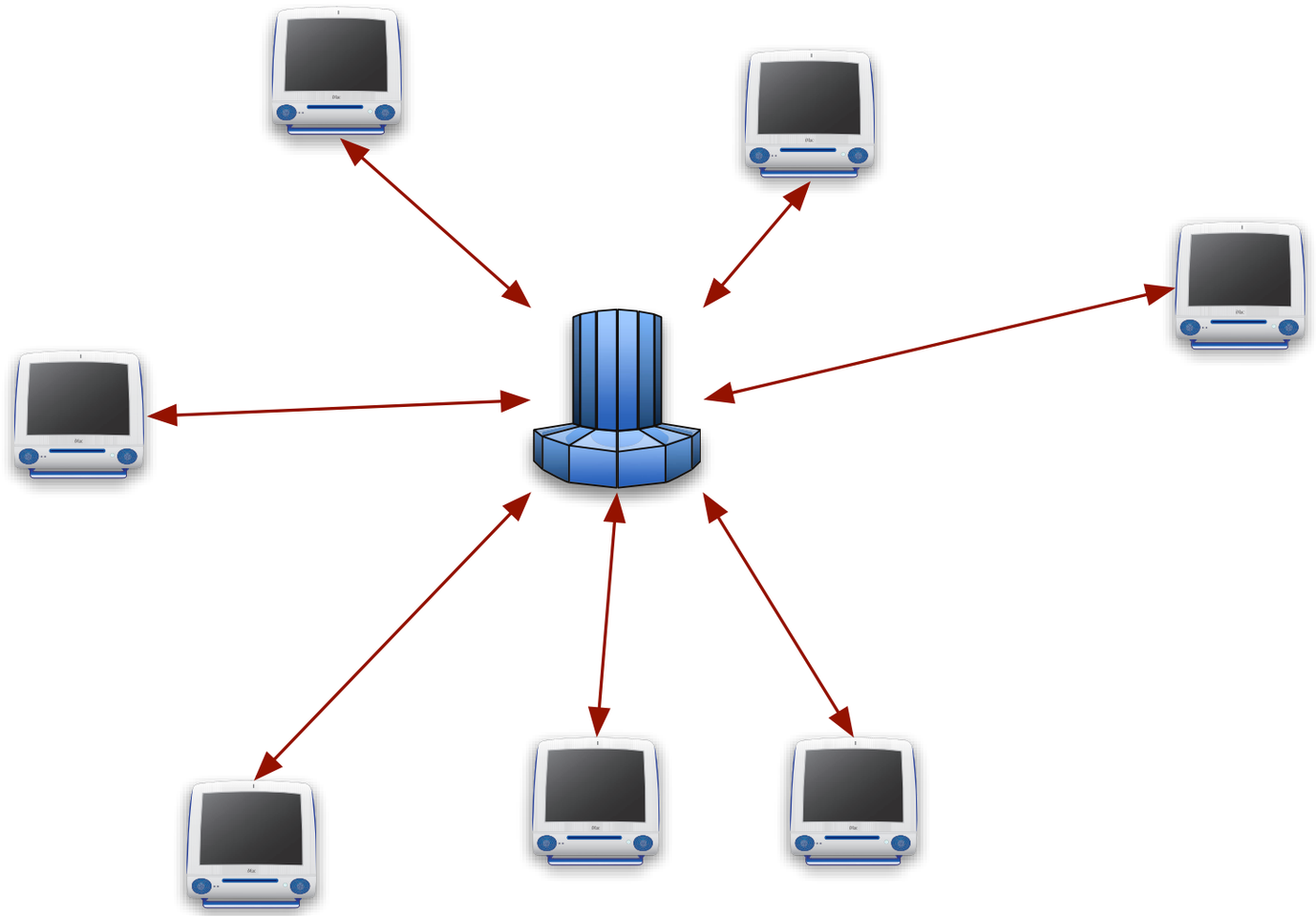
Multi Agent Architectures

- MAS can have be centralised or decentralised (peer-to-peer/P2P).
- So-called middle-agent will play role of coordinator in a centralised architecture.
- Increasing interest in achieving coordination in P2P systems.

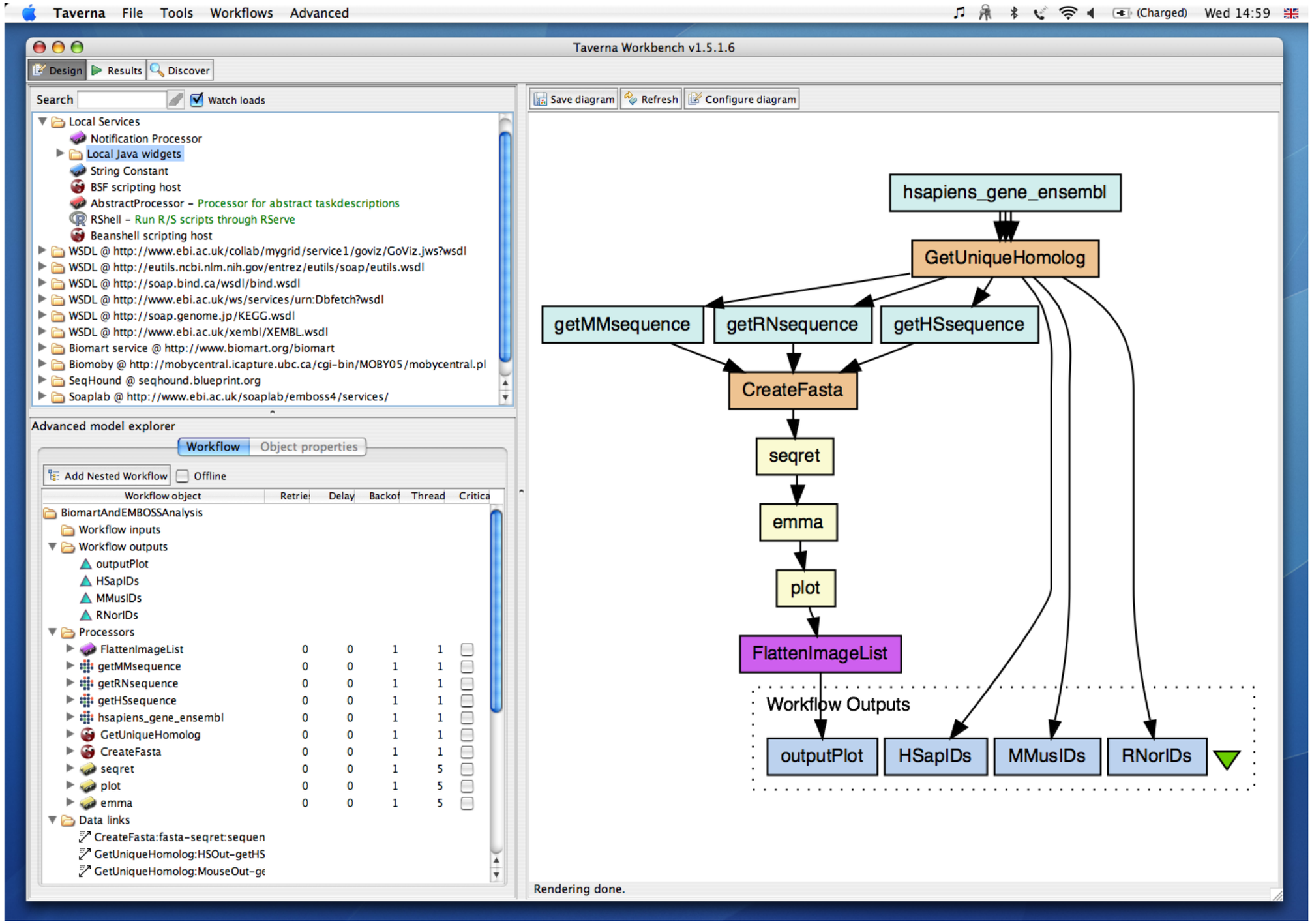
Network Architectures

- So far, mainly assumed some kind of centralised client/server architecture.
- Workflow systems are centralised:
  - workflow manager orchestrates the components services;
  - although data flow is conceptually via the services, in practise, goes via manager.
- But Service Oriented Architectures can equally well be decentralised

Centralised: Coordination via Central Controller



Workflow Example



Coordination via Middle Agents

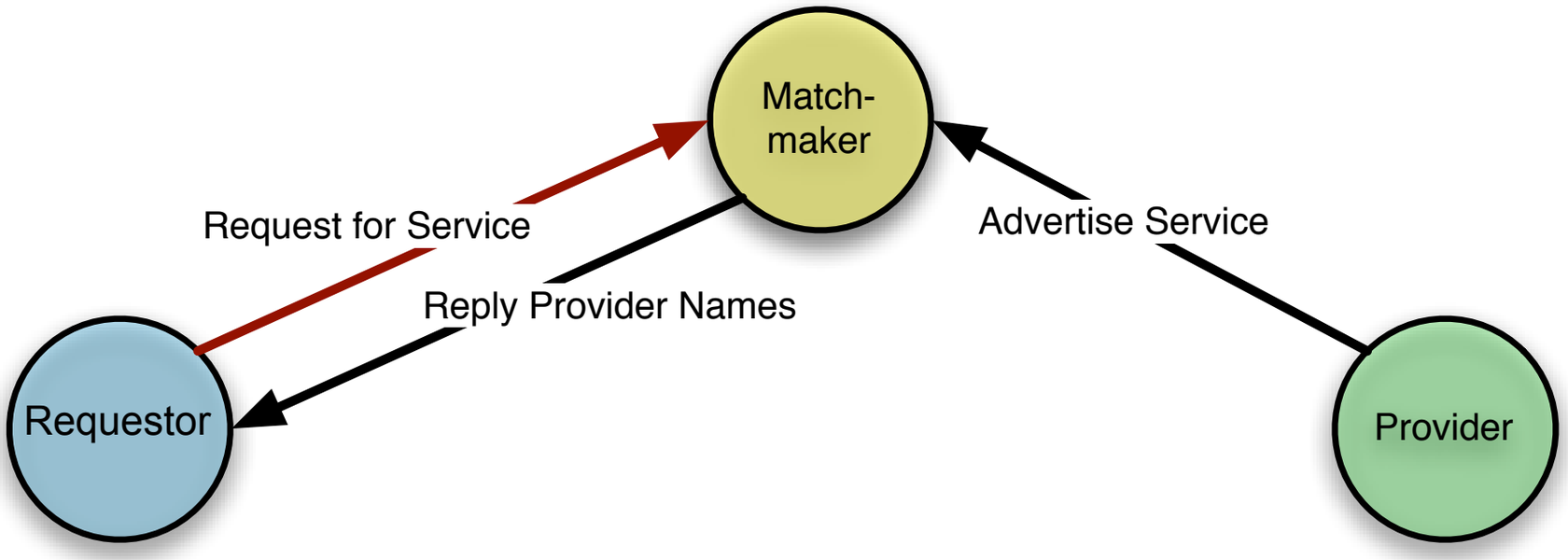
- Middle-agents:
  - specialised agent
  - assists in locating service providers
  - connects service providers with service requesters

- Two important types of middle-agent:

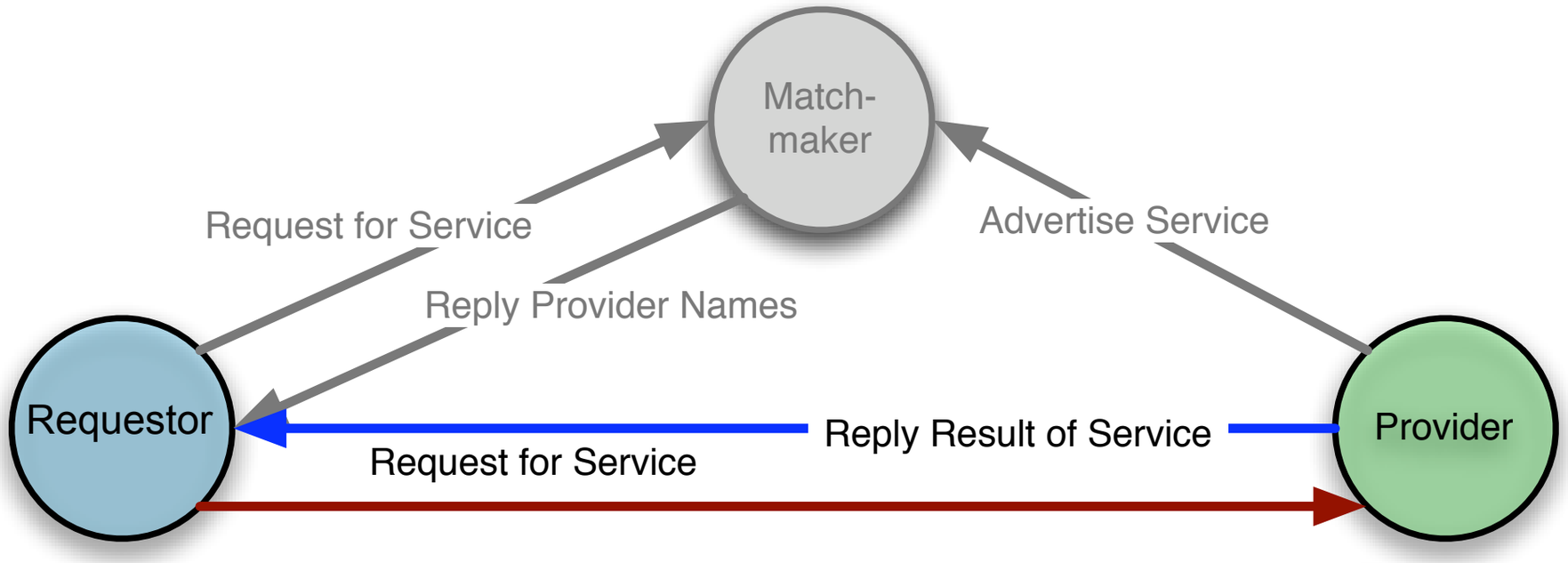
  Matchmaker: receives advertisements and matches with requests.
  Broker: like matchmaker, but also processes the requests.

Cf. `http://www.cs.cmu.edu/~softagents/middle.html`

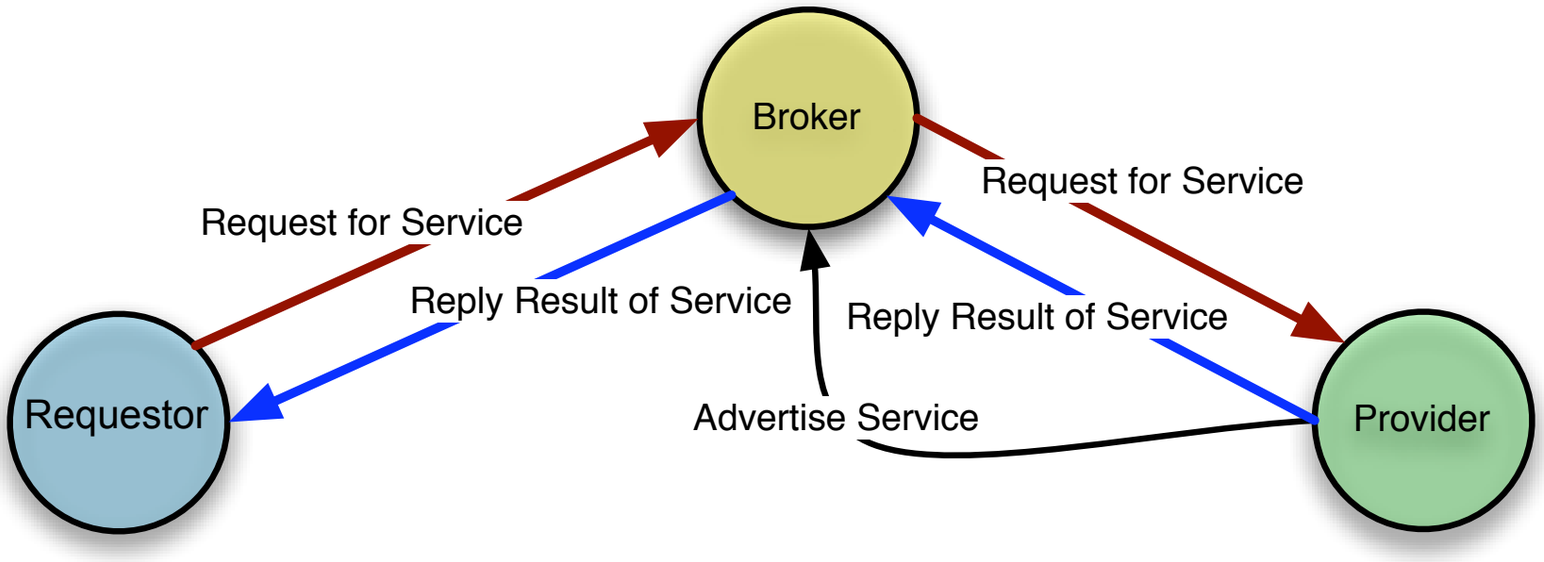
Service Matchmaking



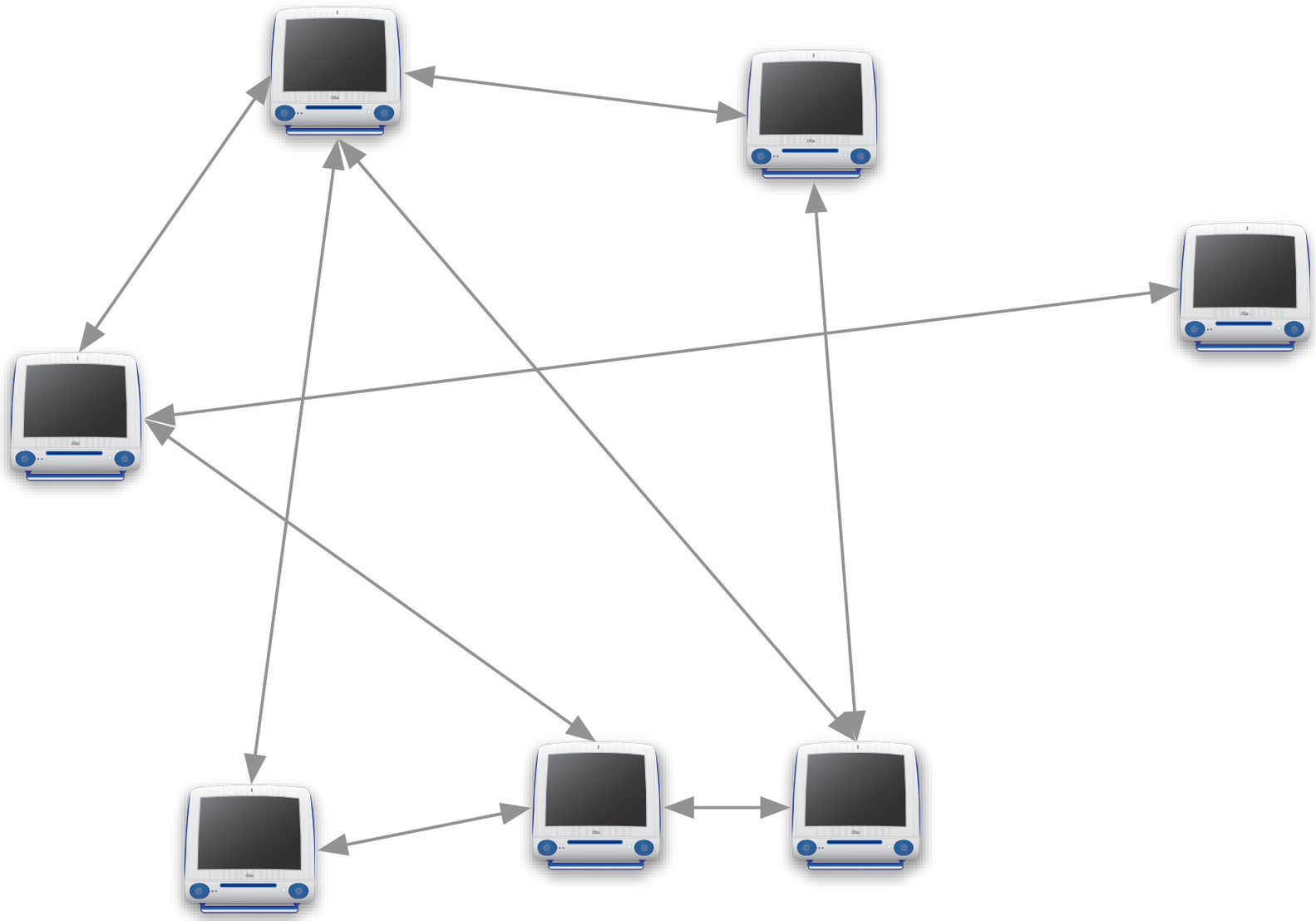
Service Matchmaking



Service Brokering



Decentralised: Peer-to-Peer



From services to agents

- Services provide decentralisation, interoperability, and encapsulation of state

- Traditional service composition requires **service orchestration**
  - In our examples so far: centralised workflow
  - One process co-ordinates execution and data flow
  - Similar to client-server model, despite notion of protocol

- **Peer-to-Peer** architectures are different
  - Components take initiative to participate
  - No central point of control

From services to agents

- Service *choreography* provides more decentralisation
  - Open interaction protocol specifications (no standard language)
  - Semantic description of constraints to be satisfied by peers
  - Peers can subscribe to protocols they can satisfy
  - Discovery may or may not be enabled by centralised service
- This assumes autonomous, self-directed action by the peers
- ... and brings us to the notion of autonomous *agents*

Peer-to-Peer Architectures

P2P is...

a self-organizing system of equal, autonomous entities (peers) [which] aims for the shared usage of distributed resources in a networked environment avoiding central services.

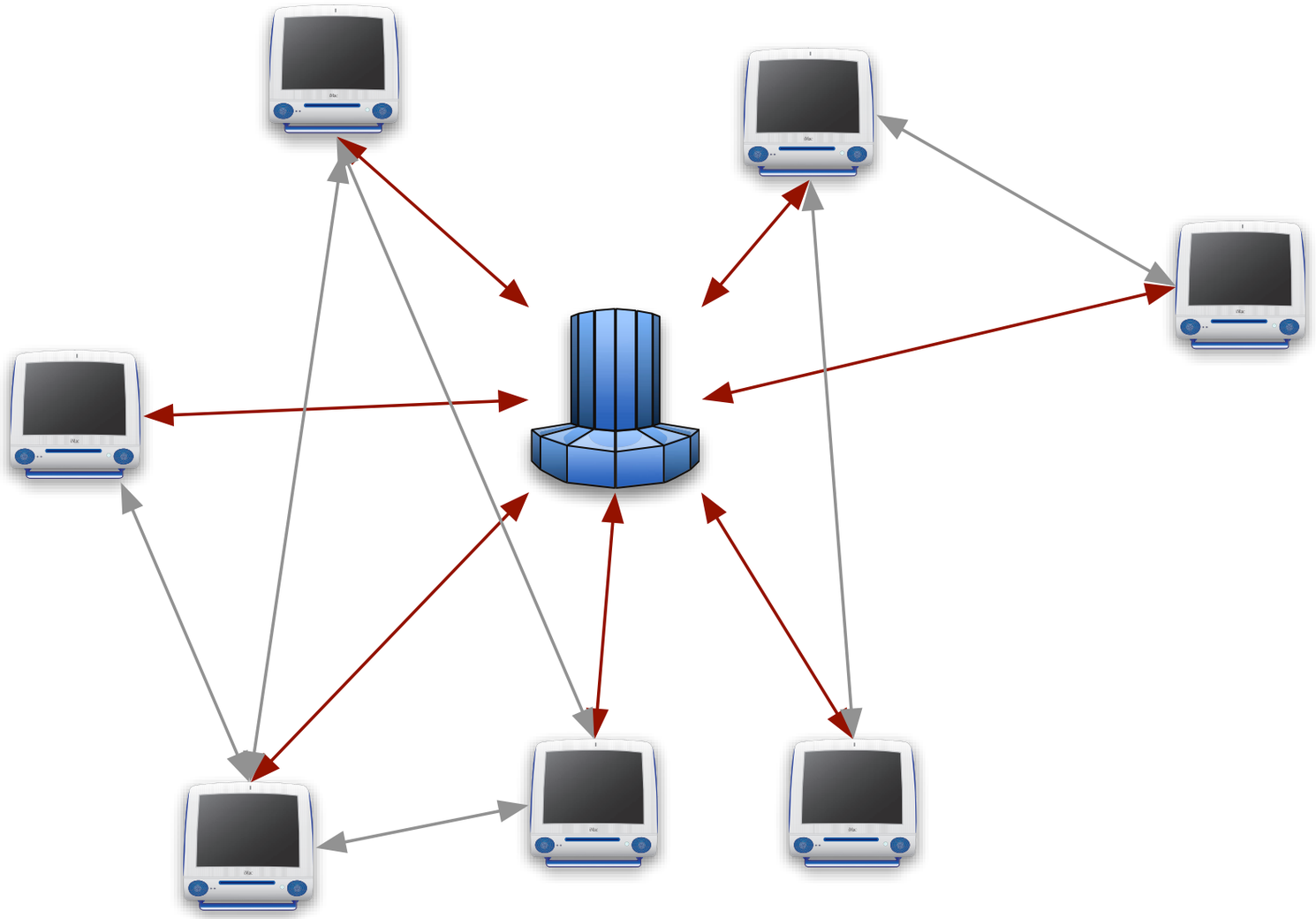
Peer-to-Peer Architectures

P2P is...

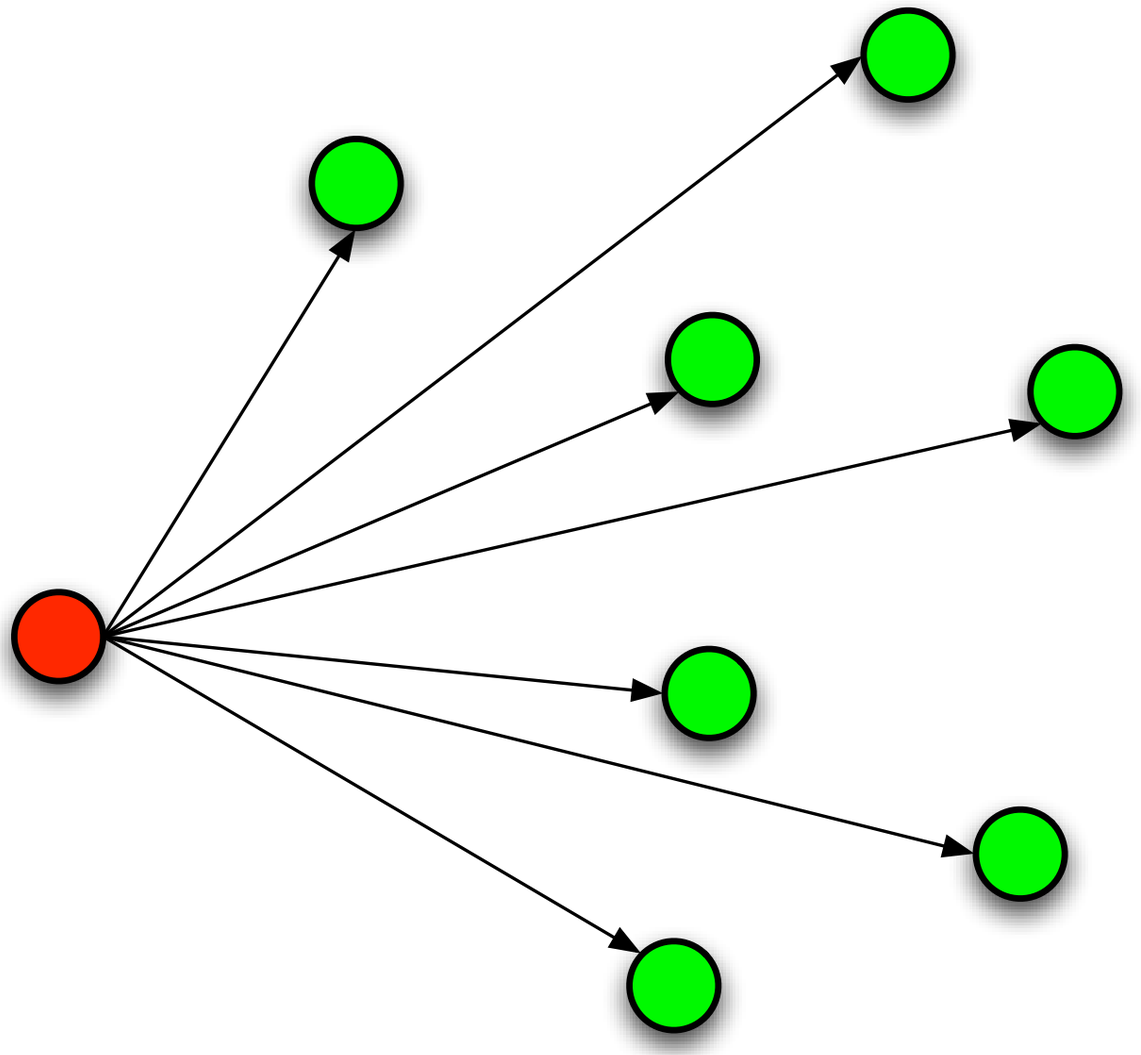
a self-organizing system of equal, autonomous entities (peers) [which] aims for the shared usage of distributed resources in a networked environment avoiding central services.

- Peers interact directly with each other, usually without central coordination.
- Each peer has autonomy over its own resources.
- Within a set of peers, each uses resources provided by other peers.
- Peers can act as both clients and servers; i.e., no intrinsic asymmetry of role.
- Performance considerations may dictate some centralised elements in P2P systems — leads to hybrid P2P systems.

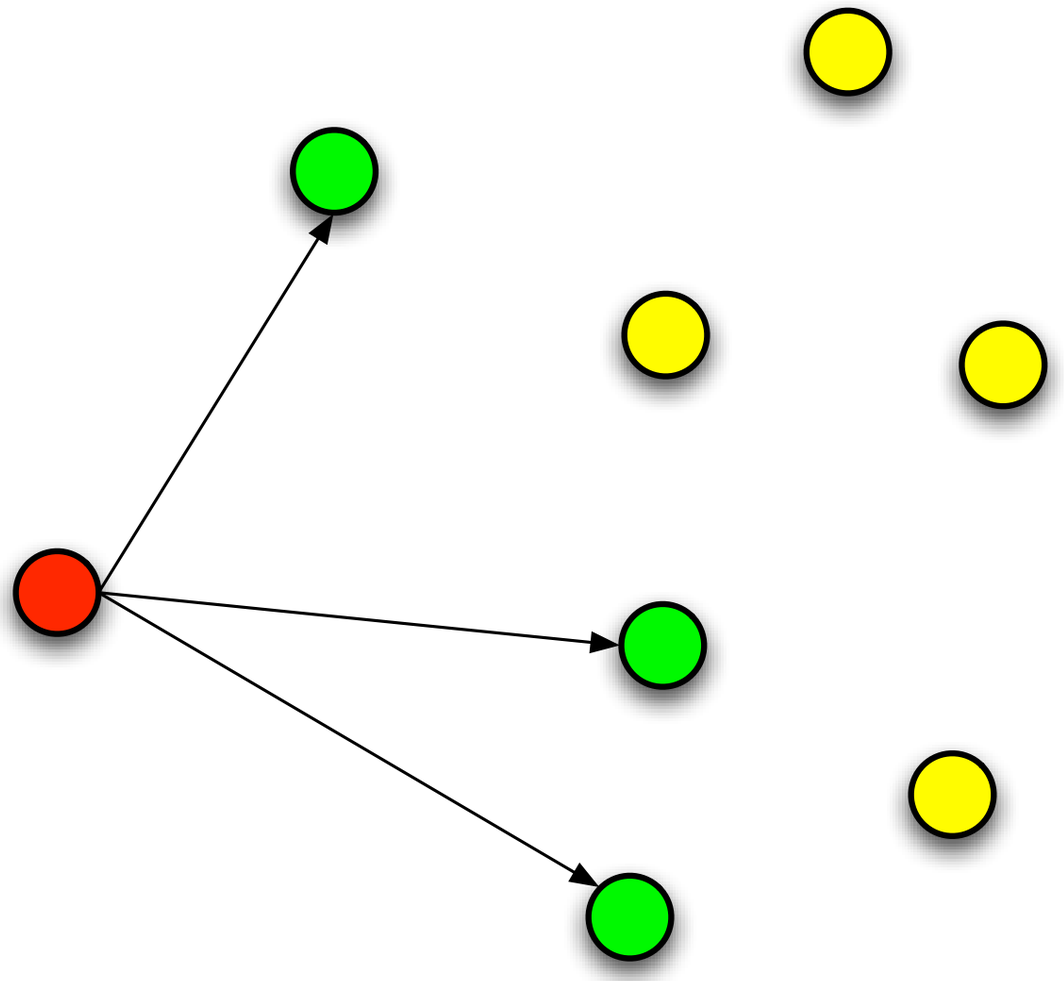
Hybrid: Peers and Super-Peers



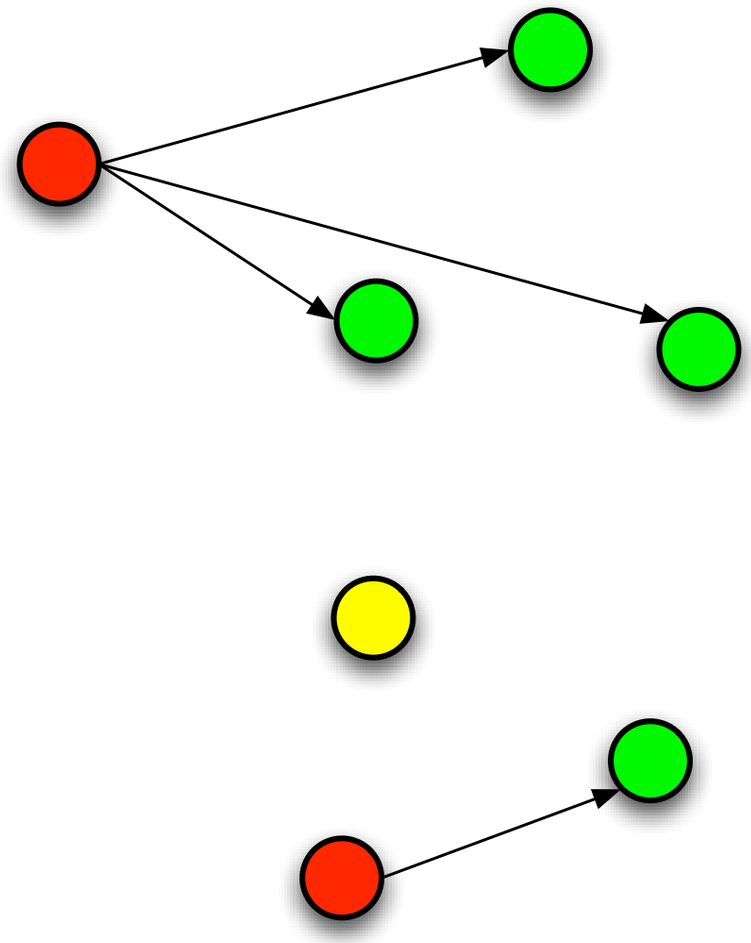
Broadcast



Multicast, 1



Multicast, 2



Coordination Problem

- Simplistic WS Model:
  - "one-shot" interactions:
  - client sends a request message to a single service operation and receives a response message.

- In practise, we want to allow more complex kinds of interaction:
  - multiple operations,
  - multiple messages exchanged,
  - messages sequenced in a particular order,
  - multiple parties involved in the interaction.

- How do we ensure that such interactions are
  - coordinated?
  - correct?
  - robust to failures?

Interaction Protocols

- ACLs define syntax, semantics and performative aspect of individual messages.

- How do messages become organised into a coherent conversation?

- Interaction protocols govern the exchange of series of messages.

- Define patterns of admissible sequences.

- Often formalised by UML sequence diagrams in FIPA

Protocol Design

1. Describe the interaction capabilities of the agents.

2. Clarify the type of messages involved.

3. Explain possible message sequences.

4. Specify the (internal) states of the agents.

5. Encode the protocol in a diagram (e.g., in AgentUML).

Contract Net Protocol, I

- Forms part of larger framework: cooperative distributive problem solving.
- Focusses on task allocation among communicating agents.
- Primary concerns: distributed control, achieving reliability, and avoiding bottlenecks.

1. Manager announces one or more tasks.
2. Agents bid to perform them.
3. Manager uses an evaluation function to rank the bids (e.g., choose the cheapest)
4. Uses the agents' private knowledge for task allocation.

Contract Net Protocol, 2

1. Agent recognises it has a problem it needs help with.

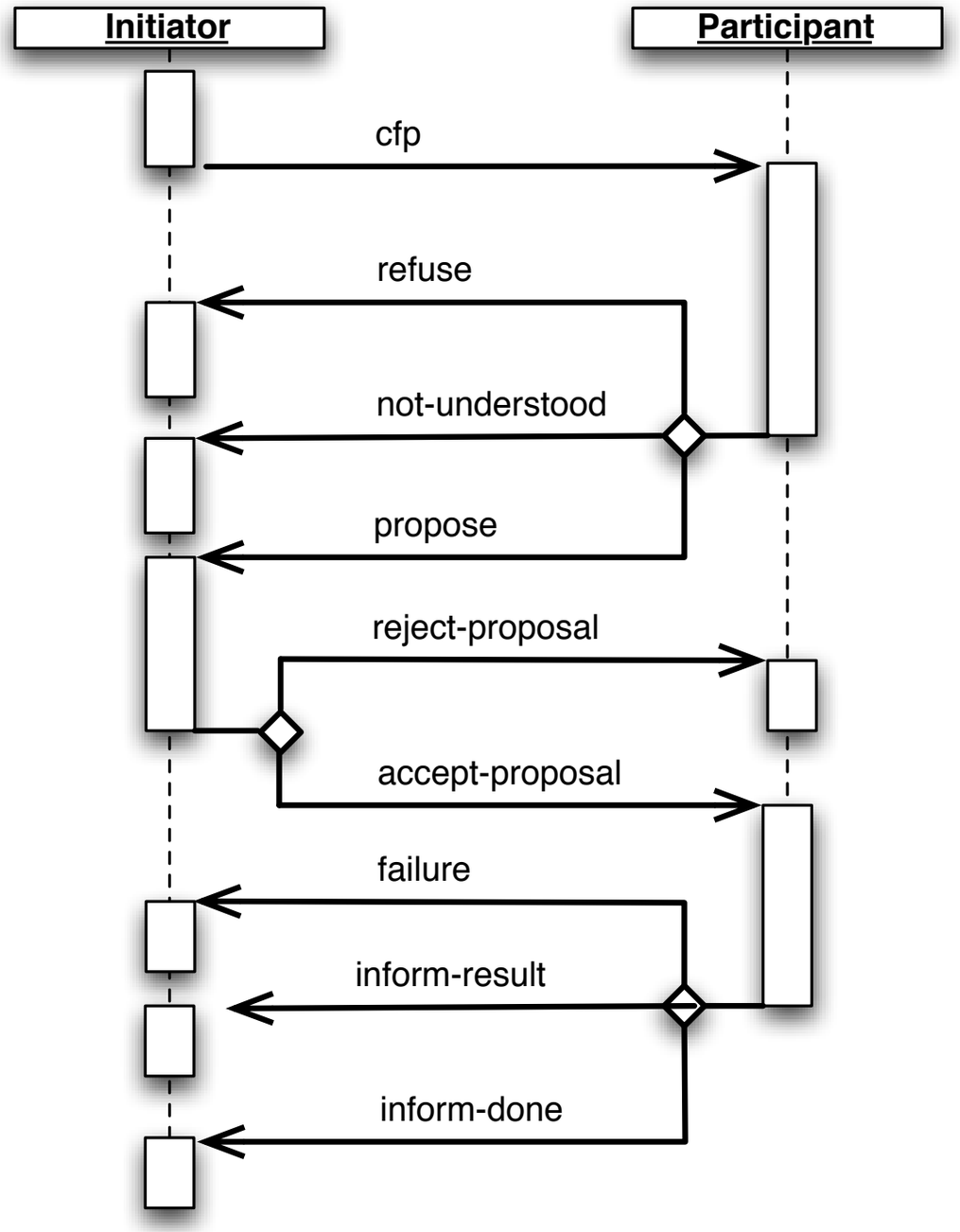
2. Has a goal, and either cannot or prefers not to achieve it in isolation (own capability, deadline, etc)

3. The collection of nodes is the "contract net"

4. Each node on the network can, at different times or for different tasks, be a manager or a contractor

5. When a node gets a composite task (or can't solve its present task), it
   - breaks it into subtasks (if possible)
   - announces them (acting as a manager),
   - receives bids from potential contractors, then
   - awards the job

Contract Net Protocol



Limitations of Contract Net

- Before sub-problems can be distributed, problem decomposition needs to be performed.

- Communication produces overhead and can be slow.

- Problems must have right granularity (rather coarse).

- Recognition stage (agent realises that it needs help with a problem) is not explicitly covered.

Agent Roles

- Protocols give us a way of specifying a class of legal interactions between agents.

- However, we often want to have higher-level ways of describing agent behaviour.

- Key notion: role that is assigned to an agent. Roles determine rights, duties and opportunities.

- The role assumed by an agent limits its possible actions.

- Example roles in interaction: seller, buyer, auctioneer
  - Seller must own goods before submitting them for sale.
  - Buyer may submit bids if credit standing is good.
  - Auctioneer may offer goods and accept bids.

Electronic Institutions, I

- Agent counterpart of human organisations.
- Specifies norms and rules to govern interaction.
- Conversation protocols are grouped into scenes.
- Agents participate in scenes by virtue of a role — can play different roles in different scenes.

- Example scenes (for auction):
  - admit buyers
  - admit sellers
  - carry out auction
  - settlement (i.e., paying for goods)
- Scenes play a similar role to policies; determine who can do what, when.

Electronic Institutions, 2

- Scenes are connected into a performative structure;
- latter governs how agents can move from one scene to another.
  - E.g., `admit buyer` precedes `auction`, `auction` precedes `settlement`
- Norms govern transitions between scenes.
  - E.g., a buyer agent that wins a bid is obliged to pay for the good.

Interaction Models

- Higher level structures govern more abstract aspects of interaction.

- Policy languages and electronic institutions: two ways of representing rights and obligations of agents.

- Key notions from electronic institutions:

  scene: bounded space in which a group of agents interact on a shared task.

  role: fixed until the end of scene; determines which parts of a protocol an agent can execute; multiple agents can take on the same role.

Example Interaction Models

$a(police, P)$ ::
$water\_level(Location) \Rightarrow a(sensor, S) \leftarrow drop\_off(Entity) \wedge suitable\_loc(Entity, Location)$ then
$water\_level(Location, Level) \Leftarrow a(sensor, S)$ then
$drop\_off(Entity, Location) \Rightarrow a(firefighter, F) \leftarrow safe\_level(Level).$

$a(sensor, S)$ ::
$water\_level(Location) \Leftarrow a(police, P)$ then
$water\_level(Location, Level) \Rightarrow a(police, P) \leftarrow detect(Location, Level).$

$a(firefighter, F)$ ::
$drop\_off(Entity, Location) \leftarrow drop\_off(Entity, Location) \Leftarrow a(police, P).$

Key elements:

- role denomination: *a(roleName,Agent)*
- *role denomination* :: *role definition*
- Message passing:
  - *to* $\Longrightarrow$ *from*, or *from* $\Longleftarrow$ *to*
- Preconditions: $\longleftarrow$

Summary: Architecture

- Centralised vs. peer-to-peer architectures.
- Middle-agent acts as coordinator in a centralised architecture.
  - Matchmaker vs. broker

Summary: Protocols

- Protocols determine possible messages and their sequencing.
- Contract Net protocol is one of the oldest and most widely used.

Reading

- Walton, Chap 6.
- Wooldridge, esp Chaps 1, 2, 8.
- Passin, Chap 9.