7. Coordination in Distributed Systems

Part II:
Cooperation (II)
Competition (Negotiation)

Ulises Cortés

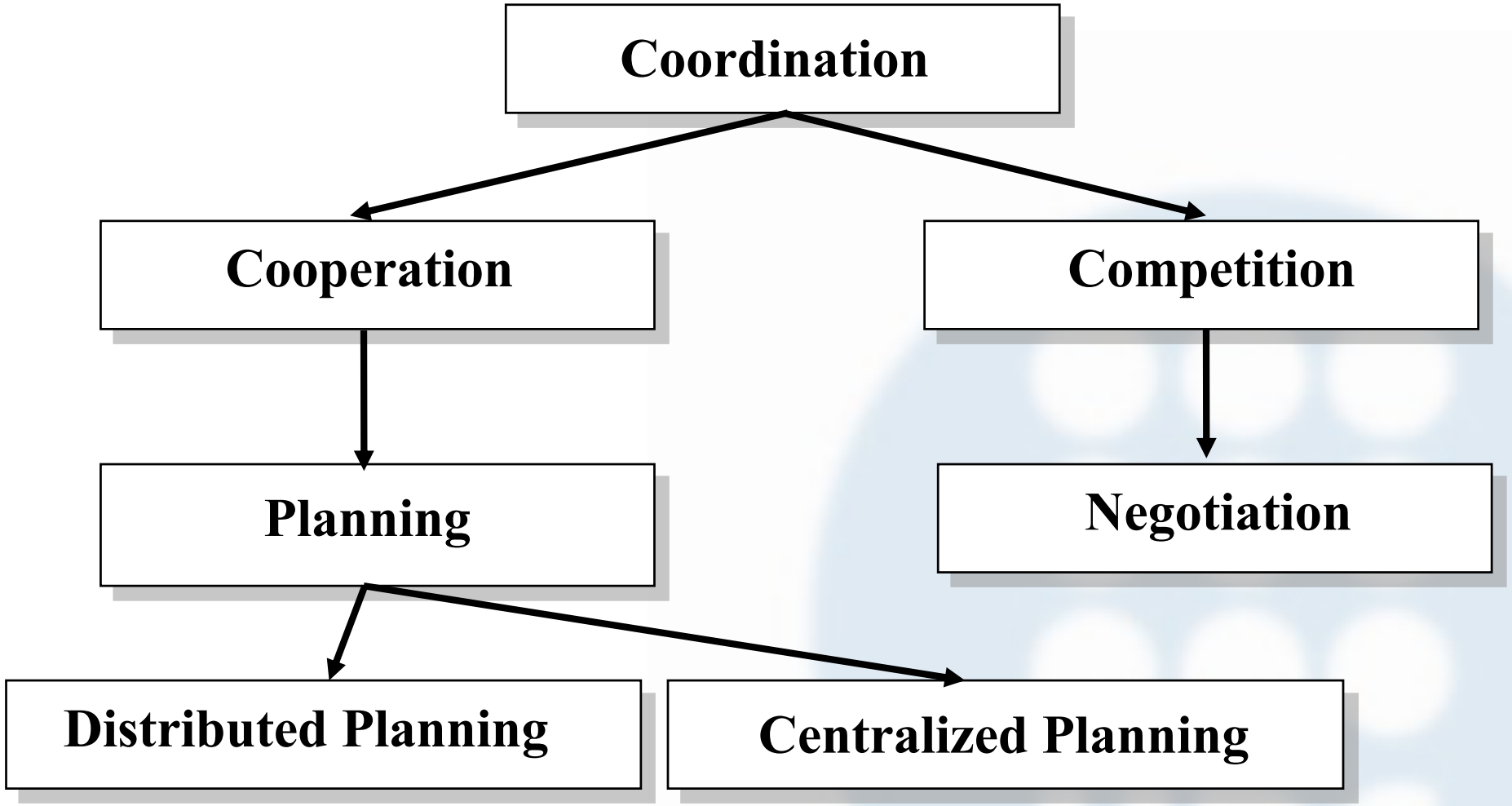
Sergio Álvarez

SID 2019

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
https://kemlg.upc.edu

Coordination

Types of coordination



Coordination Algorithms

Focusing on the nature of the distributed problem

- Coordination by “Algorithm” is somewhat controversial since some approaches do not allow for significant Agent Autonomy in the process.
- Two main approaches:
  - **Distributed Constraint Satisfaction** (DCSP): an extension of CSP solving techniques which capture several variables in each agent. Agents propagate choices for the “edge variables” which affect others.
  - **Hierarchical Authority Algorithms** (Durfee et. al.): mechanisms which enforce authority values on participation and according to these rankings drive plan interchange processes.

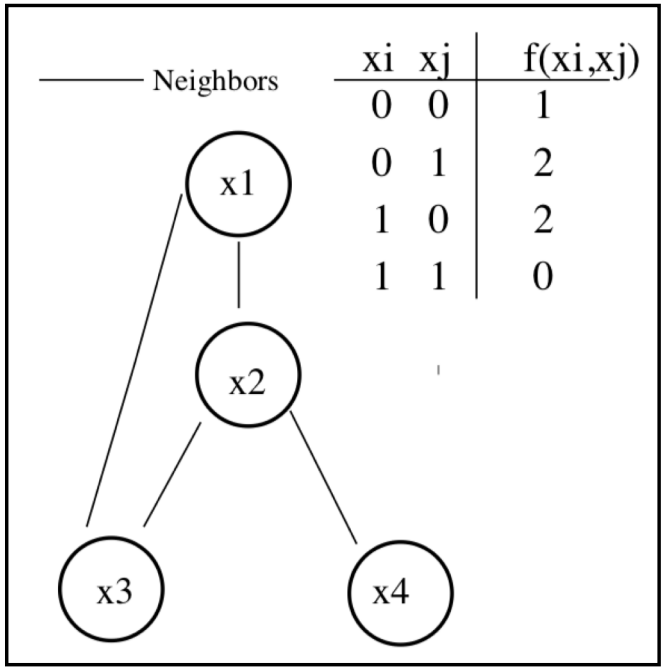
Distributed Constraint Optimisation Problems

Distributed choice of a set of values

- DCOP is a type of problem in which a set of agents distributedly assign values to a set of variables, optimising the cost attributed to those values
- All algorithms that solve DCOP can solve all problems of this family
- DCOP can be defined as a tuple $\langle A, V, \mathfrak{D}, f, \alpha, F \rangle$, where
  - A is a set of agents
  - V is a set of variables
  - $\mathfrak{D}$ is a set of domains for the values in V
  - f is a function mapping every possible variable assignment to a cost, which can be infinite
  - α is a function mapping variables to each agent
  - F is an operator aggregating individual costs for all possible variable assignments, usually a summation

Distributed Constraint Optimisation Problems

Example



xi	xj	f(xi,xj)
0	0	1
0	1	2
1	0	2
1	1	0

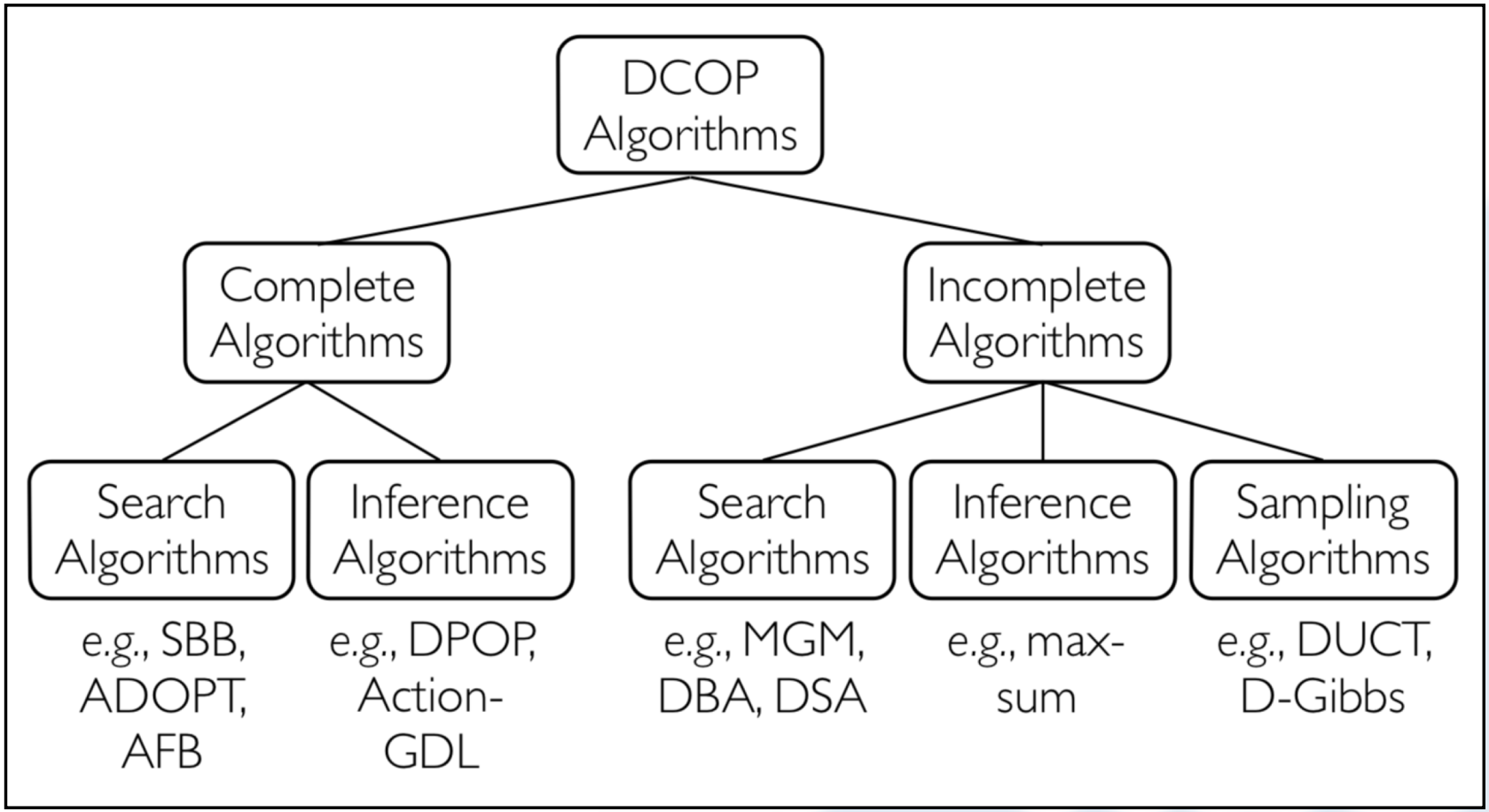
- We prefer to have neighbours with the same value, with a preference for (1, 1)
- An example of algorithm would be to let x_1 assign a value and continue with x_2, and so on
- In any case, the system will end up optimising to all 0's

$$F(\mathcal{A}) = \sum_{x_i, x_j \in X} f_{ij}(d_i, d_j) \text{ where } x_i \leftarrow d_i \text{ and } x_i \leftarrow d_i \text{ in } \mathcal{A}$$

$$F(\{(x_1, 0), (x_2, 0), (x_3, 0), (x_4, 0)\}) = 4$$
$$F(\{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 1)\}) = 0$$

Distributed Constraint Optimisation Problems

DCOP Algorithms



Distributed Constraint Optimisation Problems

Recap

- Used in many domains
  - IoT, Wireless and P2P networks
  - Meeting scheduling
  - Traffic control
- DCOPs are NP-Hard
- Complete Algorithms are optimal but do not scale well
- The performance of Incomplete Algorithms depends a lot on the application domain
  - They either tackle the communication overhead or the computational complexity
- Not much autonomy is left to the agents, esp. if the variables represent task assignment
- Not much fault tolerance, agents need to be trustworthy

Distributed Consensus algorithms

How to overcome node failures

- Consensus problem: how to achieve overall system reliability in the presence of faulty agents
- A consensus protocol is an interaction protocol that is fault tolerant or resilient w.r.t. a (limited) number of faulty agents
- In general, a consensus protocol describes how to maintain a single value in a multi-agent system, via leader election or majority voting
- The single value can be very complex, e.g., the Bitcoin ledger
- A consensus protocol defines the processes that each agent has to run in order to maintain this value

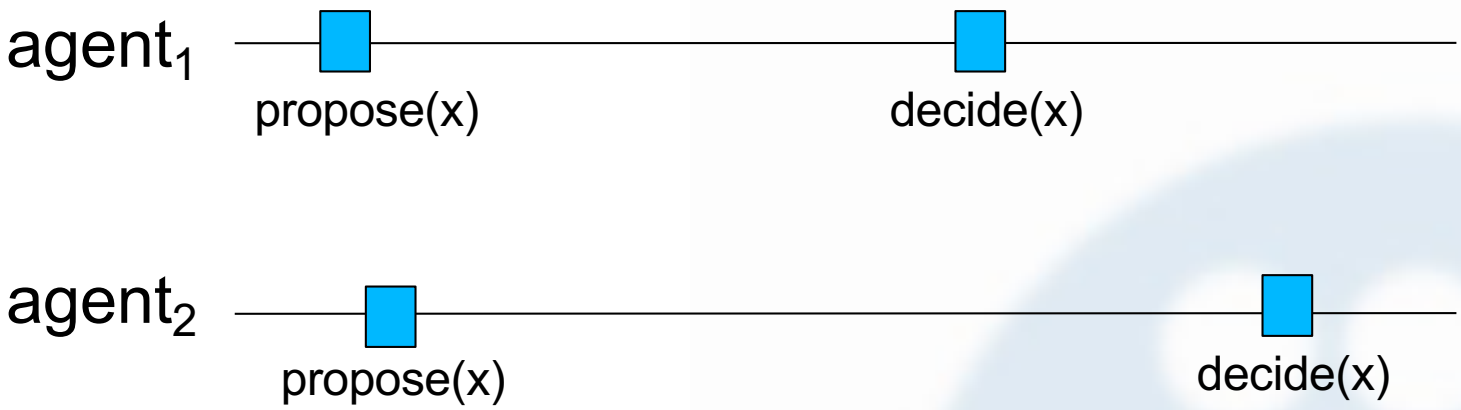
Distributed Consensus algorithms

Desirable properties

- Fault tolerance in consensus is defined as fulfilling the following properties:
  - Termination
    - Eventually, every correct process decides some value
  - Integrity (Validity)
    - If all the correct processes proposed the same value, then any correct process must decide that value
    - A node decides at most once
    - Any value decided is a value proposed
  - Agreement
    - Every correct process must agree on the same value

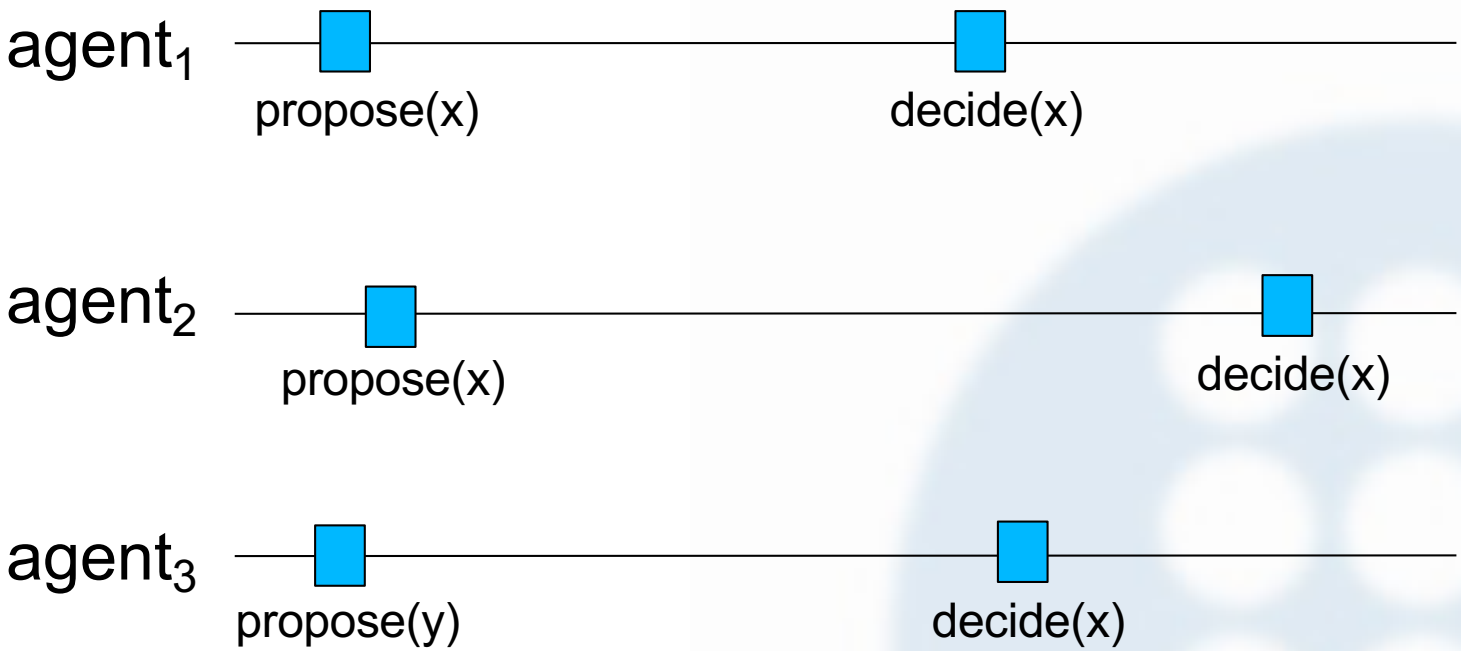
Distributed Consensus algorithms

Is this consensus?



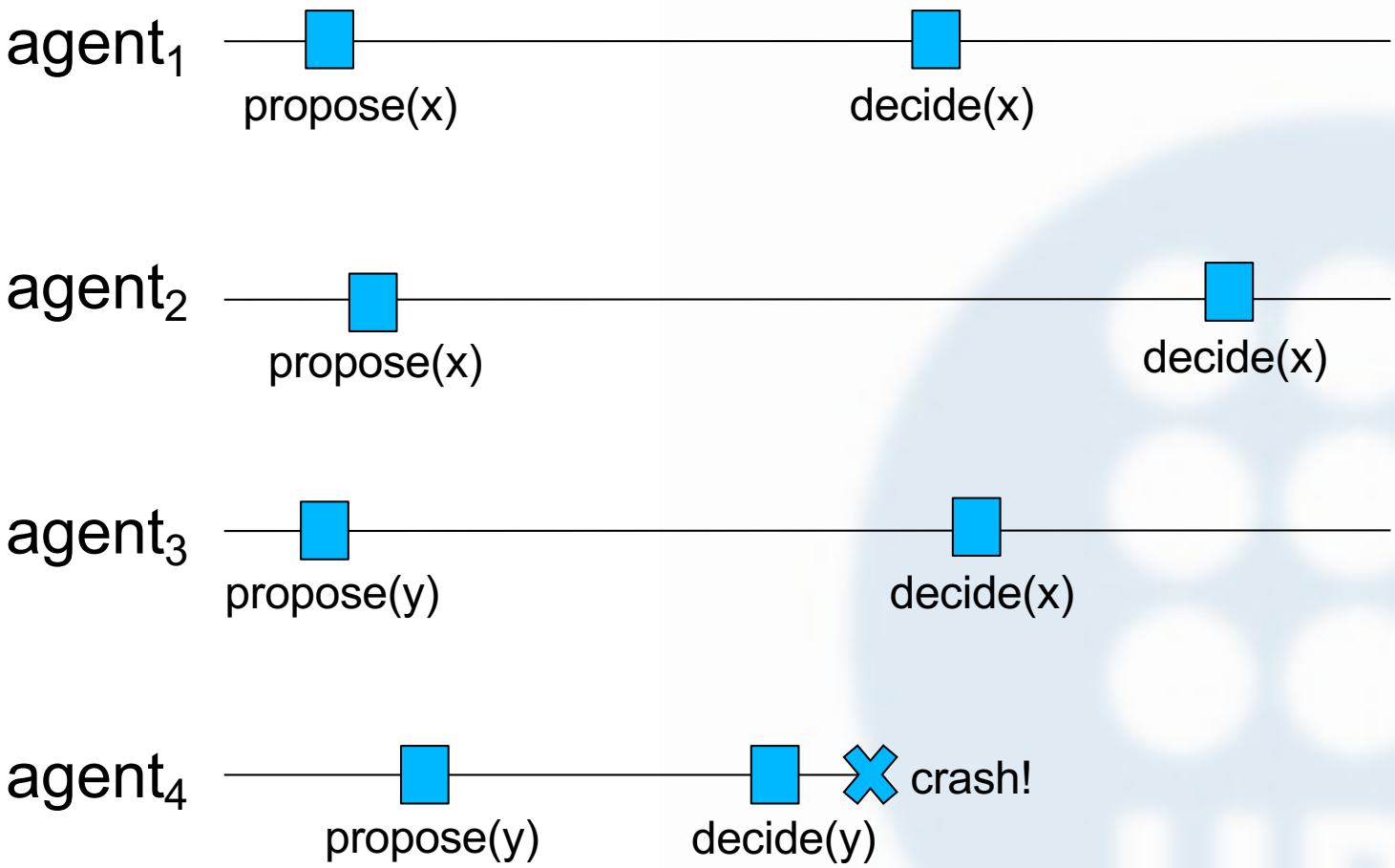
Distributed Consensus algorithms

Is this consensus?



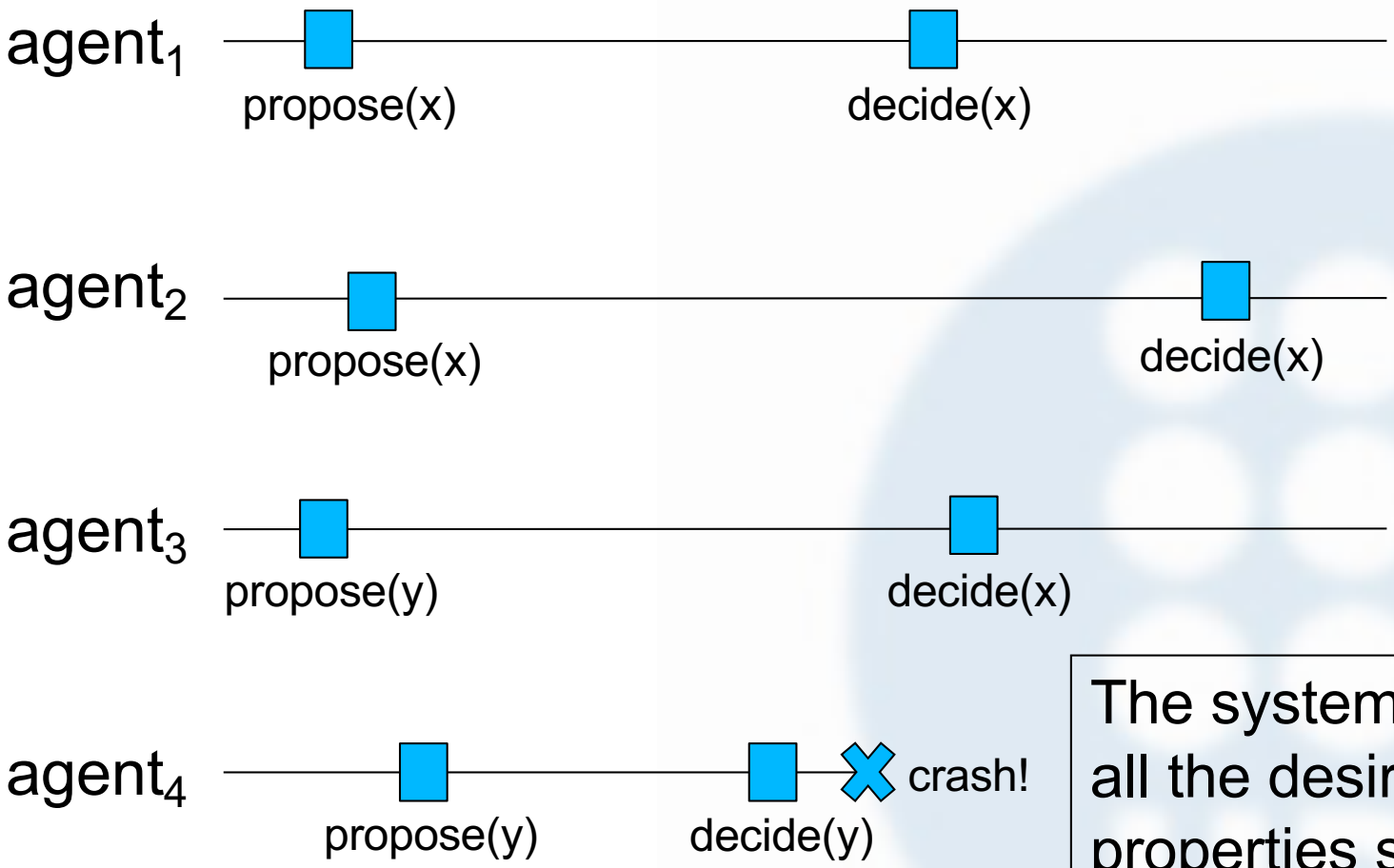
Distributed Consensus algorithms

Is this consensus?



Distributed Consensus algorithms

Is this consensus?



The system fulfills all the desirable properties so it is still a consensus

Distributed Consensus algorithms

Paxos

- Paxos [Lamport, Malkhi, Zhou 2010] is currently the most popular distributed consensus algorithm
  - Raft is a similar proposal, (arguably) more simple formally
- A domain-specific state machine is replicated across the system

Distributed Consensus algorithms

Paxos

- Paxos allows
  - Agents operating at arbitrary speed
  - Agents experiencing failures
  - Messages can be sent asynchronously
  - Messages can get lost, reordered or duplicated
  - Agents with persistent storage can re-join after failures (following a specific crash recovery procedure)
- Consistency is guaranteed as long as
  - If *F* agents fail, *2F + 1* agents have to be correctly functioning
  - Agents can send messages to any other agent
  - Messages are delivered without corruption
  - Byzantine failures do not happen

Distributed Consensus algorithms

Paxos

- In Paxos, there are three agent roles:
  - Proposer
  - Acceptor
  - Learner
- Any agent can become a proposer at any point in time
- A proposer creates a *Quorum of Acceptors* for its proposal
  - A quorum is a subset of the set of agents that is a majority, i.e. its size is larger than half the size of the full set
  - *{A,C,D}* is a possible quorum for *{A,B,C,D}*
- The rest of the agents are *learners* for the scope of the proposal

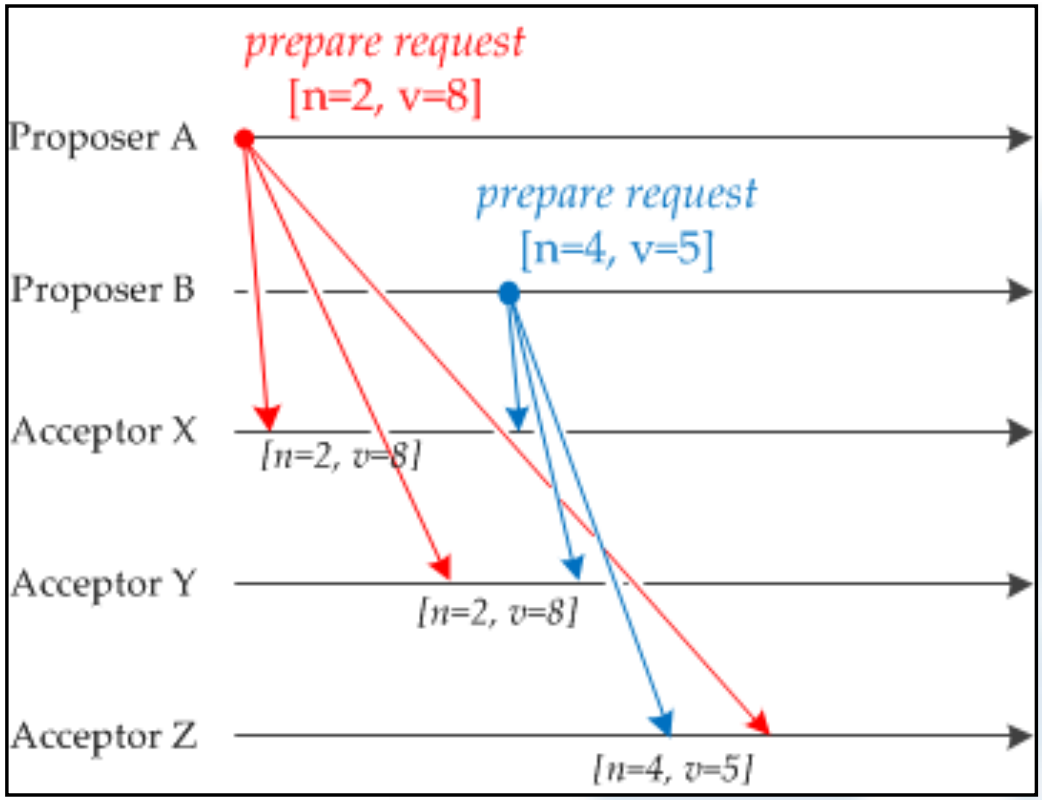
Distributed Consensus algorithms

Paxos

- Phase 1a: Prepare
  - A proposer *P* creates a proposal identified with a number *N*
  - *N* must be greater than any previous proposal number used or received by proposer *P*
  - *P* sends a *Prepare* message containing this proposal to a Quorum of acceptors
  - Proposer *P* decides who is in the *Quorum*

Distributed Consensus algorithms

Paxos (Example)



https://medium.com/@angusmacdonald/paxos-by-example-66d934e18522

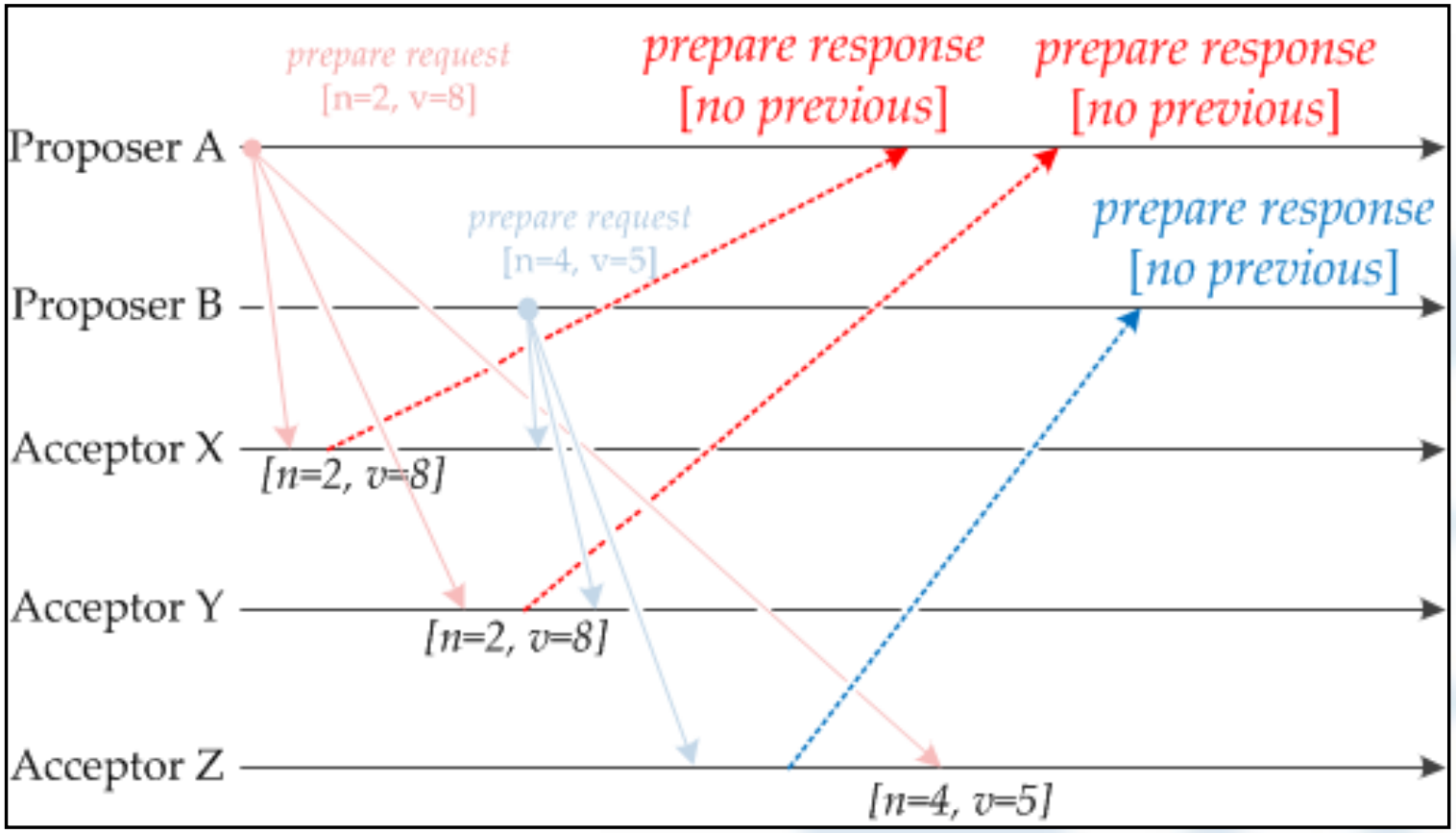
Distributed Consensus algorithms

Paxos

- Phase 1b: Promise
  - If the proposal number *N* is higher than any previous proposal number received from any proposer by an *acceptor A*, then *A* must return a promise to ignore all future proposals having a number less than *N*
  - If *A* accepted a proposal at some point in the past, it must include the previous proposal number *N'* and previous value *v* in its response to *P*

Distributed Consensus algorithms

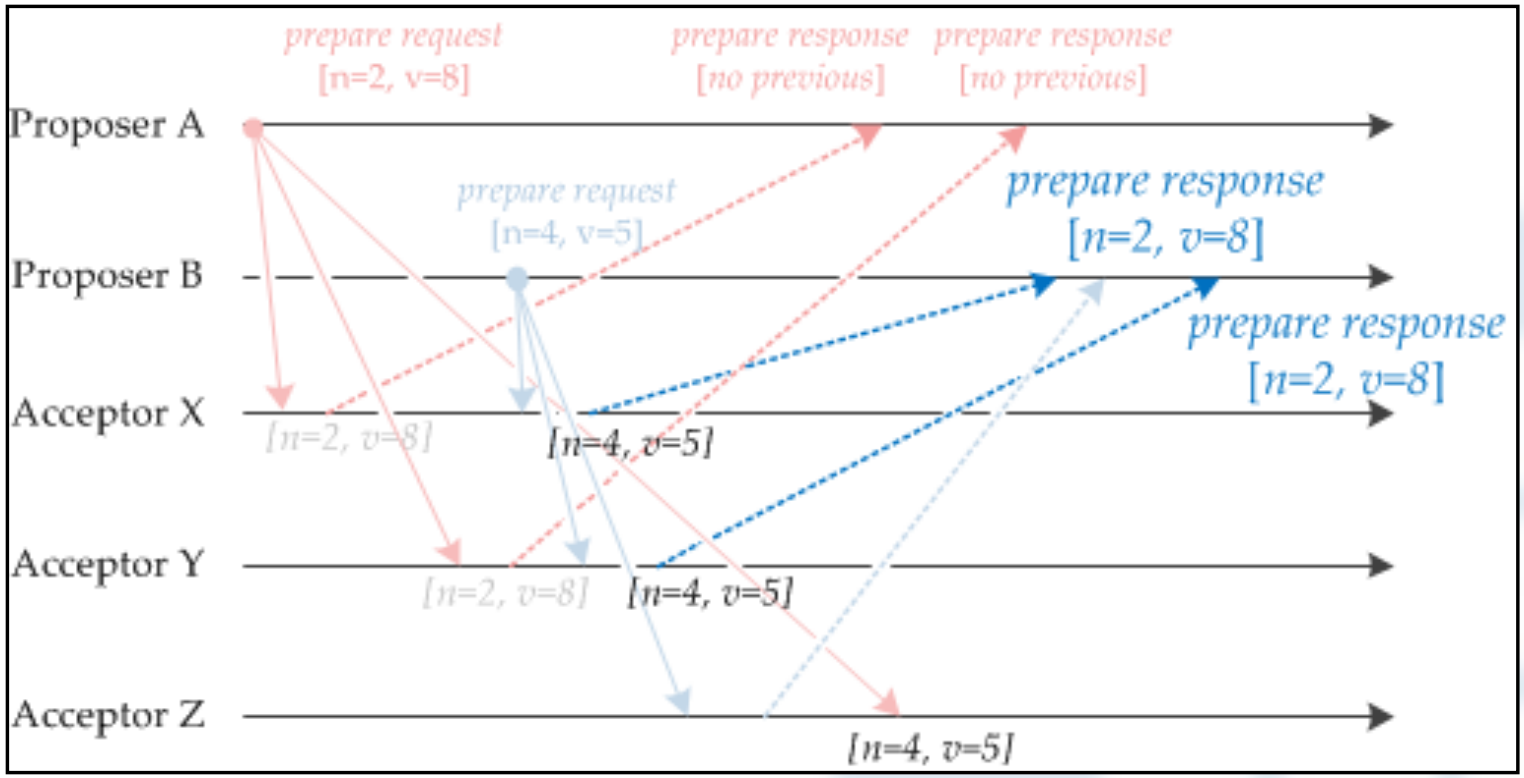
Paxos (Example)



https://medium.com/@angusmacdonald/paxos-by-example-66d934e18522

Distributed Consensus algorithms

Paxos (Example)



https://medium.com/@angusmacdonald/paxos-by-example-66d934e18522

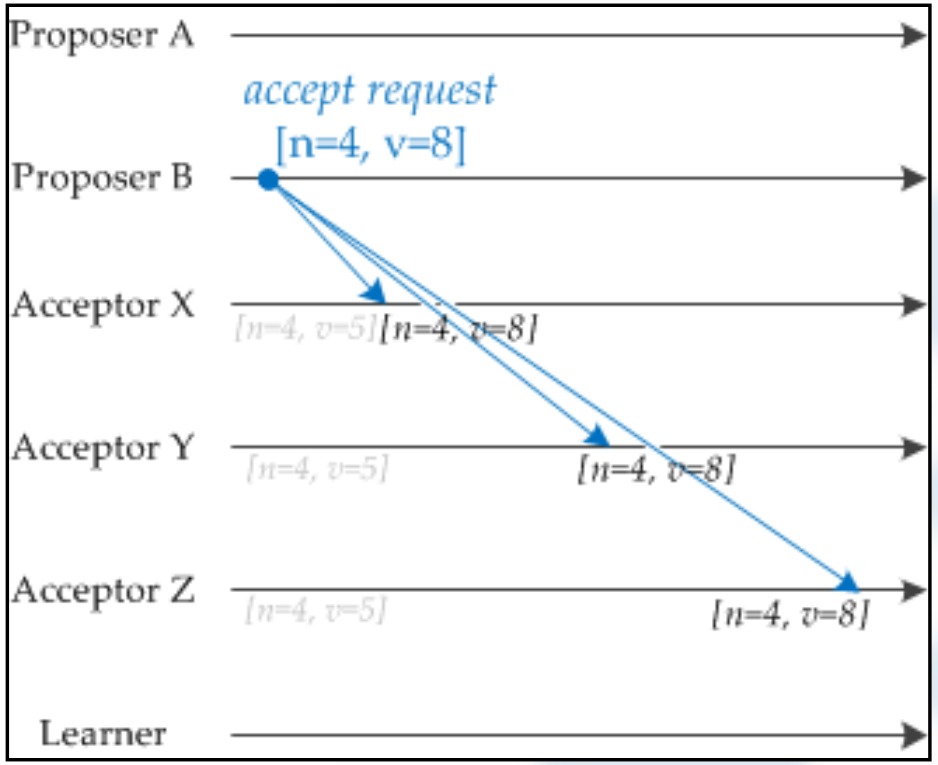
Distributed Consensus algorithms

Paxos

- Phase 2a: Accept Request
  - If *P* receives enough promises from a *Quorum of Acceptors*, it needs to set a value to its proposal
  - If any *acceptors* had previously accepted any proposal, then *P* is receiving their values
    - *P* must then set the value of its proposal to the value *v* associated with the highest proposal number reported by the acceptors.
  - If none of the *acceptors* had accepted a proposal up to this point, *P* may choose any value to its proposal

Distributed Consensus algorithms

Paxos (Example)



https://medium.com/@angusmacdonald/paxos-by-example-66d934e18522

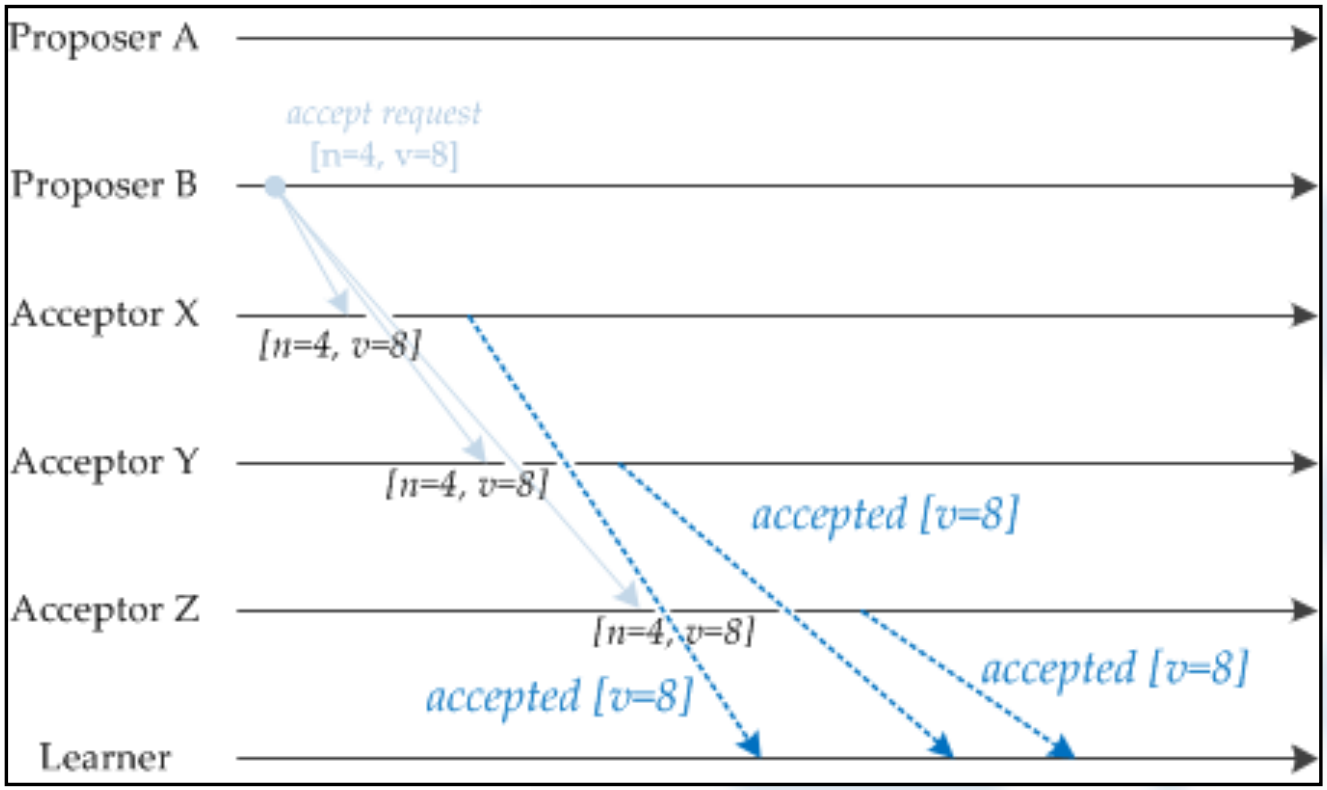
Distributed Consensus algorithms

Paxos

- Phase 2b: Accept
  - If an *acceptor A* receives an *Accept Request* message for a proposal *N*
    - If it has already promised to only consider proposals having an identifier greater than *N*
      - *A* ignores the *Accept Request*
    - If it has not promised to consider any proposals for an identifier greater than *N*
      - *A* registers the corresponding value *v* associated to proposal *N*
      - *A* sends an *Accepted* message to the proposer *P* and to every learner

Distributed Consensus algorithms

Paxos (Example)



https://medium.com/@angusmacdonald/paxos-by-example-66d934e18522

Distributed Consensus algorithms

Paxos

- Paxos and Raft (https://raft.github.io) are being used in a wide range of applications, from databases to UAVs
- Very reliable for distributed, asynchronous scenarios where occasional failure is assumed
- Like DCOPs do not allow for much agent autonomy by default
  - But do not require very complex reasoning cycle implementations
- Byzantine failures can be solved with e.g., extensions of Paxos or with Blockchain

Types of Coordination

Competition and Negotiation

- **Competition** is a kind of coordination between antagonist agents which compete with each other or that are selfish.
- We will be more interested in **Negotiation**, as it is a kind of competition that involves some higher level of intelligence.
- The degree of success in negotiation (for a given agent) can be measured by
  - The capability of this agent to maximize its own benefit
  - The capability of not taking into account the other agents' benefit or even trying to minimize other agents' benefit.

Negotiation

Resolving conflicts

- Negotiation is the act of "**Resolving inconsistent views to reach Agreement**" (Lassri)
- Negotiation could be about many things:
  - **Costs**: a linear scale – how much to pay for a service – generally using economic mechanisms and preference evaluation.
  - **Truth**: whether something is true or not – generally using *argumentation*.
  - **Action**: on which action a group of agents should take – also often using *argumentation*.

Negotiation

Negotiation as Coordination

- Negotiation is itself a coordination process since:
  - Agents agree to a pre-defined set of possible actions and **rules** for the negotiation process.
  - They have the **shared goal** of *reaching agreement*.
  - The information exchanged often contains details of **actions to be taken**.
- Agents however likely do not share exactly the same objective within the negotiation:
  - Buyers seek a low price
  - Seller seek a high price

Negotiation

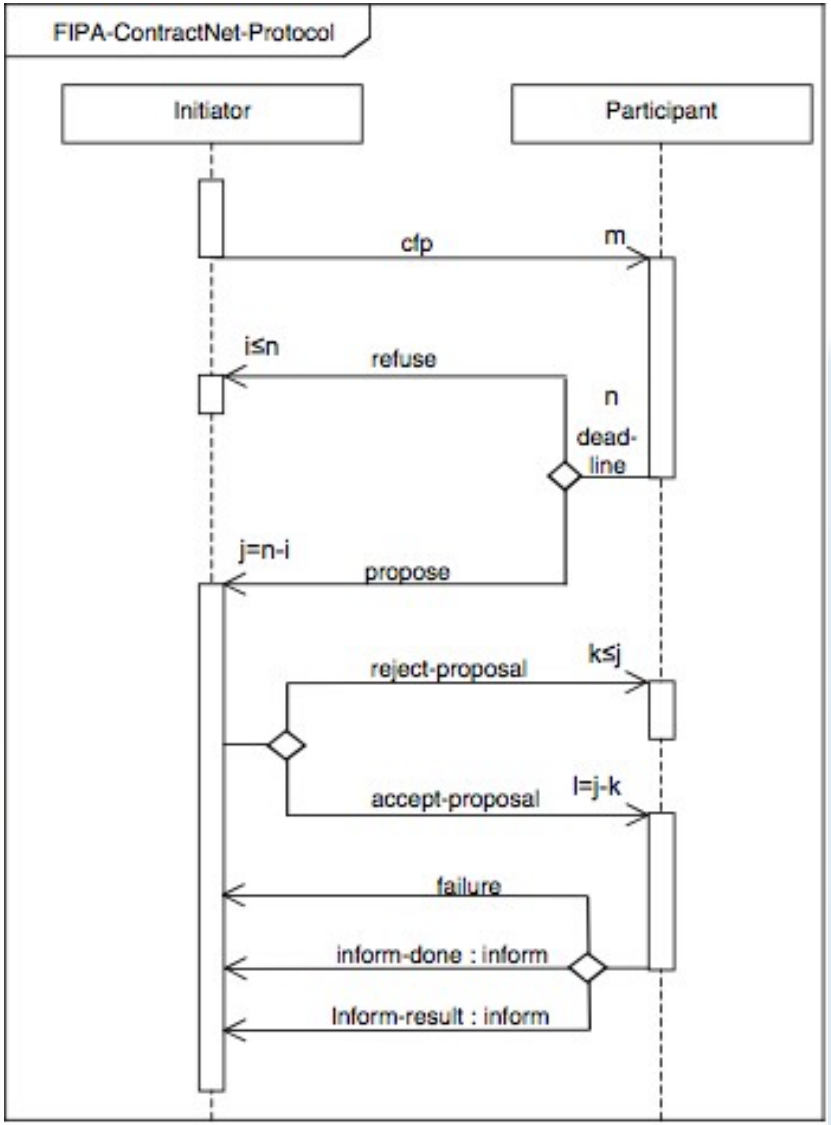
Methods for negotiation

- Common negotiation techniques include:
  - (**Iterative**) **Contract Net** (Simon and Davies): using a call-for-offers and response mechanism – in particular when counter offers are allowed.
  - **Game Theory based approaches** (Levy, Zlotkin, Roschein): sharing utility functions or seeing negotiation convergence as an iterative prisoners dilema.
  - **Recursive and Iterative methods** (Lassri and others): convergence methods / rules for multi-round negotiations.
  - **Argumentation based methods** (Castelfranchi, Parsons, McBurney and others): using logical statements and dialogue games to force agents to reach consensus.

Contract-Net

Task-sharing protocol for task allocation

- A *contract net* is a network of agents that can act as managers or contractors
  - Or even both
- Five steps:
  1. Recognition
  2. Announcement
  3. Bidding
  4. Awarding
  5. Expediting



Contract-Net

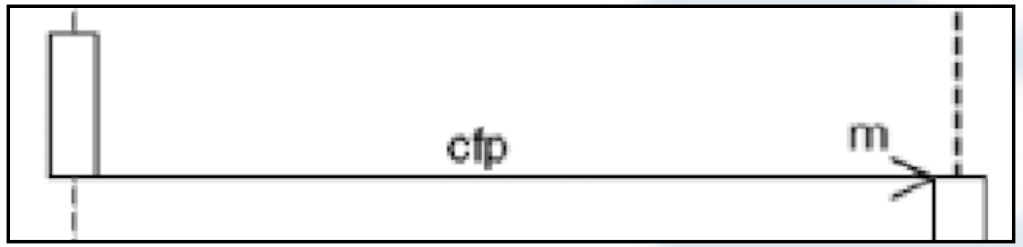
Step 1: Recognition

- The agent recognises it has a problem it needs help with
- The agent identifies a goal or intention, and either
  - It lacks the capabilities necessary for it
  - It has a lack of resources
- The tasks needed to accomplish the goals are identified by the agent

Contract-Net

Step 2: Announcement

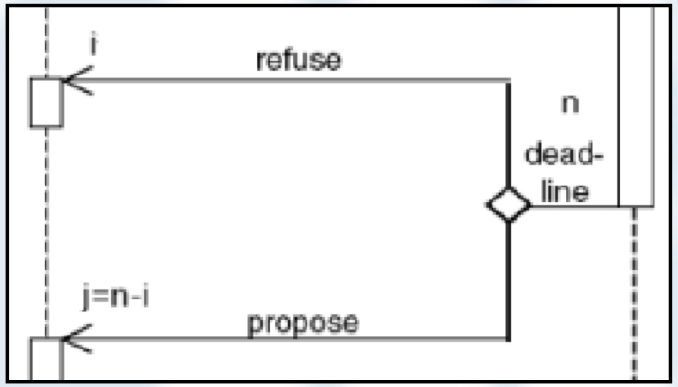
- The agent broadcasts one of the tasks identified, including a specification:
  - Description of the task
  - The task constraints (such as deadlines or QoS)
  - Meta-task information (such as preferences or compensations)



Contract-Net

Step 3: Bidding

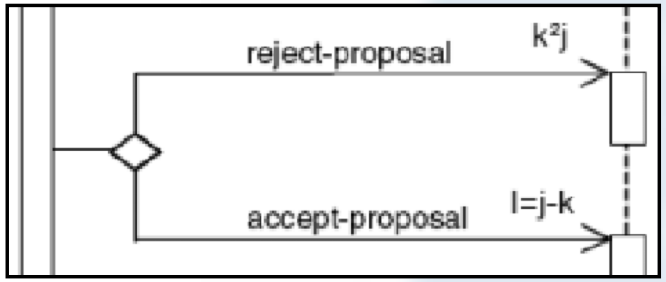
- The participants that receive the task specification have an amount of time to decide if they want to offer themselves for carrying out the task
- Participants should take into account self-interest factors: capabilities, cost, benefit
- Interested participants commit and send a tender, specifying the conditions under which the task is promised to be fulfilled



Contract-Net

Step 4: Awarding

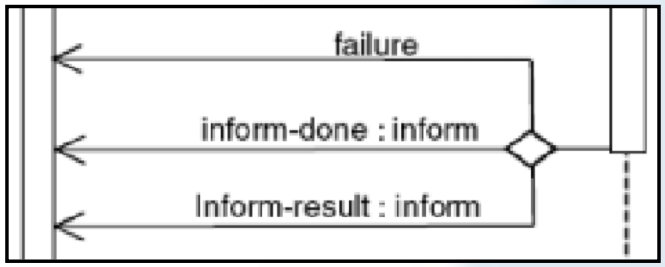
- The initiator agent makes a choice among the tenders received
- The participants are communicated of that choice in what counts as a contract (or lack of)



Contract-Net

Step 5: Expediting

- The contractor sees to it that the task is fulfilled
- Sub-contracting is possible



Contract-Net

Technical Considerations

- Implementing a Contract-Net system requires
  - A task specification
  - A definition of quality of service
  - An algorithm for selection among competing offers
  - Algorithms for creating offers
  - A way to generate tasks from goals and decompose them
- Generally implemented ad hoc, but there are some extensions in general-purpose frameworks:
  - JADE
  - Jason
  - jBPM

Contract-Net

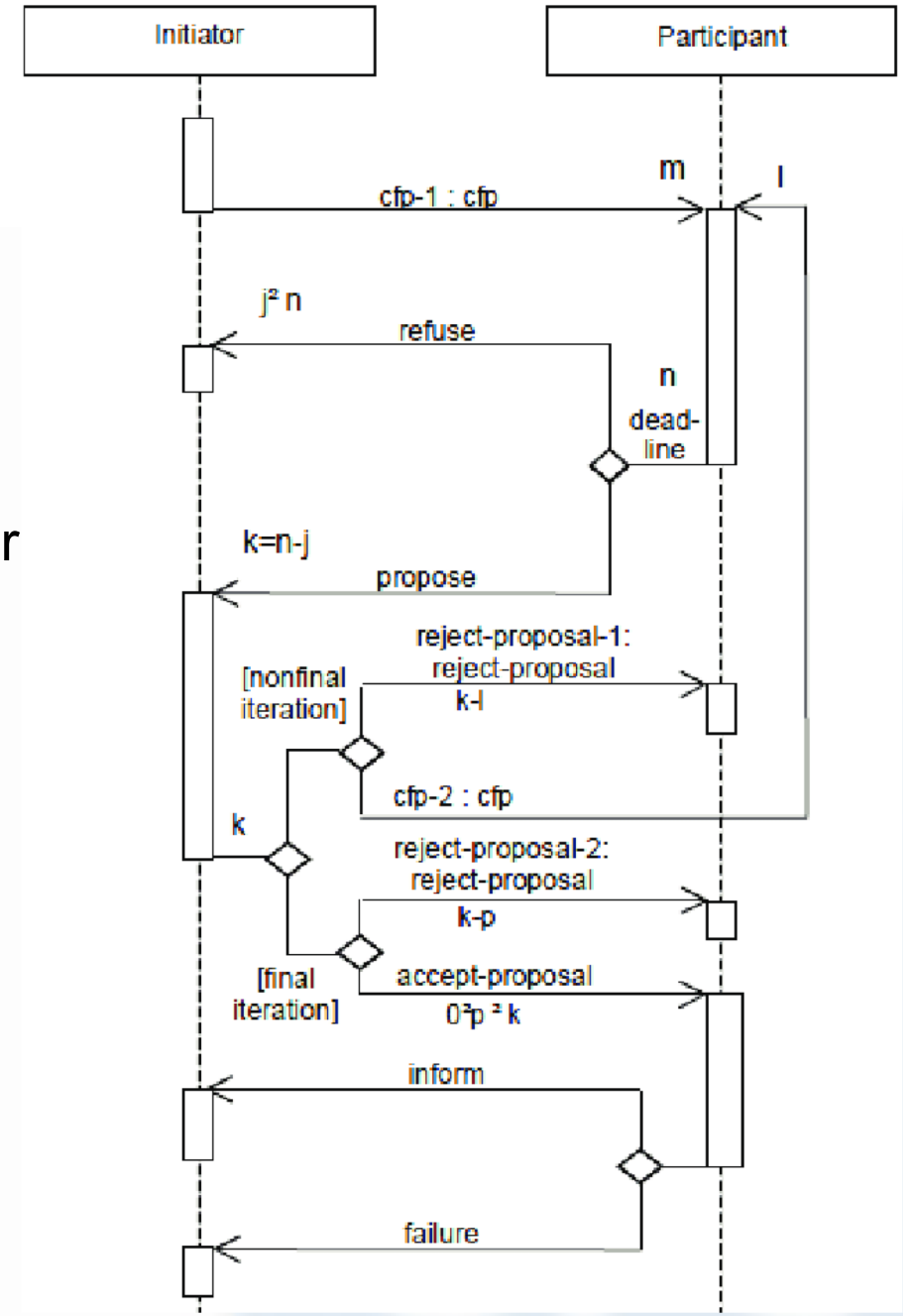
Variations

- Competitive Contract-Net [Vokrínek, Bíba, Pechoucek 2007], with 3 phases:
  - Contracting
  - Decommitment (breach of contract, penalties, bilateral cancellations)
  - Termination (evaluation of QoS)
- Contract-Net with Confirmation (CNCP) [Knabe, Schillo, Fischer 2002], in which the commitment is delayed as much as possible
  - Sending a tender does not count as commitment
  - The initiator asks for commitment to the participants one by one, in order of preference

Contract-Net

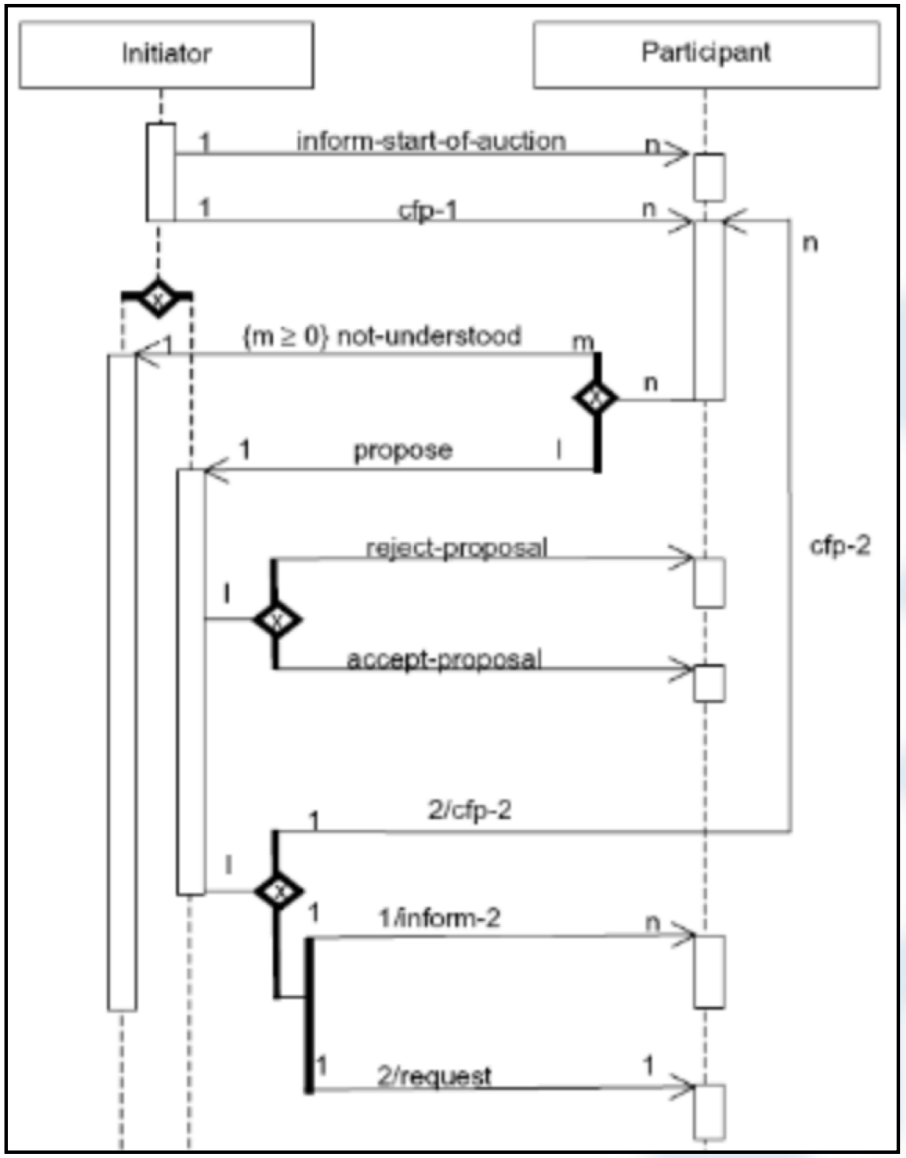
Variations

- Iterated Contract-Net Protocol (FIPA)
  - The initiator can review the call-for-proposals after receiving the proposals



Auctions

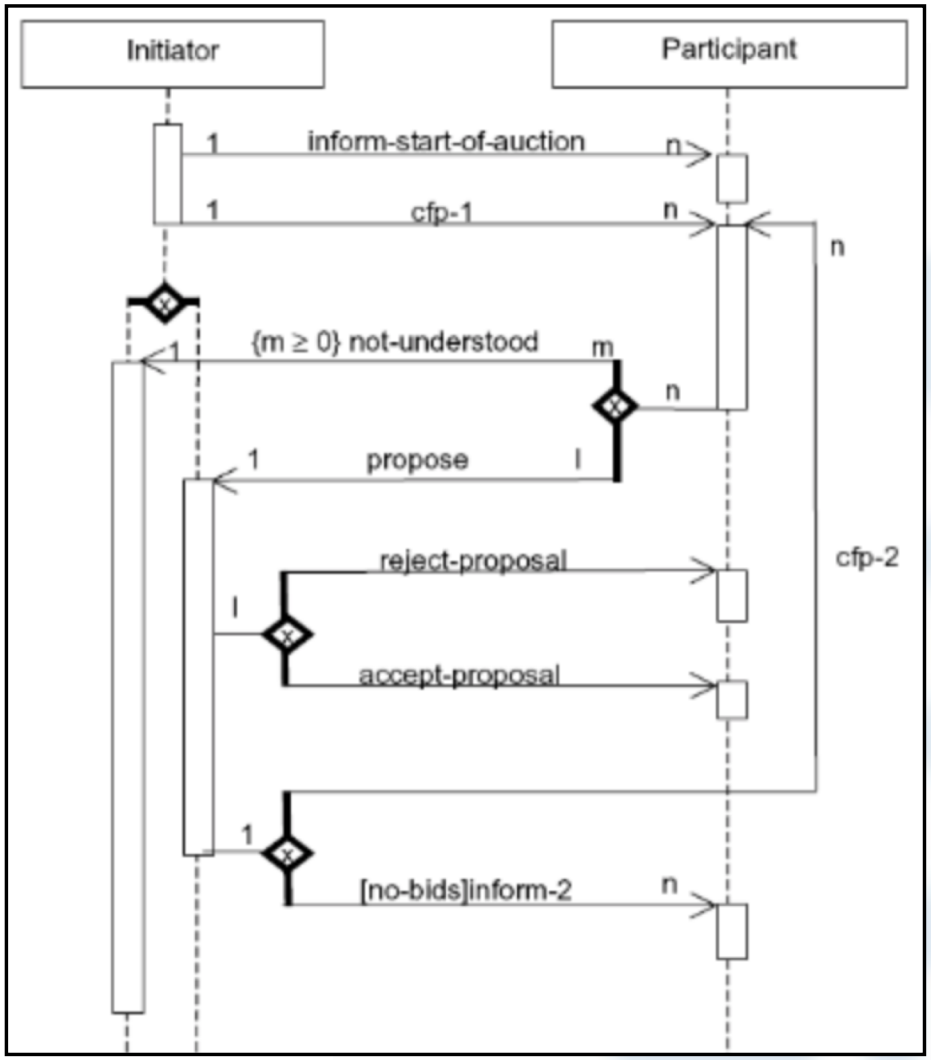
Other FIPA interaction protocols for coordination



English auction

Auctions

Other FIPA interaction protocols for coordination



Dutch auction

Negotiation

Fatio – McBurney and Parsons

- Classification of Speech Acts (Austin, Searle, Habbermas):
  - Factual
  - Expressive
  - Social Connection
  - Commissives
  - Directives
  - Inferences
  - Argumentation
  - Control
- Locutions have different effects

Locution Type	FIPA ACL Locutions
Factual Statements	confirm disconfirm failure inform inform-if inform-ref query-if query-ref
Expressive Statements	inform
Social Connection Statements	inform
Commissives	accept-proposal agree propose refuse reject-proposal
Directives	cancel cfp request request-when request-whenever
Inferences	inform
Argumentation Statements	
Control Statements	not-understood propagate proxy subscribe

From McBurney and Parsons 2004

Negotiation

Fatio – McBurney and Parsons

- Fatio defines 5 illocutions:
  - *assert*(P_i, φ)
  - *question*(P_j, P_i, φ)
  - *challenge*(P_j, P_i, φ)
  - *justify*(P_i, $\Phi \vdash^{+} \varphi$)
  - *retract*(P_i, φ)

R: The Brigade Restaurant is excellent.
P: I had a great meal at the Brigade.
Q: I am vegetarian.
S: Vegetarian food at the Brigade is awful.

1. *assert(A, R)*
   Agent A asserts *R*.
2. *challenge(B, A, R)*
   Agent B issues a challenge to A for a justification for *R*.
3. *justify(A, $P \vdash^{+} R$)*
   Agent A provides a justification for *R*, namely *P*.
4. *question(C, B, R)*
   Agent C questions B over her challenge to A, seeking a justification for the challenge.
5. *justify(B, Q & $S \vdash^{-} R$)*
   Agent B provides a justification for her challenge to A, namely that *Q* and *S*.
6. *retract(A, R)*
   Hearing this, Agent A retracts his prior assertion of *R*.
7. *question(C, B, Q)*
   Agent C asks B to justify her argument of *Q*.

From McBurney and Parsons 2004

Negotiation

Fatio – McBurney and Parsons

- Taking an approach like this:
  - Makes it possible to specify and build the agent reasoning elements
  - Makes it possible to build open-ended coordination protocols
  - Makes it possible to plug new agents (possibly built by different people) straight into the environment
- Fatio is just an example – focuses on fact / action based negotiation using argumentation.

Speech Act Based Coordination

Explicit coordination

- Messages in a negotiation or any other explicit coordination have a meaning – they imply things such as:
  - A commitment to act
  - The acceptance of a fact
  - Information about an outcome
  - ...
- Explicit semantics are needed for agents to “understand” these messages.
- Hence explicit coordination can be seen as **language** or **interaction design**.

Speech Act Based Coordination

Methods for speech act based coordination

- To achieve this interaction design there have been three families of approaches:
  - Definition of the semantics of **communication primitives** (Lux, Steiner, FIPA): focusing on the definition of meaning of individual speech act (inform, accept, etc.)
  - Definition of specific **coordination languages** (e.g. COOL): which focus only on the expression of joint action and specifically representing actions to be carried out.
  - Definition of ***coordination protocols*** (Pitt, Burmeister and others): which argues that individual speech acts have no strong semantics outside the context of a dialogue.

References

1. Lamport, Leslie. "Paxos made simple." *ACM Sigact News* 32.4 (2001): 18-25.
2. Maheswaran, Rajiv T., Jonathan P. Pearce, and Milind Tambe. "Distributed Algorithms for DCOP: A Graphical-Game-Based Approach." *ISCA PDCS*. 2004.
3. Beer, Martin, et al. "Negotiation in multi-agent systems." *The Knowledge Engineering Review* 14.3 (1999): 285-289.
4. Smith, Reid G. "The contract net protocol: High-level communication and control in a distributed problem solver." *IEEE Transactions on computers* 12 (1980): 1104-1113.
5. FIPA. "FIPA Iterated Contract Net Interaction Protocol Specification." (2001).
6. Bozdag, Engin. "A survey of extensions to the contract net protocol." *Technical report, CiteSeerX-Scientific Literature Digital Library and Search Engine* (2008).
7. McBurney, Peter, and Simon Parsons. "Locutions for argumentation in agent interaction protocols." *International Workshop on Agent Communication*. Springer, Berlin, Heidelberg, 2004.