Escola Tècnica Superior d'Enginyeria
Universitat Rovira i Virgili

[DEIM]

Lecture 4: Agent Communication

Multi-Agent Systems

Universitat Rovira i Virgili

MESIIA – Master's Degree in Computer Security Engineering and Artificial Intelligence
MAI – Master's Degree in Artificial Intelligence

Multi-Agent Systems course

- First part of the course: Agents
  - Introduction to agent technology
  - Agent architectures
  - Agent properties
  - Agent types

Multi-Agent Systems course

- Second part of the course: Multi-Agent Systems
  - Agent communication
  - Coordination
  - Negotiation
  - Distributed planning
  - Coalition formation
  - Application domains

Outline

1. Definition of a Multi-Agent System
2. Distributed problem solving
3. Agent communication
    1. Blackboard systems
    2. Message passing
        1. FIPA standards
        2. Basic communication protocols
        3. Contract Net

1. Definition of a Multi-Agent System

- It is as a collection of deliberative, autonomous, collaborative agents, that may communicate and cooperate with each other to solve complex distributed problems
- They are specially useful to solve problems with distributed data/expertise

1. Definition of a Multi-Agent System - Benefits

- Modularity
  - Each agent is specialised in the solution of a particular kind of problems (leading also to re-usability)
  - The complexity of the construction of agents is reduced
  - The process of solving a complex problem is reduced to solving easier sub-problems

1. Definition of a Multi-Agent System - Benefits

- Efficiency
  - Problems can be solved more quickly, due to the inherent concurrency/parallelism
  - Different agents are working at the same time in different parts of a problem
    - These sub-problems can be independent or (slightly) dependent
    - Share partial results
    - Coordinate the use of shared resources

1. Definition of a Multi-Agent System - Benefits

- Reliability
  - Avoid single point of failure in centralised systems
  - We can have redundancy
    - Different agents of the same type
    - Different agents that can do a certain task
  - If an individual agent fails, the other agents can take its work and re-distribute it dynamically

1. Definition of a Multi-Agent System - Benefits

- Flexibility
  - Agents can be created/deleted dynamically, depending on the amount of work to be done, the available resources, etc.
  - Agents can dynamically generate subtasks and look for helping agents
  - Agents with different skills may dynamically form teams/coalitions to work together

2. Distributed problem solving

- Agents work together to solve problems that require collective effort
- Requires:
  - Coherence
    - Need to *want* to work together
  - Competence
    - Need to know *how* to work together
  - Coordination
    - Need to follow a common plan

2. Distributed problem solving - Steps

1. Task Decomposition
2. Task Allocation
3. Task Accomplishment
4. Result Synthesis

2. Distributed problem solving – Task decomposition

- Divide a complex problem into a set of tasks (e.g., a set of sub-problems)
- Decompose (*recursively*, if necessary) large tasks into sub-tasks that can be tackled by different agents
- Tasks can be totally independent or can have some relationships
- This decomposition may be done by the *user*, by a *central coordinator agent* or may be negotiated at run time by *all the agents* of the system

2. Distributed problem solving – Task allocation

- Assign the sub-problems to different agents
- Easy case: there is only one agent capable of solving each sub-problem
  - No flexibility
- Interesting case: the same sub-problem can be solved by different kinds of agents
  - Dynamic mechanisms of run-time task allocation (e.g., Contract Net)
  - Reassignment if agent fails

2. Distributed problem solving – Task accomplishment

- Each agent solves (on its own) its assigned sub-problems
- This phase can be done without communication if the problems are independent
- In some cases,there may be dependencies between sub-problems
  - Potential *conflicts* may appear

Conflict management

- Examples
  - Two sub-solutions are incompatible
  - Conflicts in the use of shared resources
- Agents have to communicate with each other to solve these situations
- Need to find an agreement

Agent support in task execution

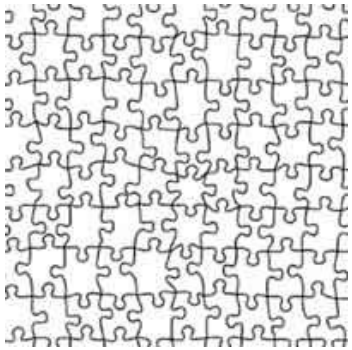
- Task sharing
  - An agent can request the help of other agents to solve a particular task
    - Too complex / expensive for the agent to do individually
    - It can know that other agents have the appropriate knowledge/skills to solve that task
    - It can know that other agents already have to solve that task
  - Problem of *task assignment*
    - Who can I ask for help?
    - How do I know what tasks can other agents do?

Agent support in task execution

- Result sharing
  - Use intermediate results obtained by other agents
    [For example, shortest routes to certain points of the map]
  - Agents can provide intermediate sub-solutions to help other agents in their work
  - That allows a fast recognition of
    - Incorrect solutions
      - An agent, on the basis of its knowledge, can detect an error on the results of other agents
    - Conflictive solutions
      - An agent can detect possible conflicts between its results and sub-solutions of other agents
  - Cooperation/Negotiation to solve these problems

2. Distributed problem solving – Result synthesis

- Put together the results of all agents to find the complete solution
- Who makes it?
- How is it made?
- If each sub-problem has a unique solution, it is a relatively easy step
- Otherwise, there may be need of conflict detection, solution merging, reallocation of subtasks, …



Summary – Questions about the design of a collaborative MAS

- *Who* decomposes the problem?
  - One agent / all agents / user
- *How* is the problem decomposed?
  - 1 level / n levels
  - Static – beginning / Dynamic – execution time
- Who assigns tasks to agents? How is the assignment made?
  - Static – beginning / Dynamic – execution time

Summary – Questions about the design of a collaborative MAS

- What kind of *interactions* can exist between agents when they are solving tasks?
  - Conflict detection
  - Conflict resolution
- How are tasks shared?
- How are results shared?
- Who/how merges the sub-results to get the final result?

3. Agent communication

3. Agent Communication – Why do we need it?

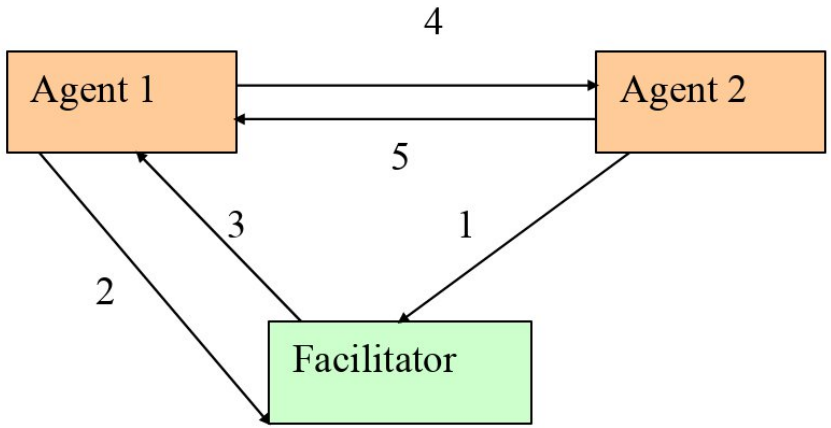
- Multi-agent systems allow distributed problem solving
- This requires the agents to coordinate their actions
- Agent communication facilitates this by allowing individual agents to interact
  - Allows cooperation
  - Allows information sharing

The Sendee – Addressee Link

- Communication can be
  - Point to Point
    - An agent talks directly to another agent
  - Broadcast
    - An agent sends some information to a group of agents
  - Mediated
    - The communication between two agents is mediated by a third party
    - Example: facilitators

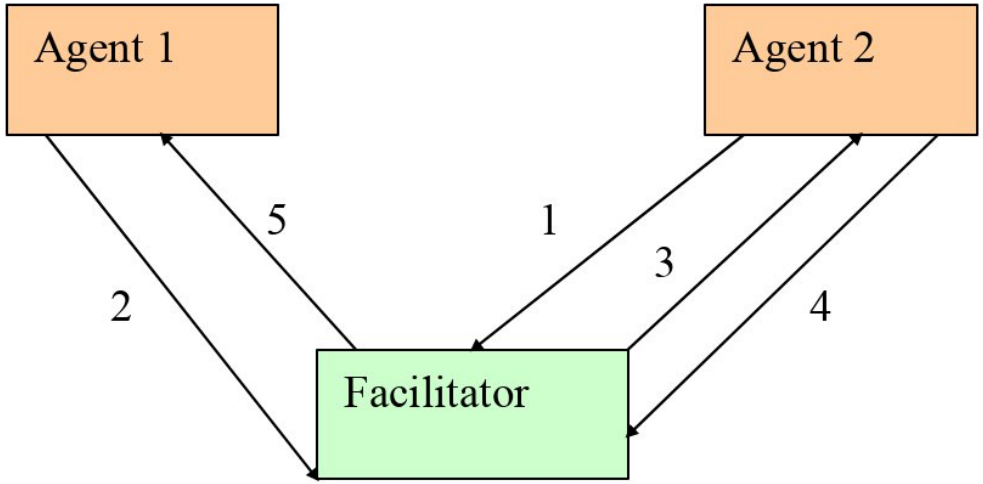
Communication via facilitator

1. *Agent_2* tells the *facilitator* the services it provides
2. *Agent_1* asks to the *facilitator* who can provide a certain service with some conditions
3. The *facilitator* tells *agent_1* that *agent_2* can do that service
4. *Agent_1* requests the service from *agent_2*
5. *Agent_2* sends the answer to *agent_1*



Communication via facilitator

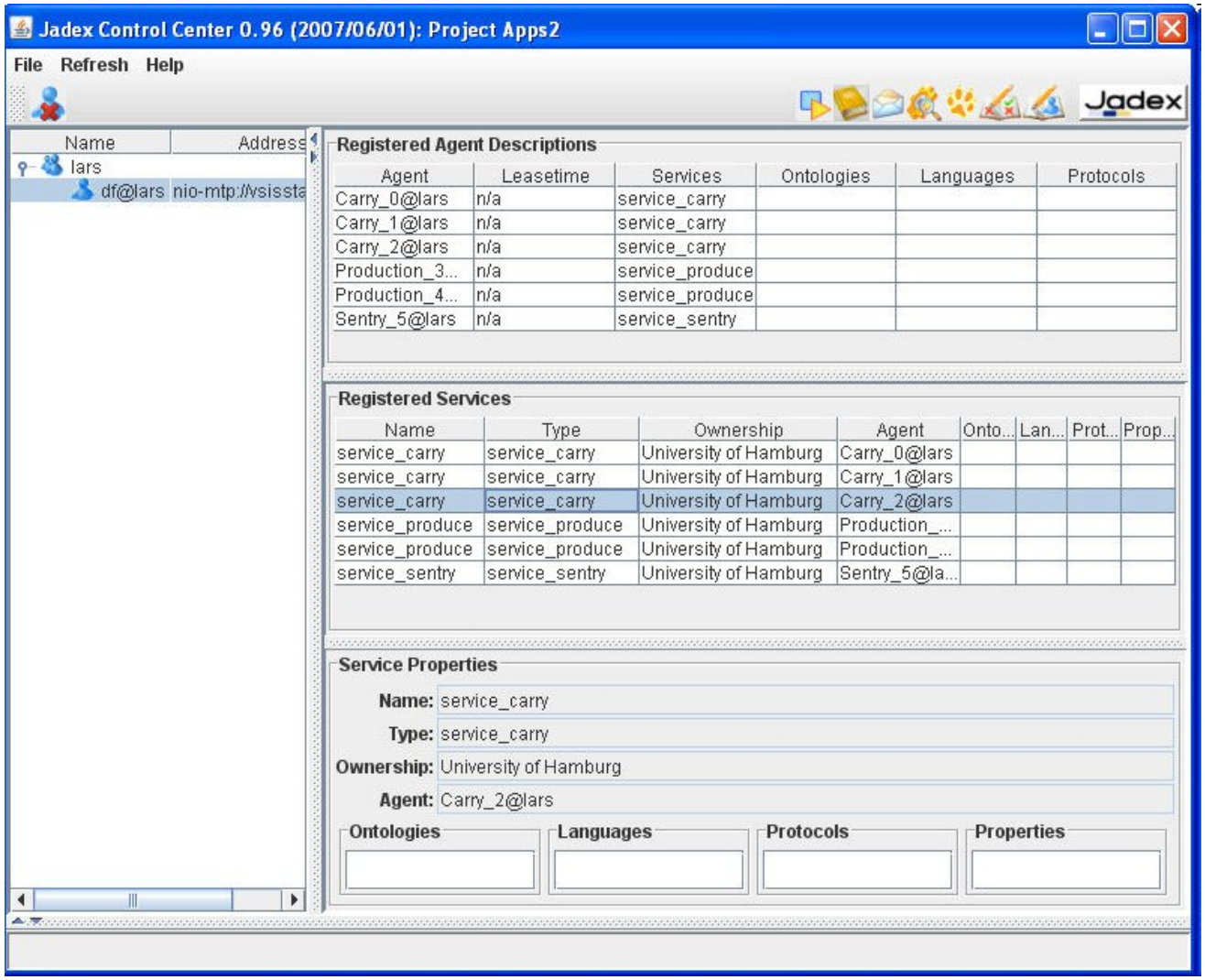
1. *Agent_2* tells the *facilitator* the services it provides
2. *Agent_1* asks to the *facilitator* who can provide a certain service with some conditions
3. The *facilitator* requests the service to *agent_2*
4. *Agent_2* provides the answer
5. The *facilitator* sends the answer to *agent_1*



Locating other agents

- Unless we use some broadcast techniques (e.g., blackboard systems), agents must know the addresses of other agents - possible solutions are
  - Complete internal directory
  - Partial/hierarchical internal directory
  - Mediated (e.g., JADE's DF)

JADE – Directory Facilitator



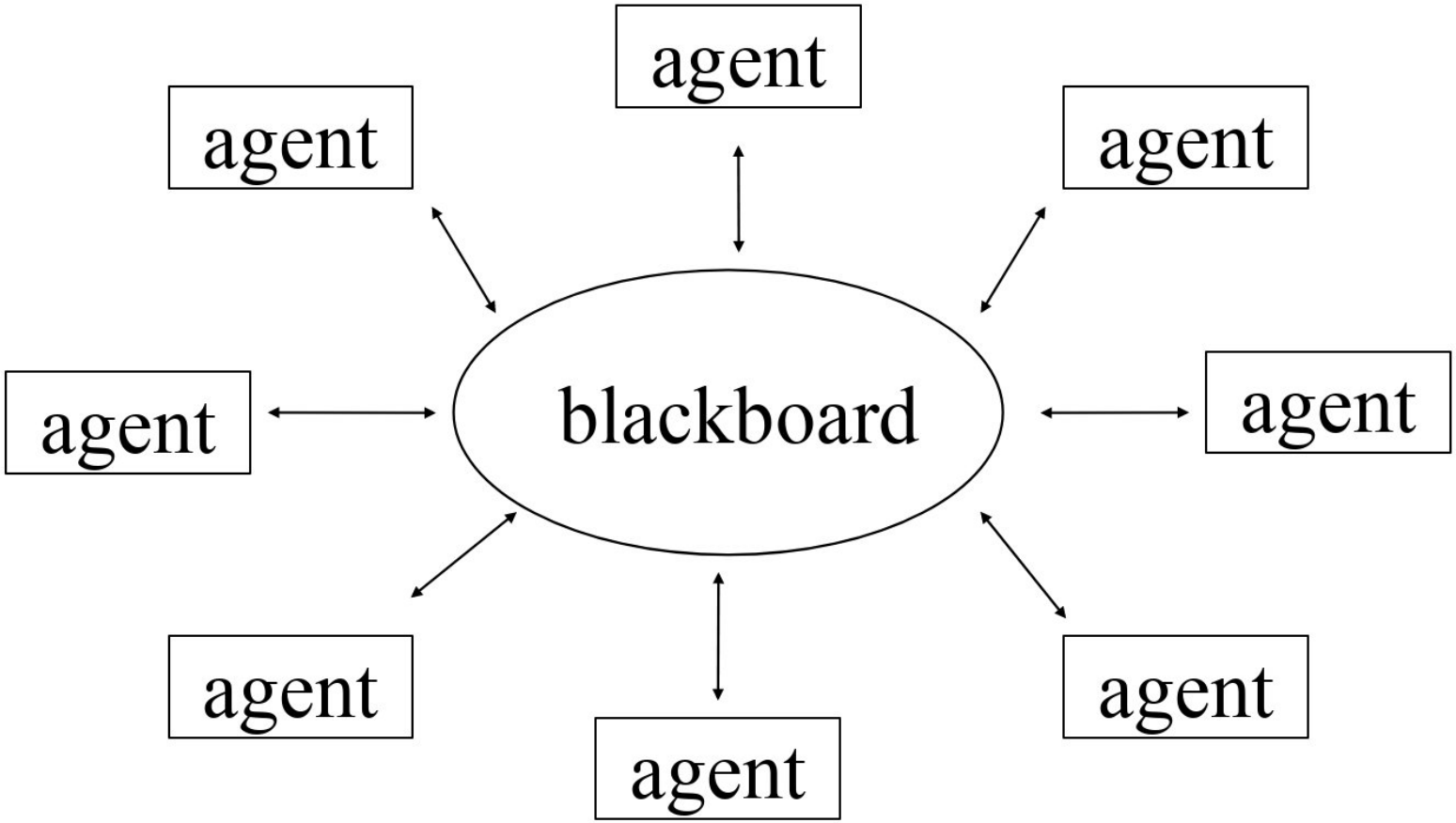
Nature of the medium

- Direct routing
  - Message sent directly to other agent(s) with no interception or attenuation in strength
- Signal propagation routing
  - Commonly used by reactive agents
  - Agent sends signal whose intensity decreases according to distance (e.g., radio signal in physical robots)
- Public notice routing
  - Blackboard systems

3. Agent communication

- We will study the two basic options used in collaborative MAS
  - Blackboard systems
  - Direct message passing

3.1. Agent communication: Blackboard Systems



3.1. Agent communication: Blackboard Systems

- Each agent can put information/data/knowledge on the common information space
- Each agent can read from the blackboard at any moment
- There is no direct communication between agents

3.1. Blackboard Systems - Information

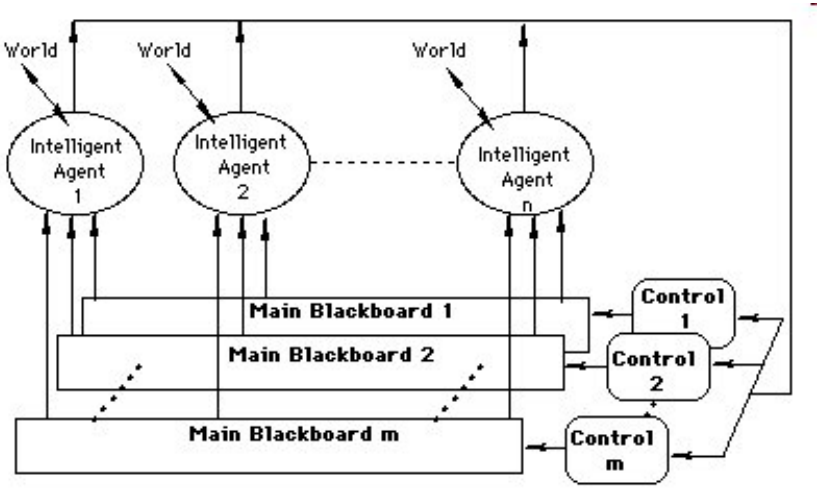
- Data of the common problem
- Current state of the solution
- Next sub-problems to be solved
- Requests of help
- Present task of each agent
- Intermediate results

3.1. Blackboard Systems - Uses

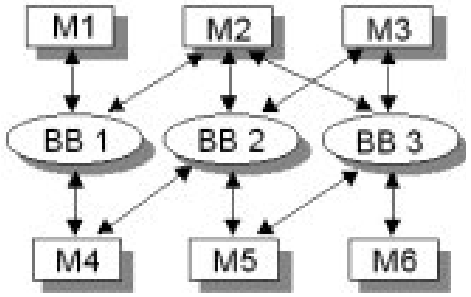
- Detect conflicts
  - Different agents that want to perform the same task
- Notice incompatible solutions
  - Solutions using a shared resource at the same time
- Share results
  - Agents can use partial/complete results obtained by other agents
- Share tasks
  - Agents can request help in solving sub-tasks

3.1. Blackboard Systems - Advantages

- Flexible mechanism for communication/cooperation
  - E.g., *n* blackboards
- Independent of cooperation strategy
- It does not place any restriction on the agents' internal architecture



Multi-Blackboard Architecture

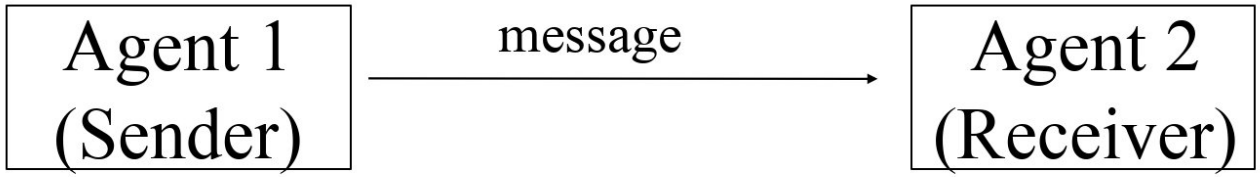


3.1. Blackboard Systems - Disadvantages

- Centralised structure
- System bottleneck
- Everyone has to write info on the blackboard
- Everyone has to read from the blackboard
- Single point of failure

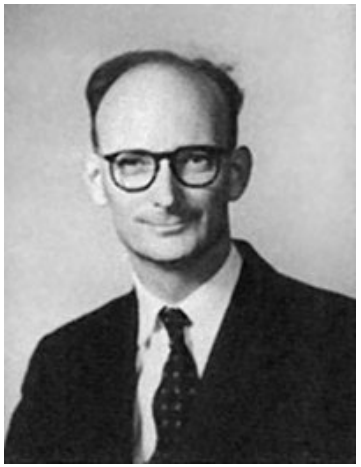
3.2. Agent communication: Message passing

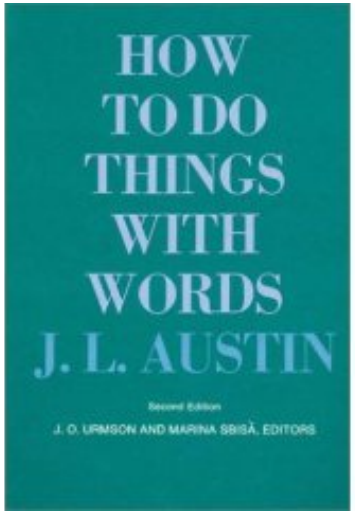
- Information is passed from one agent to another. The nature of this information can be very varied. Speech acts provide one way to describe this variety



3.2. Message passing – Speech Acts

- Most treatments of communication in Multi-Agent Systems borrow their inspiration from *speech act theory*
- Speech act theories are *pragmatic* theories of language, i.e., theories of language use: they attempt to account for how language is used by people every day to achieve their goals and intentions
- The origin of speech act theories is usually traced to Austin's 1962 book, *How to Do Things with Words*





3.2. Message passing – Speech Act Theory

- Austin noticed that some utterances are like 'physical actions', in the sense that they appear to *change the state of the world*
- Paradigmatic examples would be:
  - Declaring war
  - 'I now pronounce you married'
- But more generally, *everything* we say is uttered with the intention of satisfying some goal
- A theory of how utterances are used to achieve intentions is a speech act theory

3.2. Message passing – Types of Speech Act

- *inform* other agents about some data
- *query* others about their current situation
- *answer* questions
- *request* others to act
- *promise* to do something
- *offer* deals
- *acknowledge* offers and requests
- …

3.2. Message passing – Communicative Acts

- Is the door open?
- Open the door (for me)
- OK! I'll open the door
- The door is open
- I am unable to open the door
- I don't want to open the door
- Tell me when the door becomes open
- Does anyone want to open the door?
- I can open the door for you… at a price
- Door? What's that? I don't understand…

query
request
agree
inform
failure
refuse
subscribe
CFP (call for proposals)
propose
not-understood

Same content, **"open(door)"**, but different meanings depending on the performative used

3.2. Message passing – Speech Acts components

- In general, a speech act can be seen to have two components:
  - a *performative verb:*
    (e.g., request, inform, promise, …)
  - *propositional content:*
    (e.g., “the door is open”)

3.2. Message passing – Conversations

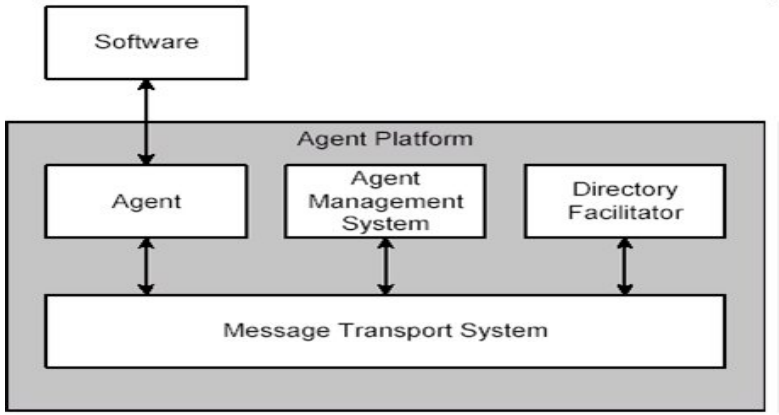
- Speech acts are rarely carried out in isolation
- Speech acts must be considered in terms of the other speech acts of the conversation
- Speech acts therefore can have different meanings based on the previous discourse
- Establishing meaning is thus more than understanding individual utterances

3.2. Message passing – Communication Standards

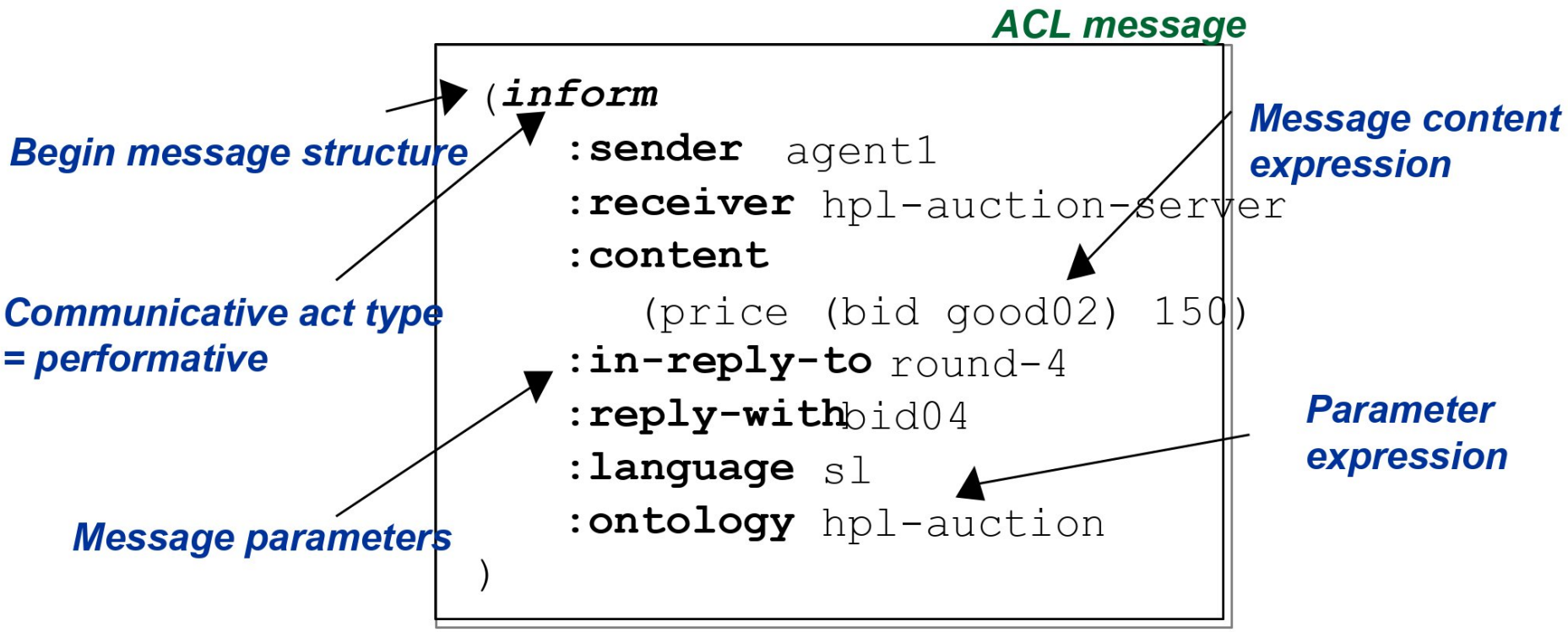
- Agents must understand each other even if running on different machines and/or different operating systems
- Standards
  - Allow different groups to write cooperating agents
  - Help abstract out communication, by defining high-level general languages and protocols
- LLMs
  - Thanks to the usage of LLMs, LLM-based agents can talk to each other using natural language

FIPA

- Foundation for Intelligent Physical Agents
- Beginning (1996): stand-alone non-profit organisation
- Now: IEEE Computer Society standards committee
- Mission: develop and promote agent standards
  - MAS architecture
  - Agent communication language (FIPA-ACL)
- Communication protocols



FIPA-ACL message



FIPA-ACL performatives

performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

FIPA-ACL performatives: Inform

- Content: statement
- The sender informs the receiver that a given proposition is true
- The sending agent
  - Holds that some proposition is true
  - Intends that the receiving agent also comes to believe that the proposition is true
  - Does not already believe that the receiver has any knowledge of the truth of the proposition

```
(inform
      :sender     (agent-identifier :name i)
      :receiver        (agent-
identifier :name j)
      :content    "door( now, open )"
      :language   Prolog)
)
```

FIPA-ACL performatives: Query-if

- Content: proposition
- The sender asks the receiver if a given proposition is true
- The sending agent
  - Does not know if the proposition is true
  - Believes that the receiver knows if the proposition is true

```
(query-if
   :sender (agent-identifier :name i)
   :receiver (set (agent-identifier :name j))
   :content
     "((registered (server d1) (agent j)))"
   :reply-with r09
   …)
```

```
(inform
   :sender (agent-identifier :name j)
   :receiver (set (agent-identifier :name i))
   :content
     "((not (registered (server d1) (agent j))))"
   :in-reply-to r09)
```

FIPA-ACL performatives: Query-ref

- Content: descriptor
- The sender asks the receiver for the object referred to by the descriptor
- The sending agent
  - Does not know which object (or set of objects) corresponds to the descriptor
  - Believes that the receiver knows about the objects corresponding to the descriptor

```
(query-ref
  :sender (agent-identifier :name i)
  :receiver (set (agent-identifier :name j))
  :content
    "((all ?x (available-service j ?x)))"
  …)
```

```
(inform
  :sender (agent-identifier :name j)
  :receiver (set (agent-identifier :name i))
  :content
    "((= (all ?x (available-service j ?x))
       (set (reserve-ticket train)
            (reserve-ticket plane)
            (reserve automobile))))"
  …)
```

FIPA-ACL performatives: Request

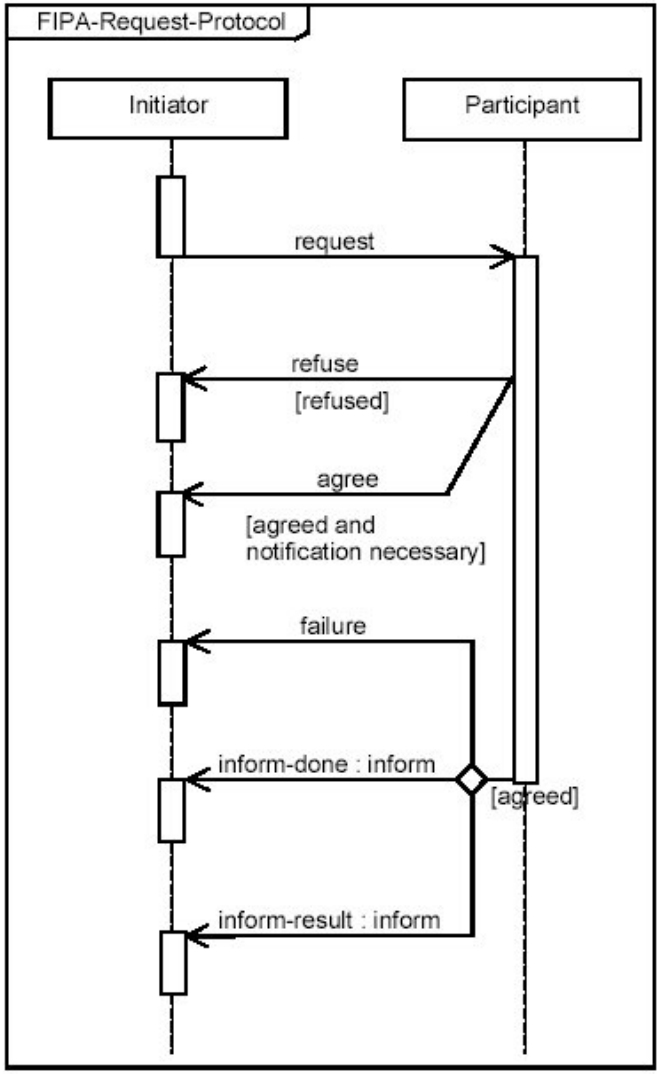
- Content: action
- The sender requests the receiver to perform some action
- The sending agent
  - Intends the action to be performed
  - Believes recipient is capable of performing this action
  - Does not believe that receiver already intends to perform action

```
(request
  :sender       (agent-identifier :name i)
  :receiver     (agent-identifier :name j)
  :content      (action (agent-identifier :name j)
                    open_the_door)
  :language         fipa-sl
)
```

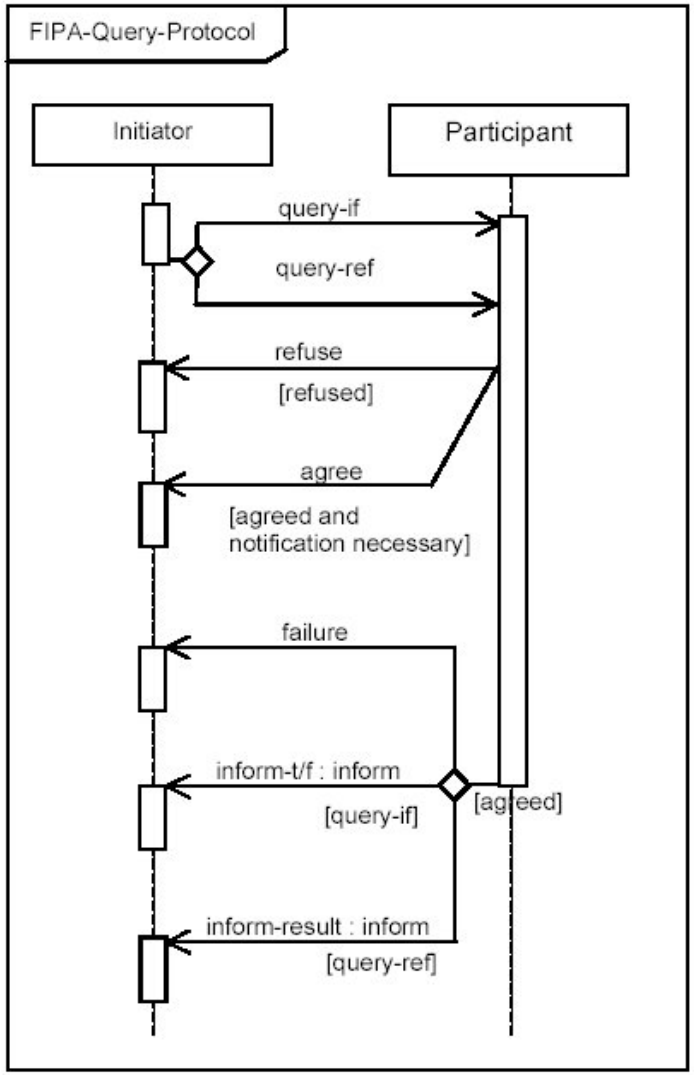
Communication protocols

- There are many situations in which agents engaged in a dialogue with a certain purpose exchange the same sequence of messages
  - When an agent makes a question to another: QUERY
  - When an agent requests a service from another: REQUEST
  - When an agent looks for help from other agents: CONTRACT NET
- To ease the management of this typical message interchanges we can use predefined *protocols*

Communication protocols: FIPA-Request



Communication protocols: FIPA-Query



Contract Net

- An agent asks for other agents to solve a task that it cannot do.
- It is a *task-sharing* protocol consisting in:
  1. Recognition
  2. Announcement
  3. Bidding
  4. Awarding
  5. Expediting

Contract Net: Recognition

- In this stage, an agent recognizes it has a problem it wants help with
- An agent has a goal, and either…
  - Realizes it cannot achieve the goal in isolation — does not have capability
  - Realizes it would prefer not to try to achieve the goal in isolation (typically because of solution quality, deadline, use of resources, etc.)

Contract Net: Announcement

- In this stage, the agent with the task sends out an *announcement* which includes a *specification* of the task to be achieved
- Specification must encode:
  - Description of the task itself
  - Any constraints (e.g. deadlines, quality constraints)
  - Meta-task information (e.g. preference on attributes)
- The announcement is then *broadcast*

Contract Net: Bidding

- Agents that receive the announcement decide for themselves whether they wish to *bid* for the task
- Factors:
  - Agent must decide whether it is capable of expediting task
  - Agent must evaluate the cost of making the task and the benefits it can get from making it
  - Agents must take into account other previous tasks that have not yet been assigned, or other simultaneous announcements…
- If an agent chooses to bid, then it submits a *tender*, detailing the conditions on which it can execute the task

Contract Net: Awarding

- The agent that sent the task announcement must choose between bids & decide who to "award the contract" to
- The result of this process is communicated to the agents that submitted a bid

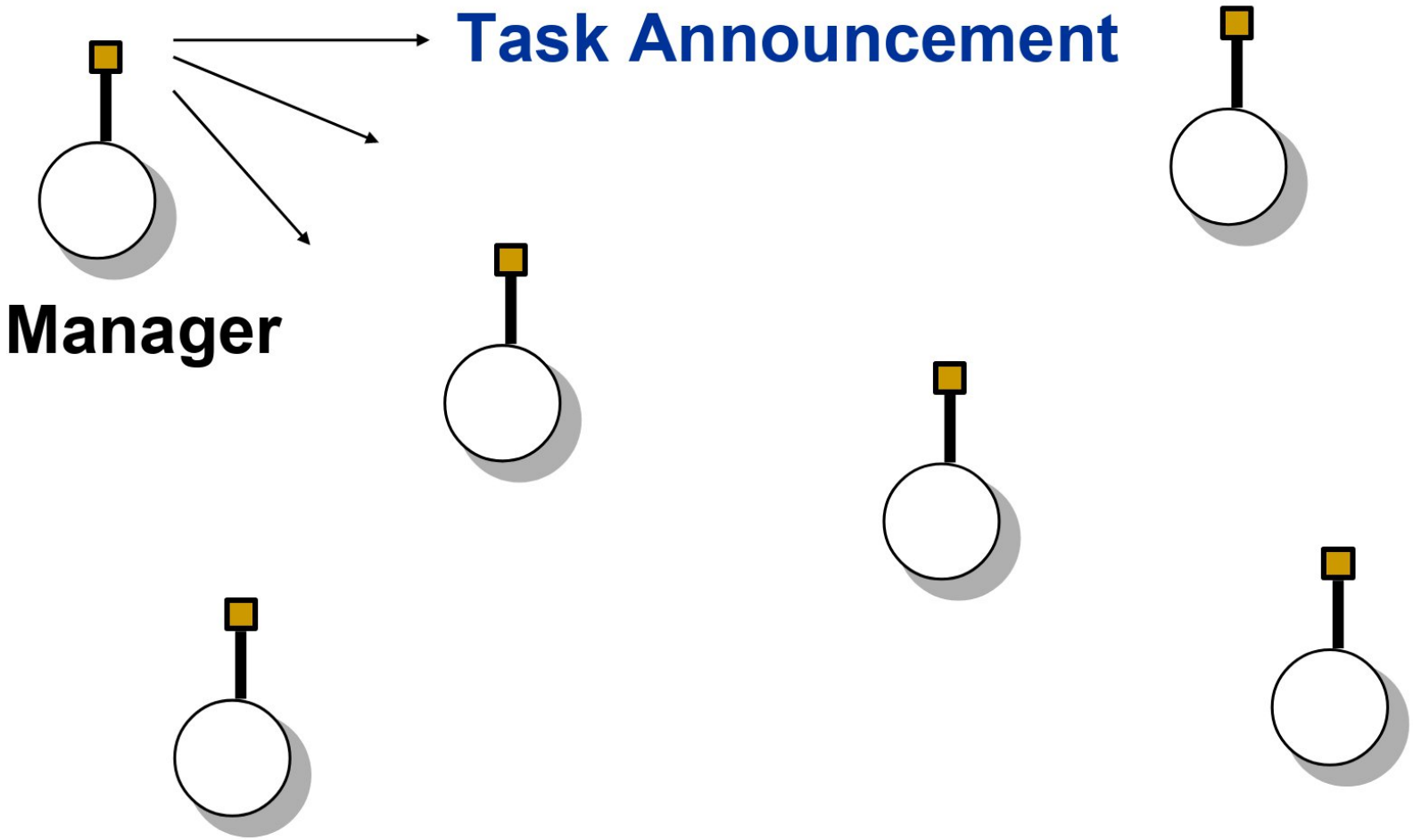
Contract Net: Issues

- How to…
  - … specify *tasks*?
  - … specify *quality of service*?
  - … select between competing offers?
  - … differentiate between offers based on multiple criteria?

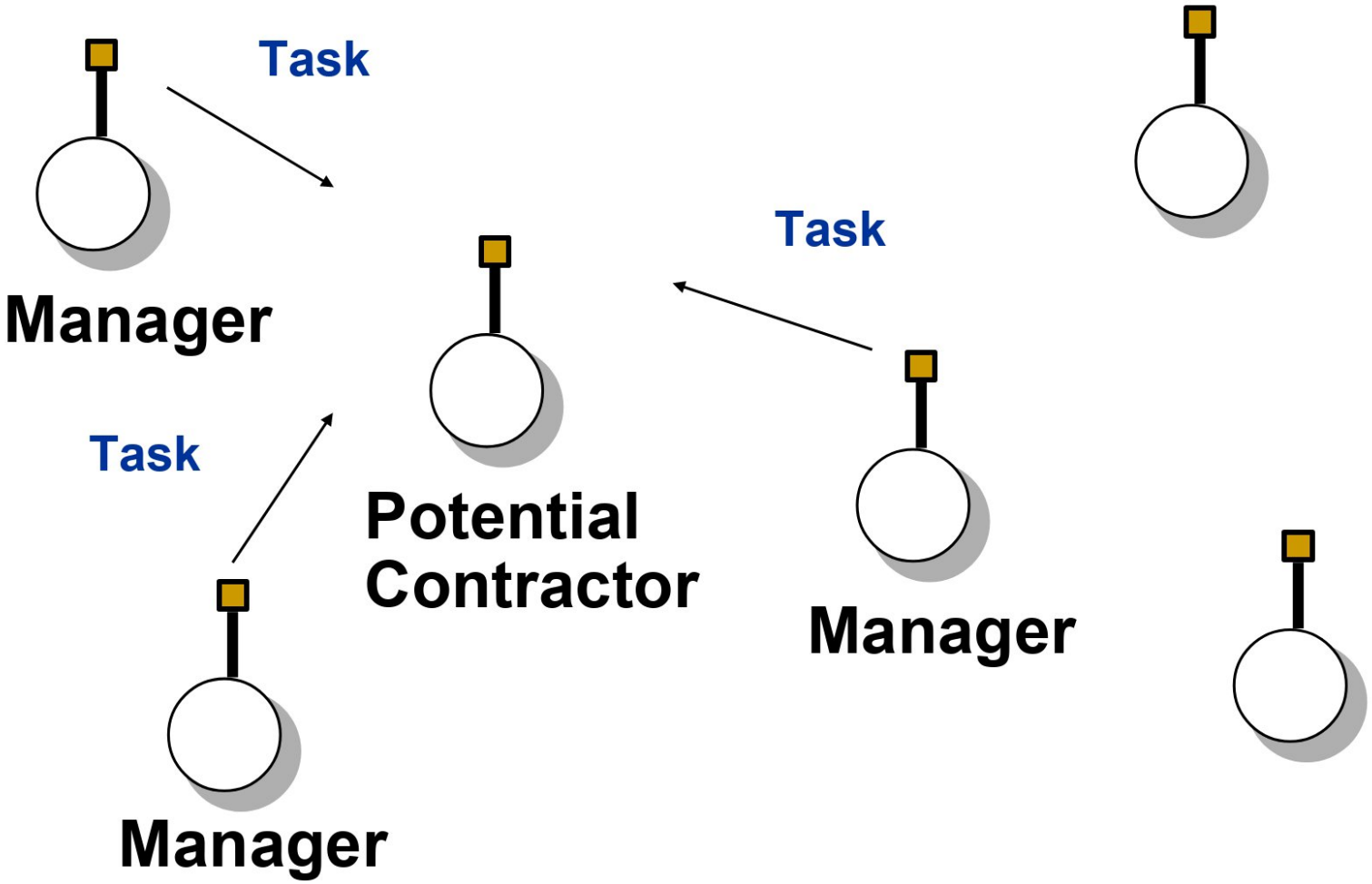
Contract Net: Expediting

- The successful *contractor* then expedites the task
  - That may involve generating further manager-contractor relationships: *sub-contracting*

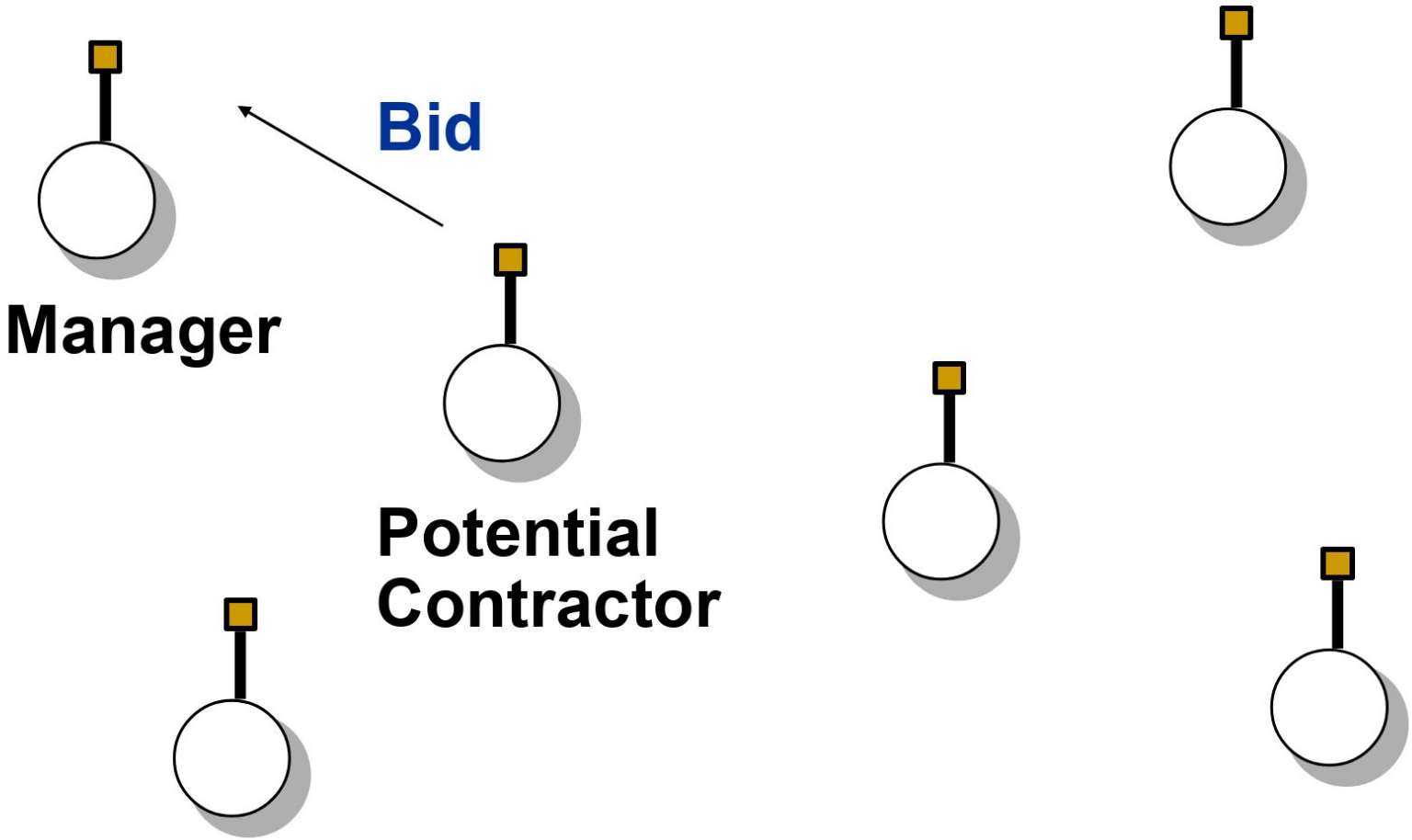
Contract Net: Agent Issues Task Announcement



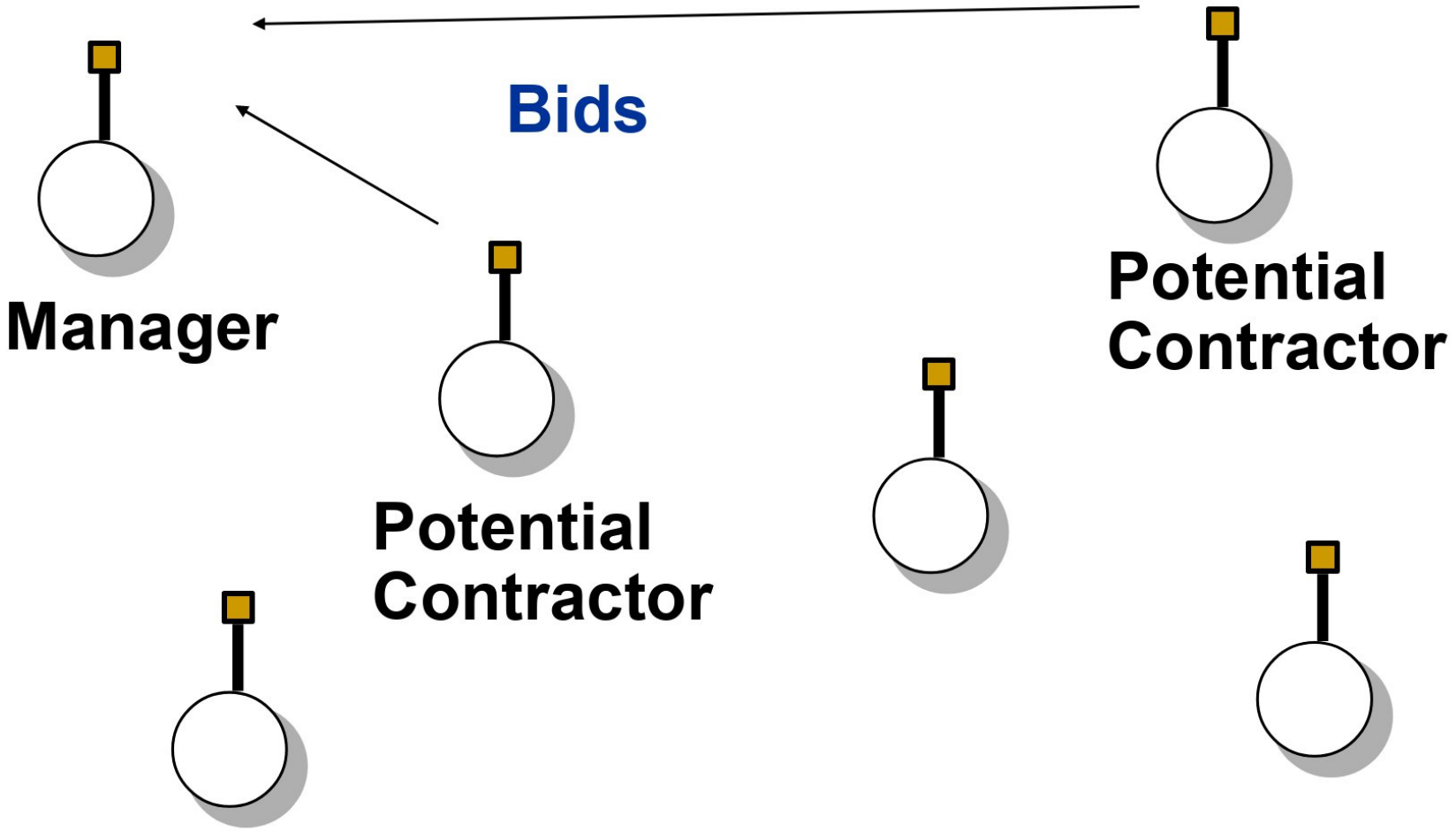
Contract Net: Idle Agent Listening to Task Announcements



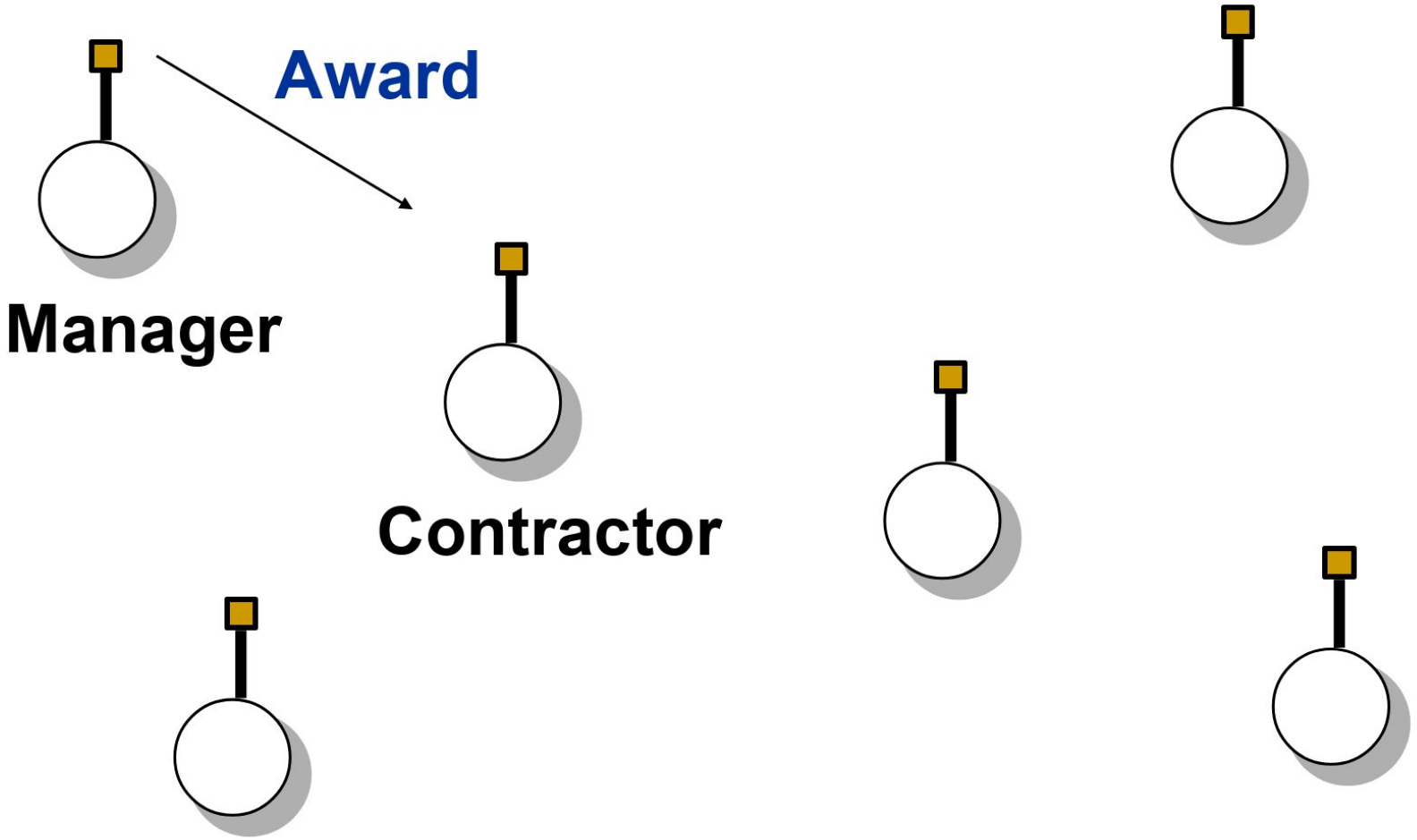
Contract Net: Agent Submitting a Bid



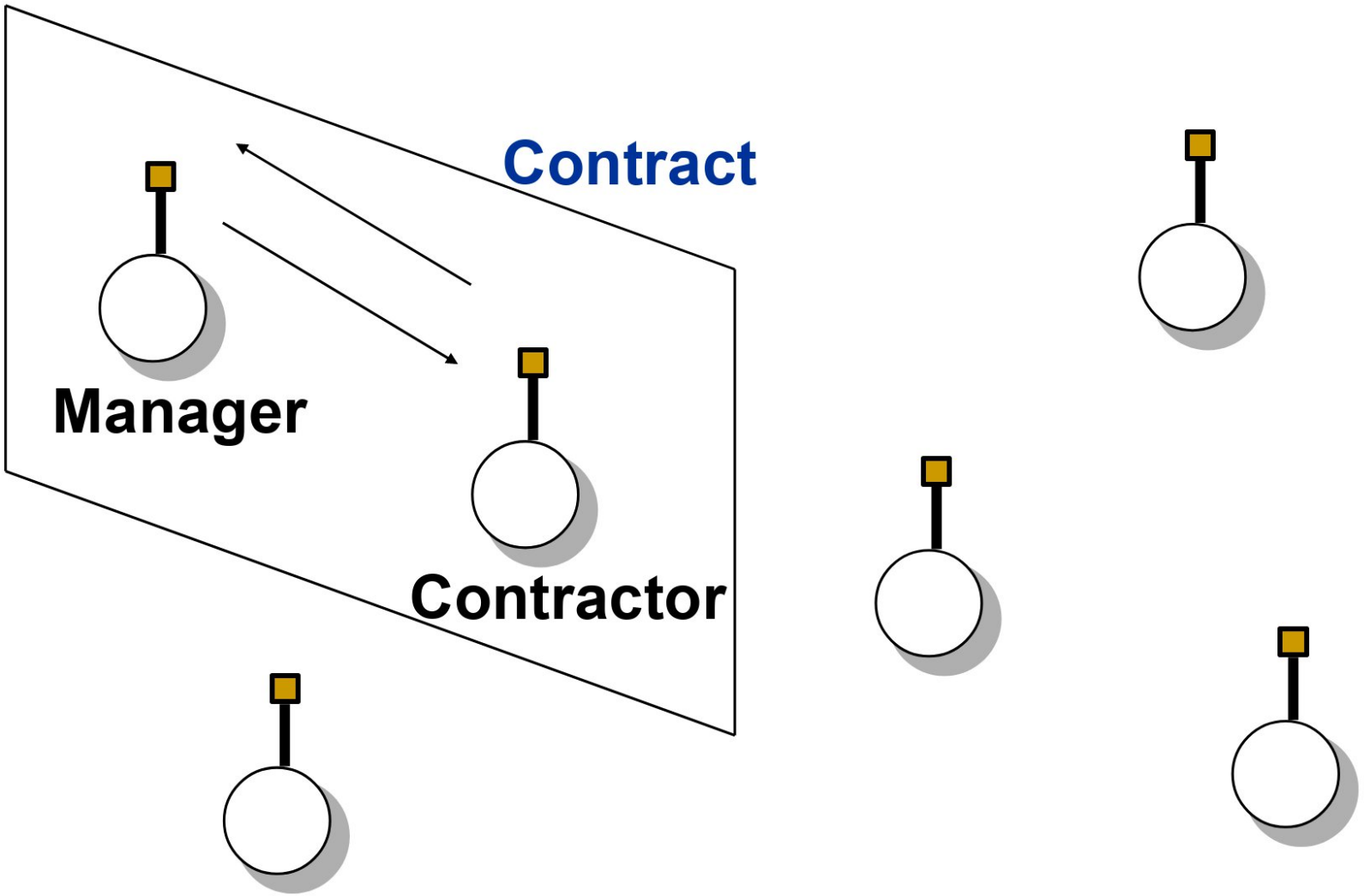
Contract Net: Initiator Listening to Bids



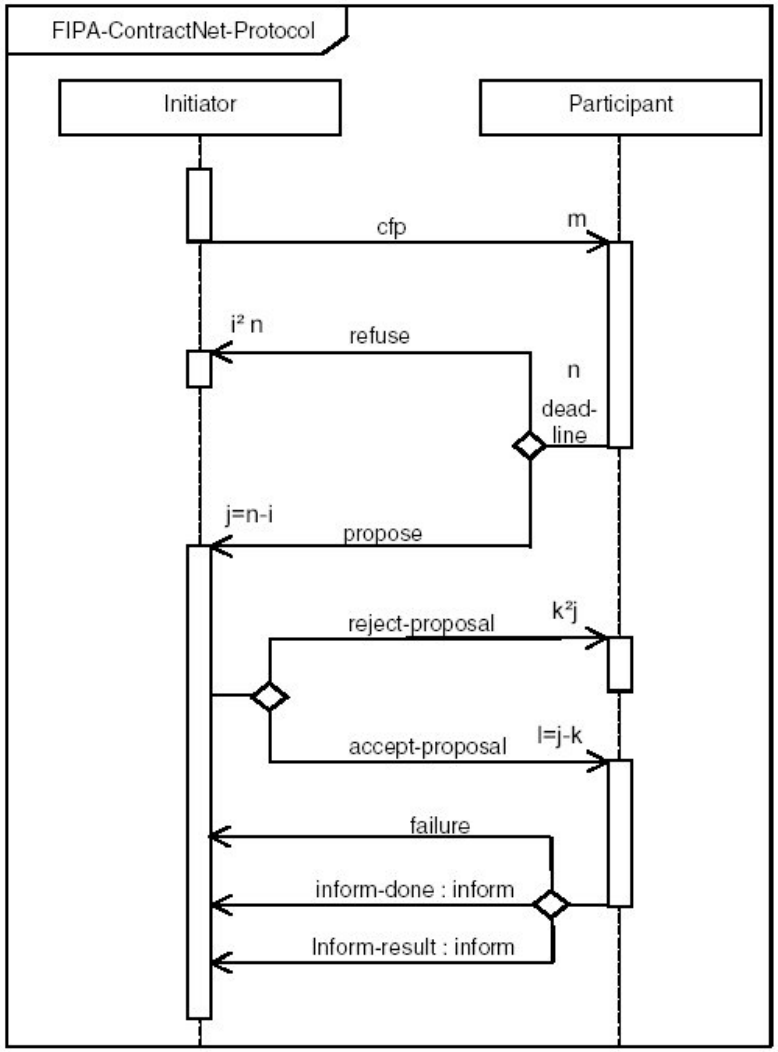
Contract Net: Initiator Making an Award



Contract Net: Contract Established



Contract Net



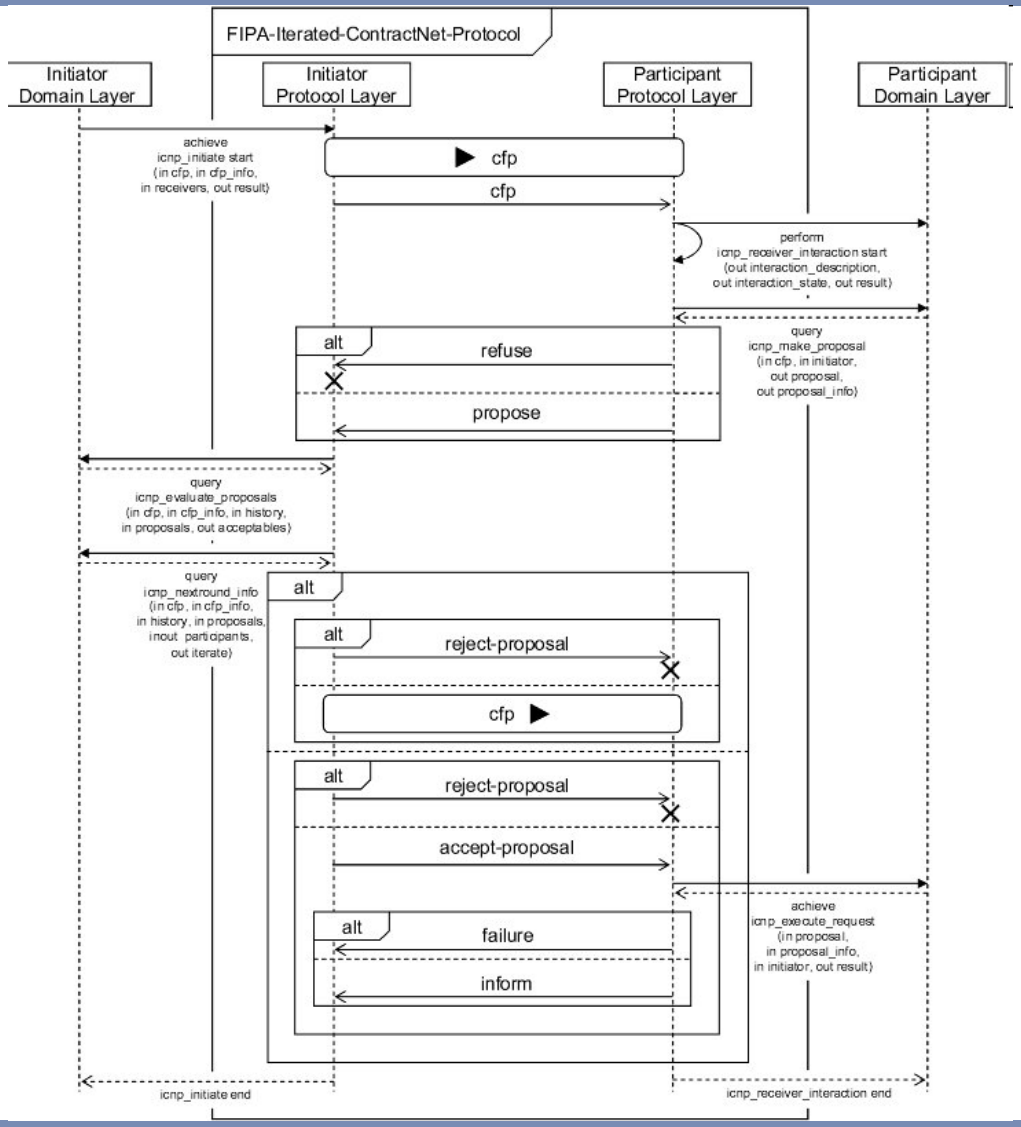
Contract Net: Modules

- Local database
  - Knowledge base, info. on the state of negotiations and the state of the solution of tasks
- Interface module
  - Sends/receives messages, deals with the communication with the other nodes
- Task processor
  - Executes the tasks assigned to the node
- Contract processor
  - Studies new offered tasks, submits bids, formalizes contracts

Contract Net: Efficiency Modifications

- Focused addressing — when general broadcast isn't required
  - Agents could automatically *learn* which are the most appropriate nodes for common tasks
- Directed contracts — when manager already knows which node is appropriate
  - For instance when a very similar task has already been done in the past
- The nodes can make proactive offers to potential managers of the kind of tasks they are able to execute

Iterated Contract Net



Contract Net: Features

- Two-way dynamic transfer of information
- Mutual selection
  - Bidders select from among task announcements
  - Managers select from among bids
- Local evaluation
  - Preserving autonomy and private information of agents

Contract Net: Limitations

- Some stages of distributed problem solving are non-trivial:
  - Problem Decomposition
  - Solution Synthesis
- Computational overhead
  - Messages
  - Time – deliberation, analyze offer/bid, wait for decisions

Recommended readings

- Read chapters 6 and 7 of book by Wooldridge: Introduction to multiagent systems