Unit 4. Cooperative and Distributed Solving of Problems

Based on “An Introduction to MultiAgent Systems” http://www.csc.liv.ac.uk/~mjw/pubs/imas and chapters 3 & 4 from “Multiagent Systems” by Gerald Weiss.

Working together

- Why and *how* agents can work together?
- It depends on what type of agents are involved:
  - *benevolent agents*
  - *self-interested agents*

Self-interested agents

- If agents represent individuals or organizations then we can not assume that they are going to show a benevolent behaviour.
- Agents try to reach their self interests, maybe making some kind of competition with each other.
- This can lead to a *conflict* outcome.

Benevolent agents

- If we, as software programmers, design all the system then we can force agents to help each other when they are asked for.
- So, we can get a *benevolent* behaviour.
- *Benevolent behaviour simplifies the system design a lot!*
- The solving of problems in benevolent systems is coined **Cooperative and Distributed Problem Solving** (CDPS) (CDPS implies two factors *Coherency* + *Competence*)

Cooperative and Distributed Problem Solving (CDPS)

- What is a bird-view of whatever *problem solving*?
  - Search from a *starting* state …
  - to a *final* state that satisfies a given goal.
- There is no central control or global storage (no one agent has enough information to reach the given goal on its own).
- So, control and data are inherently distributed.
- Given that communication is slower than computation tasks, this approach gets as more good results as the involved agents …
  - Weakly coupled.
  - Efficient protocols.
  - Overall problem can be seen as composed of nodular subproblems.
  - Gross grain problems.

Features and consequences of CPDS

- Try to avoid that an agent can behave as a bottleneck:
  - Distributing Data.
  - Distributing Control.
  - But, a complete organised behaviour is very hard to guarantee (due that maybe neither agent has the global view of the solving process)

- There are two main ways to cooperatively solving problems:
  - ***Sharing tasks***:
    task's components are distributed among the available agents in the system.
  - ***Sharing results***:
    intermediate results are shared between available agents .

Sharing tasks

- Four main steps:

1. Decompose a task into subtasks.
2. ***Place subtasks in the more adequate agents.***
3. Communicate partial solutions.
4. Combine partial solutions into the global (or subglobal) solution.

- How step 2 can be done? There are several techniques:
  - Matchmaking: make a central repository of the features of each agent to be contracted with the current needs.
  - Brokering: make a repository that accept all the tasks from a given type and then to assign a capable agent (from a stable set of agents suitable to perform that task) to each task.
  - **Contract Net**.

Contract Net

- It is a protocol focused on distributing tasks among agents.
- The assignation of a task to an agent is seen as a *negotiation contract*.
- "Protocol" specifies not just its form, but the communication content.
- Bidirectional information exchange is a natural way to exchange control mechanisms.

- The total set of agents (lets called, from now on, just *nodes*) is called the **contract net**
- Each node of the net can sometimes show a role of *manager* or a role of *contractable*
- When a node has a decomposable task (or for whatever reason, this node can not solve the task by its own) , it splits this task into subtasks and announces these subtasks (acting now as a manager), it receives offers from the potential available contractable nodes, review the offers, select the best offers and assign winners for each task (or subtask) by making contracts.

Contract Net

Main steps:

1. Announce.
2. Offer reception.
3. Assign subtasks.
4. Solving subtasks.

Previous step

In this previous stage the agent recognizes that it has to deal with a task that cannot be solved by itself because

- it has not enough capabilities.
- maybe the deadline for the solution of the tasks is very close and it can not be solved so fast or maybe the quality of its solution is far from an acceptable quality.

How to decompose the task? No general solution.

Announce

- In this step an agent with a task sends an advertisement that involves an **specification** of the task to solve.
- That **specification** must include:
  - A description of the task to solve (maybe in a executable form)
  - Some constraints (e.g. deadlines, degree of quality)
  - Meta-task information (e.g. “offers must be sent by …”)
- The advertisement is *broadcasted*

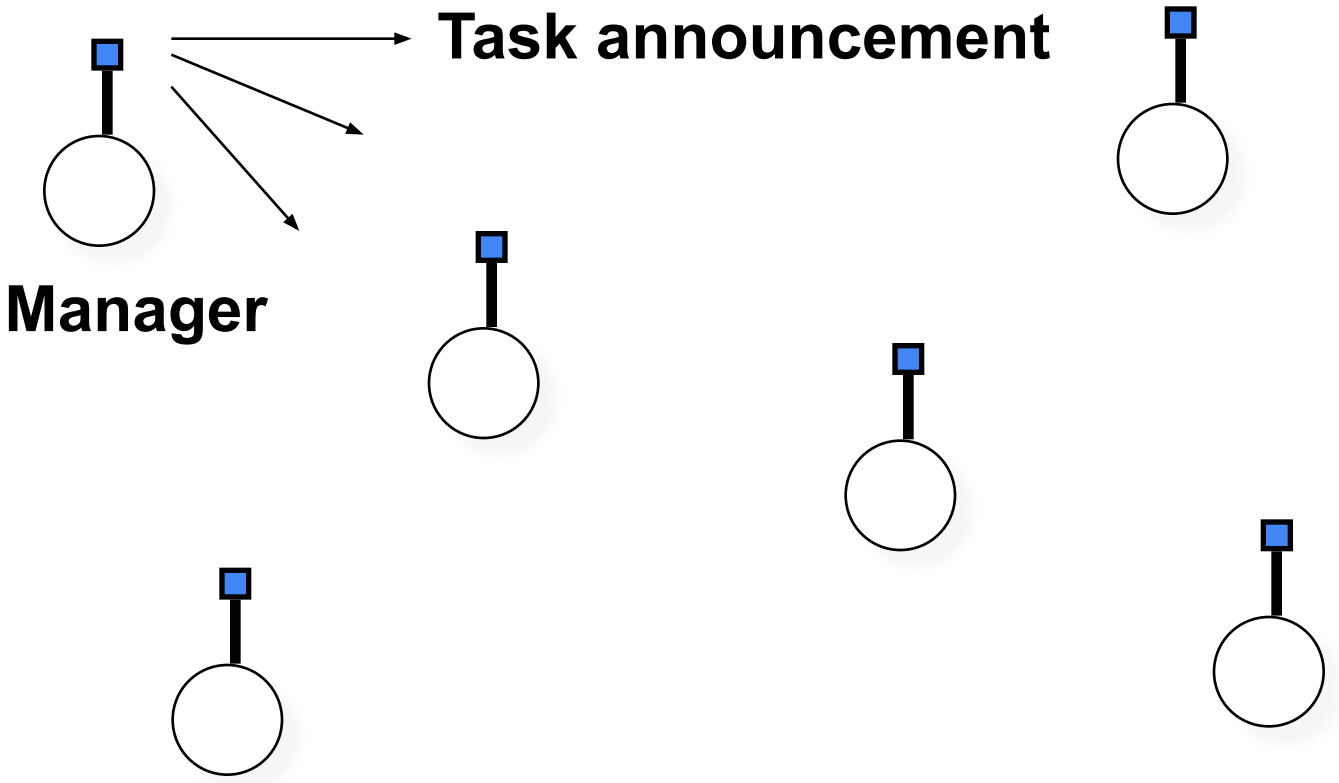
Offers

- Agents receiving the advertisement must decide if they want to make an **offer** for this task
- Factors:
  - The agent has to figure out if it is capable of doing it (given its current state and compromises)
  - And maybe, the agent has to figure out the quality constraints and the price that it will be received for doing that task.
- Send an offer if they want to.

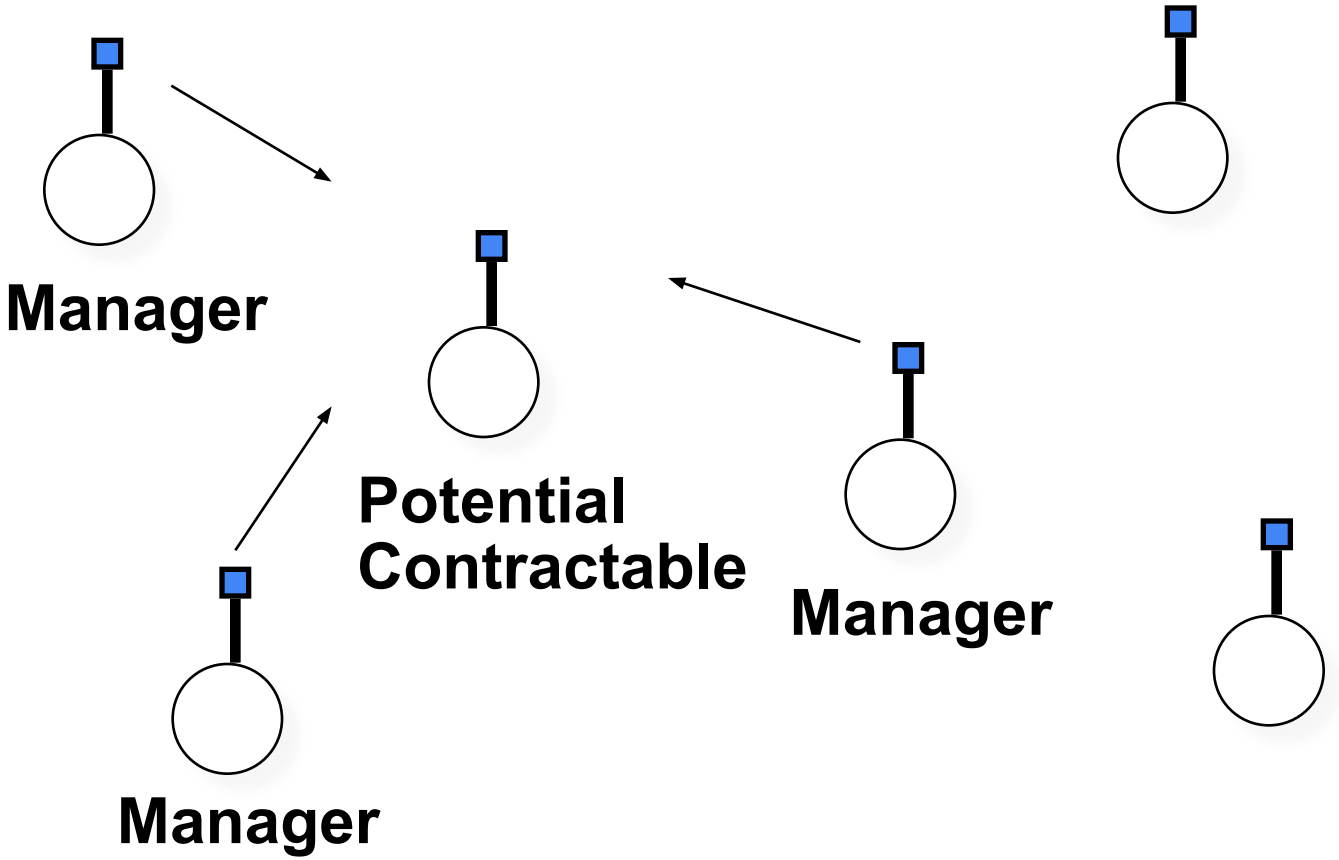
Choosing and solving

- The agent that sends the advertisement has to choose among the received offers and to decide to whom the contract is assigned to, which one is the winner (Just to one or maybe to several ones?)
- The decision has to be sent to all agents that sent an offer (winners & losers)
- The *contracted* agent (winner) starts solving the task (and then, maybe this agent will send periodic intermediate informs or just one final inform?)
- Even more, the contracted agent could, sub-contract the task to other agents (maybe using another instance of the contract net protocol)

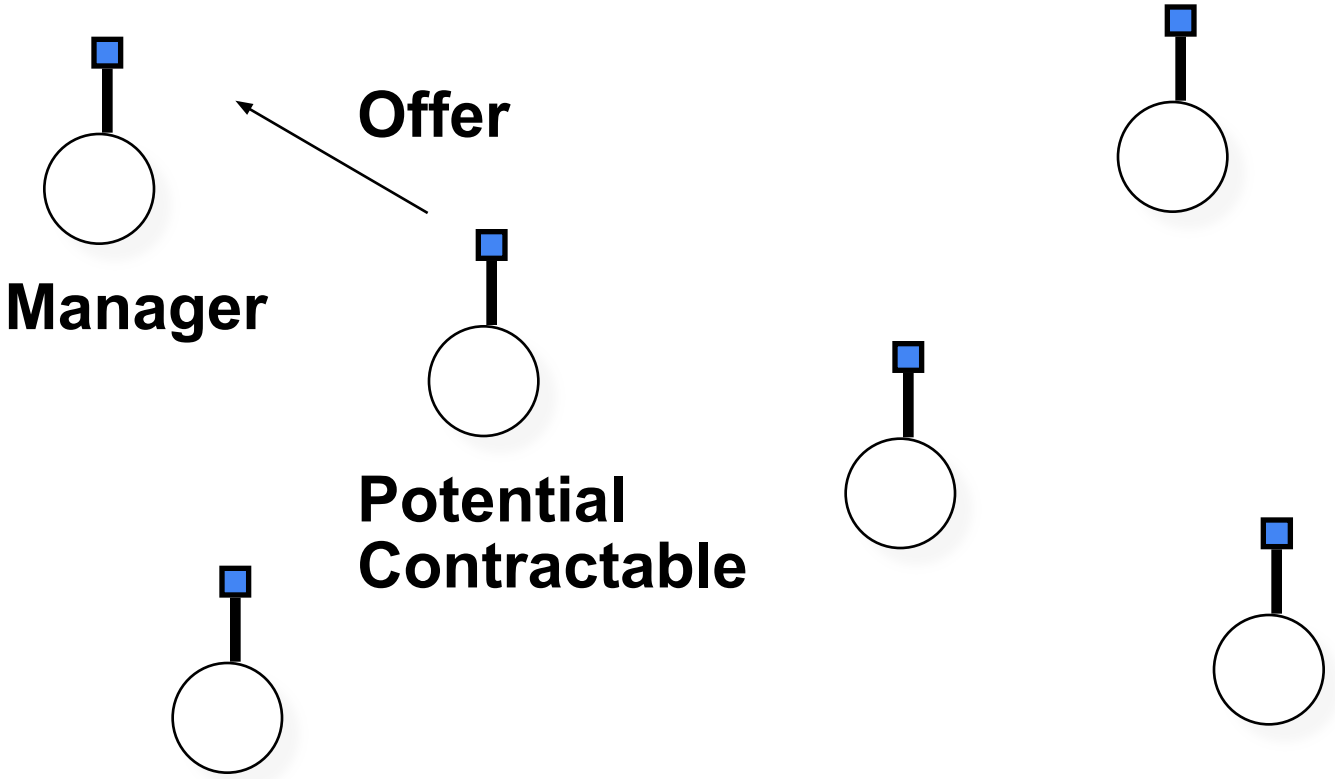
Task announcement



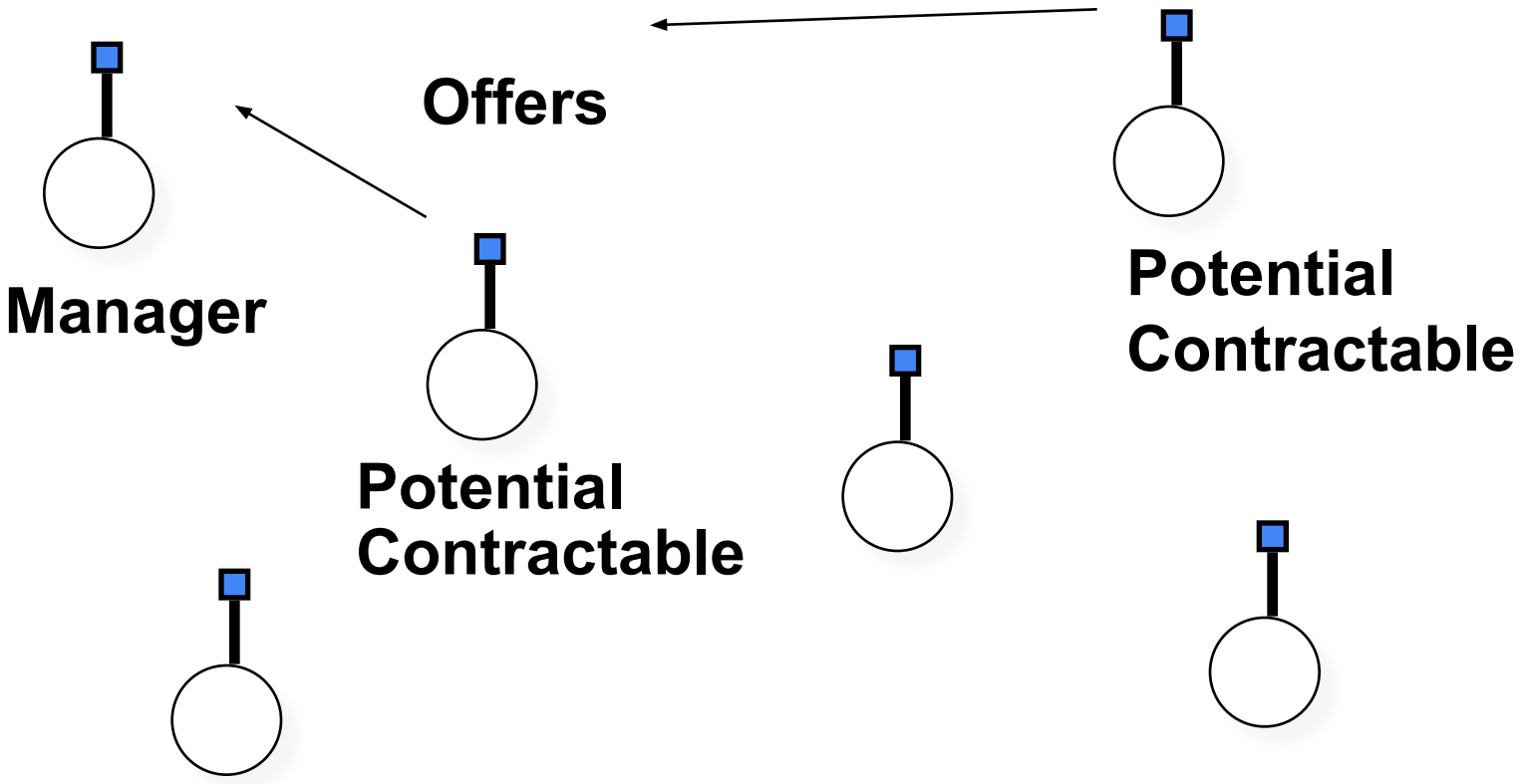
Idle nodes receive announces



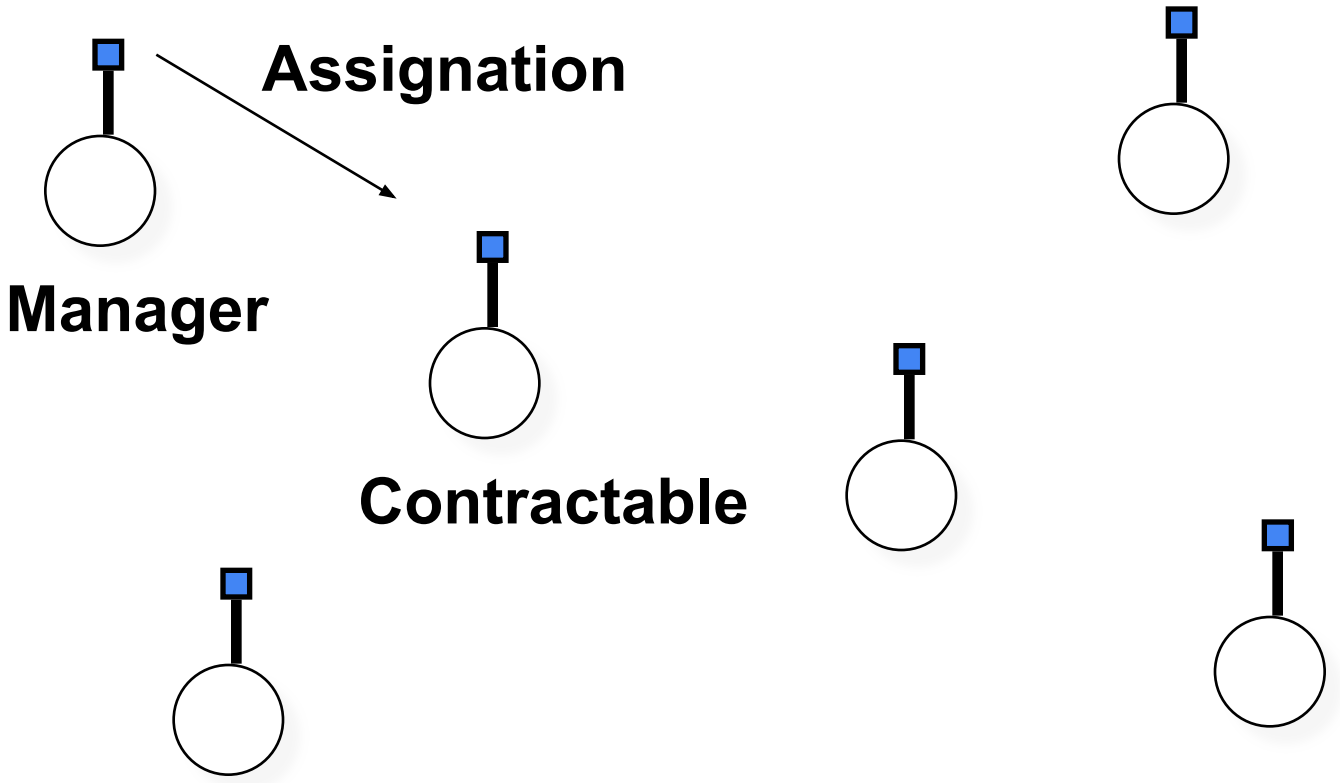
Node sends an offer



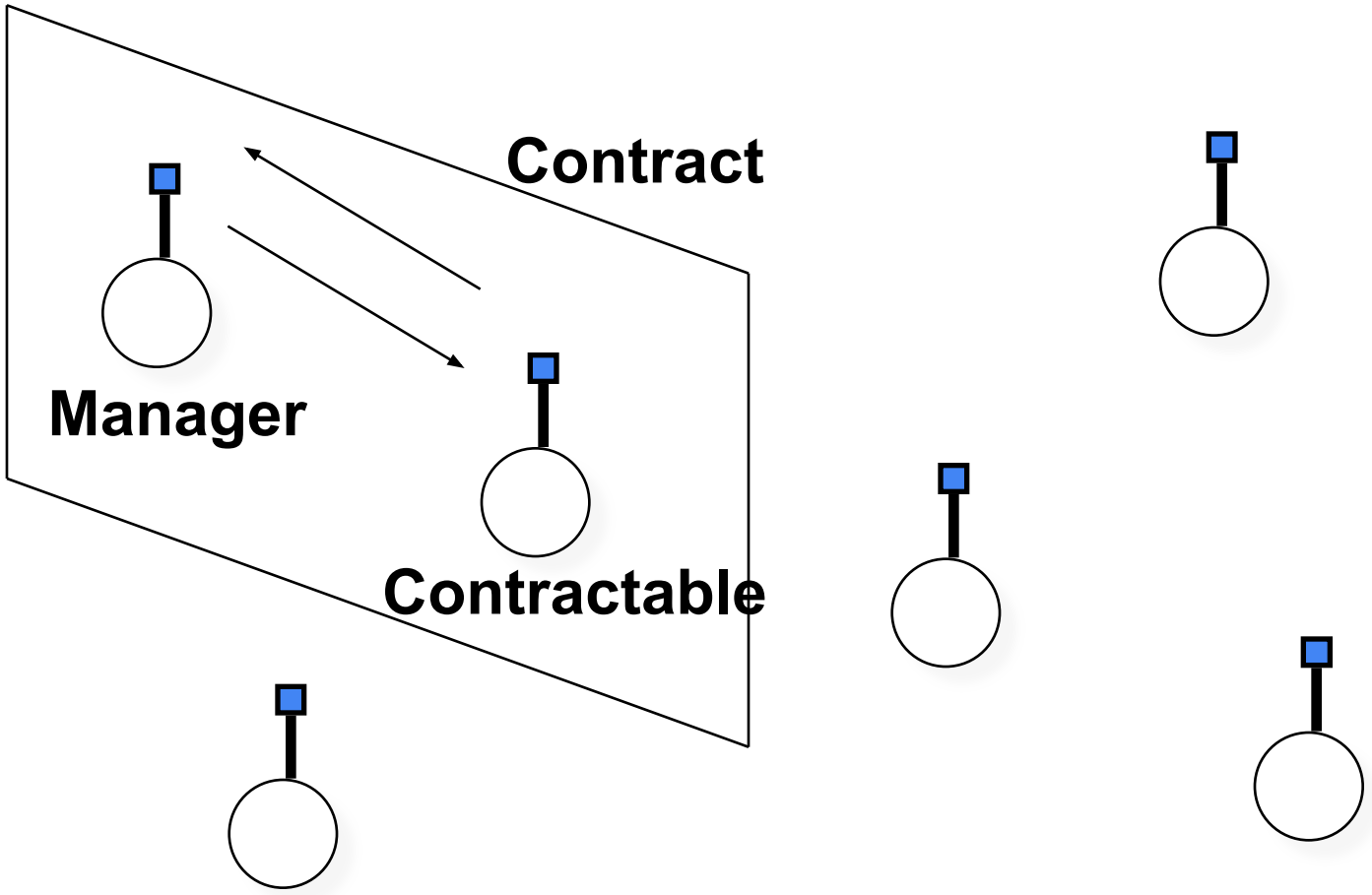
Manager receives offers



Manager assigns winner



Contract is set



Several issues when programming the Contract Net

- How to…
  - …specify *tasks*?
  - …specify *quality of service*?
  - …clear out among competitive offers?
  - …evaluate among offers based on multiple criteria?

Domain specific evaluation

- The manager evaluate received offers by using the available specific tools for the domain of the task.
- Usage of a common ontology can help in understanding offers from different domains, and to write adequate offers to the manager …

Potential efficiency enhancements

- Direct addressing — when there is not a need for a broadcast.
- Direct contracts — when the manager just knows who is the appropriate agent.
- Simple request-response mechanism — to make a simple transaction without the need to sign contracts.
- Available nodes messages — Initiative of the negotiation is flipped.
- Greedy: To store task results to avoid solving them again.
- Envision: Assign the task using future contracts.

Types of message involved in Contract Net

- Task announcement
- Offer
- Assignation
- Periodical informs (work in progress)
- Final inform
- Finalization message if the manager wants to break down the contract

Syntax of the messages

- **Task announcement**:
  - Legibility specification.
  - Any abstract description on the task to be solved.
  - How the offers must be fill in.
  - Deadline for admit offers.

To: *
From: 25
Type: Task Announcement
Contract: 43–6
Eligibility Specification: Must-Have FFTBOX
Task Abstraction:
Task Type Fourier Transform
Number-Points 1024
Node Name 25
Position LAT 64N LONG 10W
Bid Specification: Completion-Time
Expiration Time: 10678594324

Example

Another example: Announcement of a signal task

To: *
From: 25
Type: Task Announcement
Contract: 22–3–1
Eligibility Specification:
Must-Have SENSOR
Must-Have Position Area A
Task Abstraction:
Task Type Signal
Position LAT 47N LONG 17E
Area Name A Specification (…)
Bid Specification: Position Lat Long
Every Sensor Name Type
Expiration Time: 106785943246

Offers (to send to the Manager)

To: 25
From: 42
Type: BID
Contract: 22–3–1
Node Abstraction:
LAT 47N LONG 17E
Sensor Name S1 Type S
Sensor Name S2 Type S
Sensor Name T1 Type T

Example

Assignation (from Manager to contractable)

To: 42
From: 25
Type: AWARD
Contract: 22–3–1
Task Specification:
Sensor Name S1 Type S
Sensor Name S2 Type S

Example

What if nobody answers?

- Repeat the announcement in a cyclic way.
- Contractable nodes send their capabilities.
- To mix previous ways.
- To revise down the quality of the requested response.
- Try to decompose the task into an alternative way.

Contract Net main features

- Bidirectional information exchange
- Local evaluation
- Mutual selection (contractables choose among the announced tasks, managers choose among received offers):
  - Example: contractables choose closest managers, managers use the distribution of available sensors to choose the contractables set by its location and type of provided data from its sensors.

When to apply contract net?

Applications involving:

- Task hierarchy.
- Data support several abstract levels of usage.
- The adequate selection of contractables is important.
- Big subtasks (and, then, it is worthy to dedicate an extra effort to wisely distribute tasks among other nodes)
- Main issues are in the distribution of control, to gain fiability and to avoid bottlenecks.

Limitations

- There exists stages not easy to deal with:
  - Problem decomposition
  - Partial solution combination
- Overload
- Alternative methods to deal with the broadcasted announcement of tasks, with the evaluation of tasks and offers

FIPA performatives for the Contract Net

- cfp (call for proposals) – For task announcements.
- Propose, refuse – To get ready to send or not to send an offer.
- accept, reject – To accept or deny an offer.
- inform, failure – To send the outcome of the task or the failure in its execution.

Event diagram