

Coordination in Distributed Systems

SID 2021/22q2

Sergio Álvarez

Cooperation

Consensus

DCOP

Scenario

- We have a system of N drones with wireless communication capabilities up to a certain range of x kilometers
 - Individual units can be specialized (e.g. better camera, higher water tank capacity, faster speed, ...)
 - Each action (including communication) requires an amount of energy
 - Each drone has a variable amount of available energy

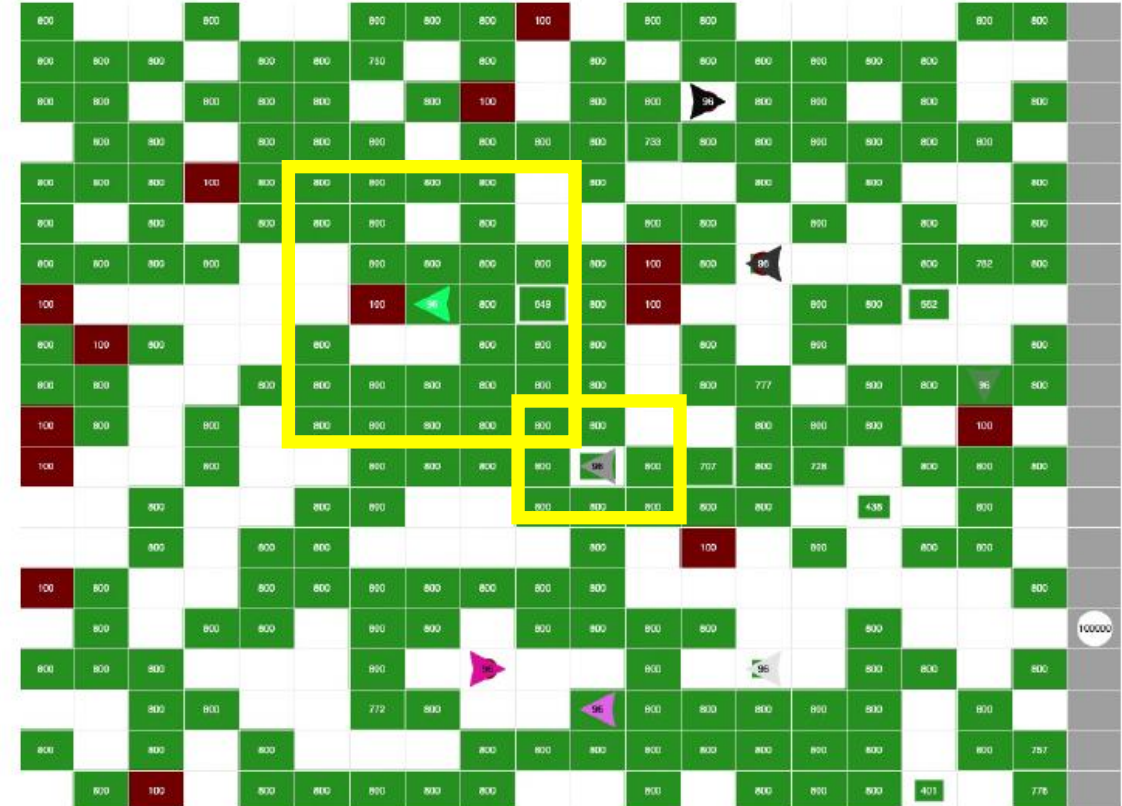


- There is additionally a regular computer at a fixed location that may be used or not
- They have to coordinate to fight fire spreading on a forest (simplified by a discrete map)

Problem definition

- We want the drones to allocate a sector to scan and extinguish
 - The visibility capabilities are variable among drones so the sectors are decided by each agent
- Let $A=\{A_1, ..., A_n\}$ be the set of **n drones**
 - each agent A_i can select the area to scan and extinguish among **k sectors** $s_i=\{s^1_i, ..., s^k_i\}$
 - A_i , at a given point of time, will be assigned a particular s^j_i
- Let w be a function over pairs of sectors s^p_i, s^q_j where
 - $w(s^p_i, s^q_j)$ is the surface of the area common to these two sectors

- Communication might fail dynamically
- We want to minimize the sum of overlaps (w)



Consensus Model

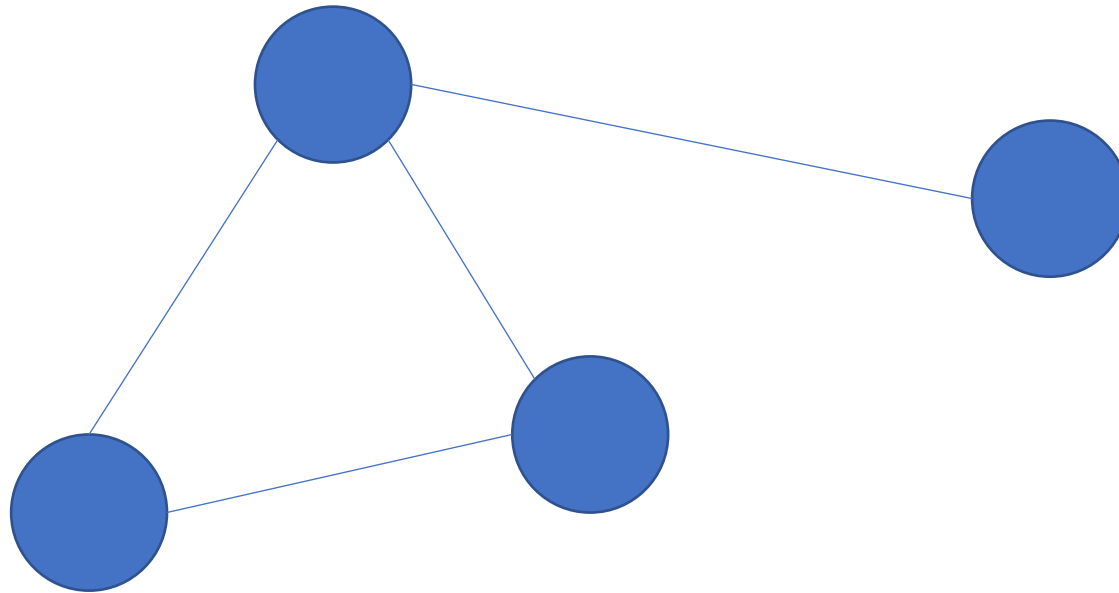
- What is the value of each decision?
- Who are...?
 - Proposers
 - Acceptors
 - Learners

Consensus Model

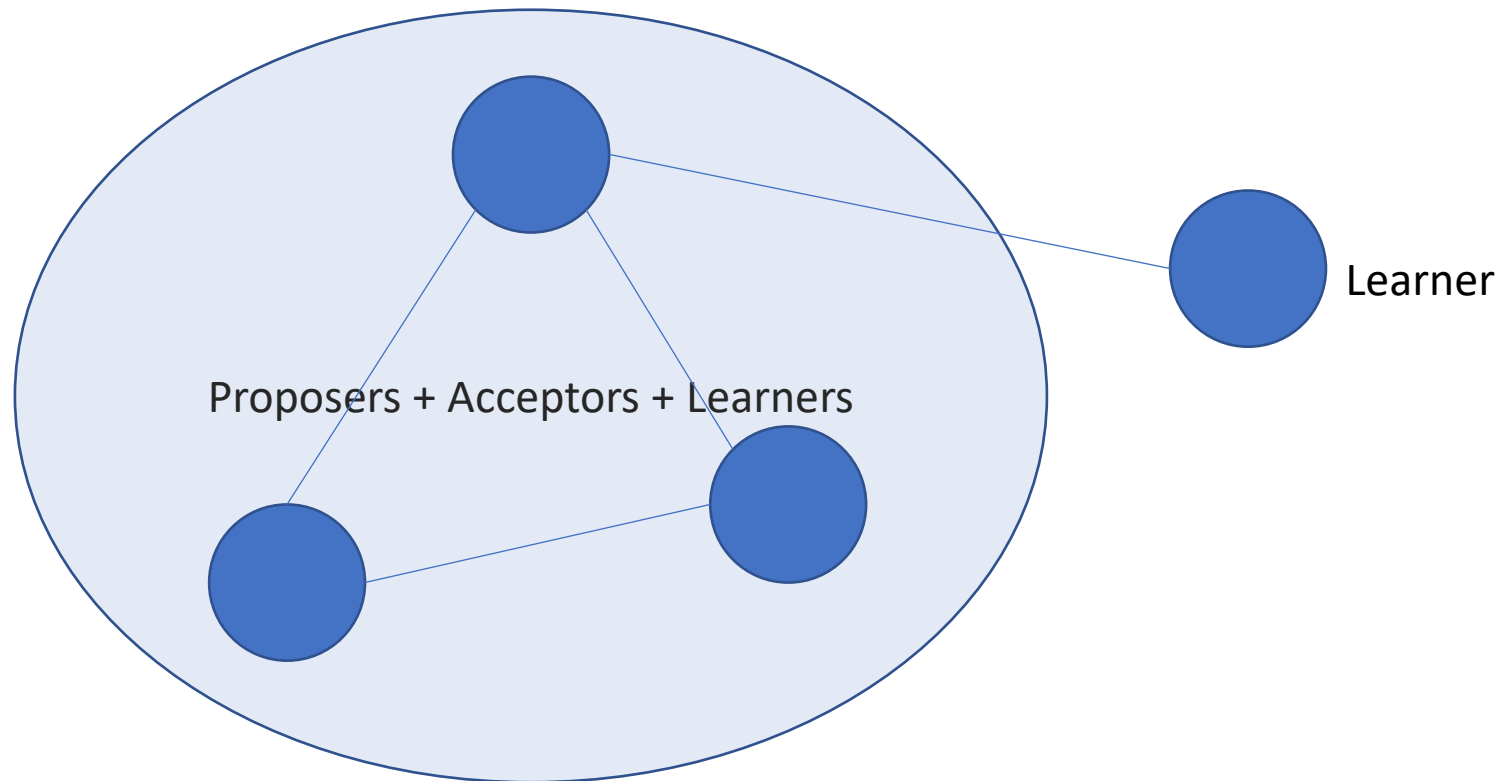
- What is the value of each decision?
 - Either an agent descriptor (the leader)
 - Or the complete allocation itself
- Who are...?
 - Proposers
 - Acceptors
 - Learners
 - Will depend on the difference in capabilities of agents
 - If all agents are equal, all of them can enact the three roles
 - Also depends on the communication topology
 - Agents that cannot guarantee total communication should be learners only

Consensus Model (example)

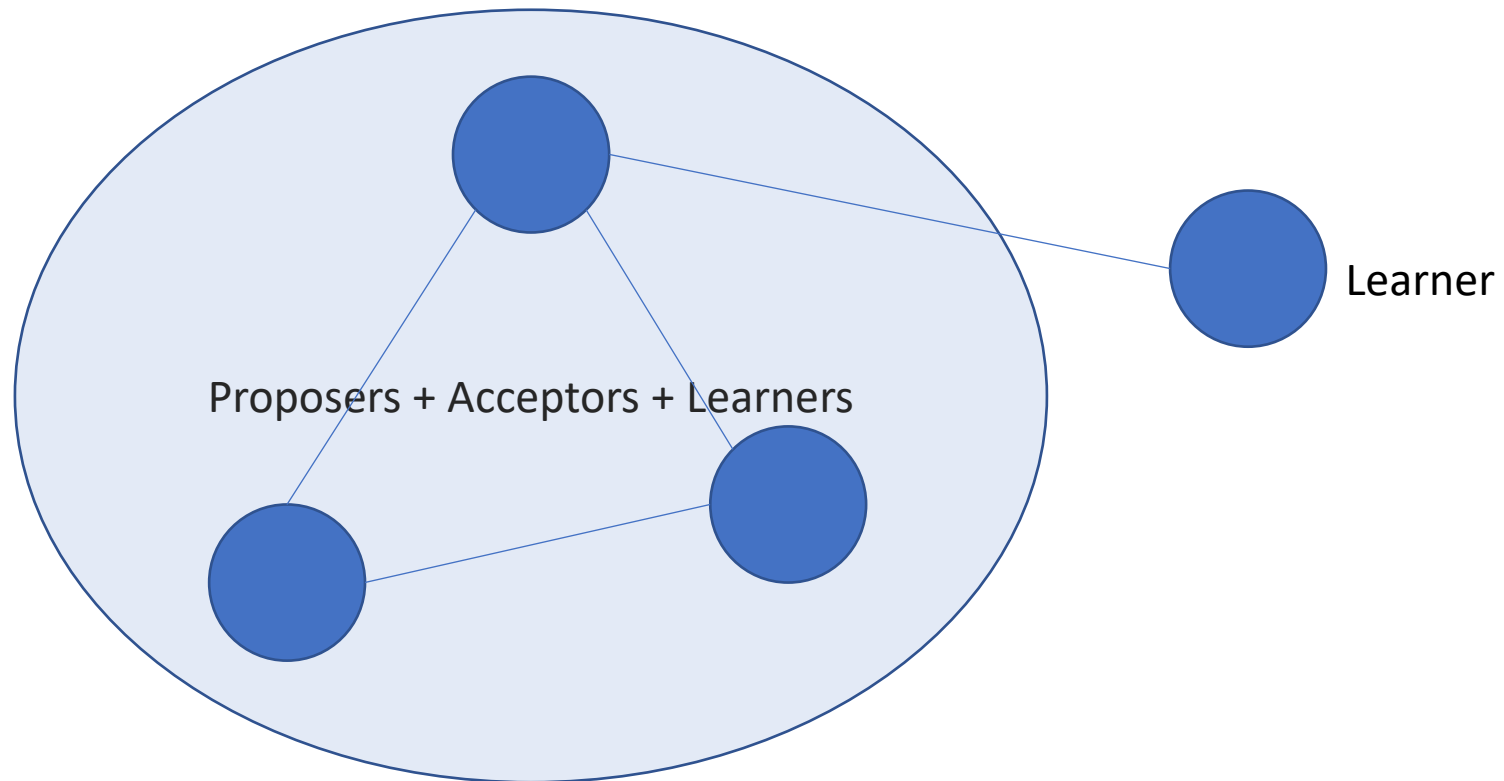
- Assuming all agents equal and no central computer



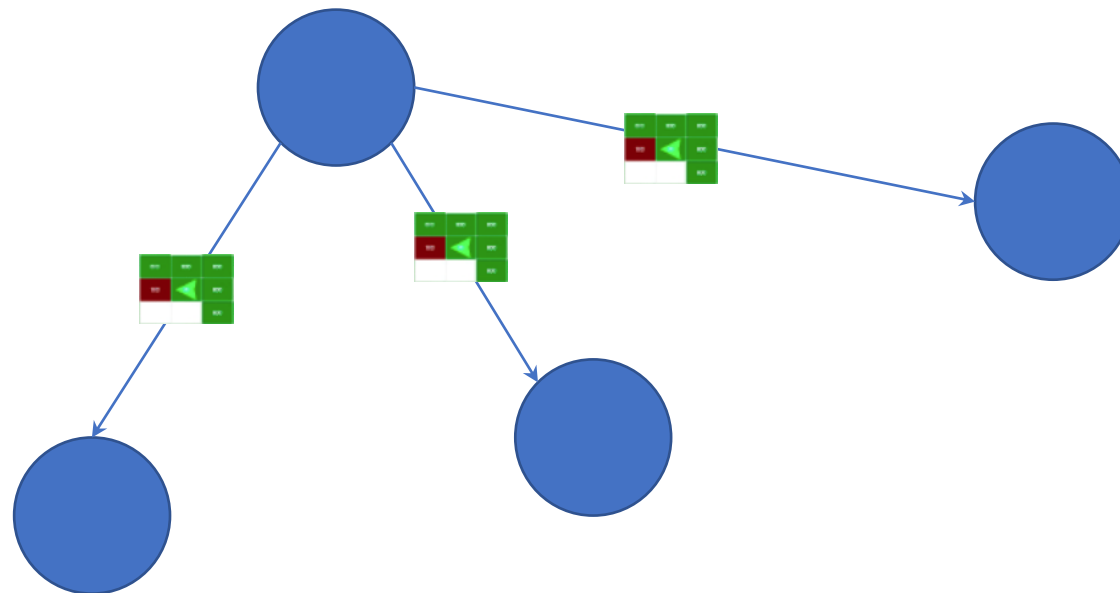
Consensus Model (example)



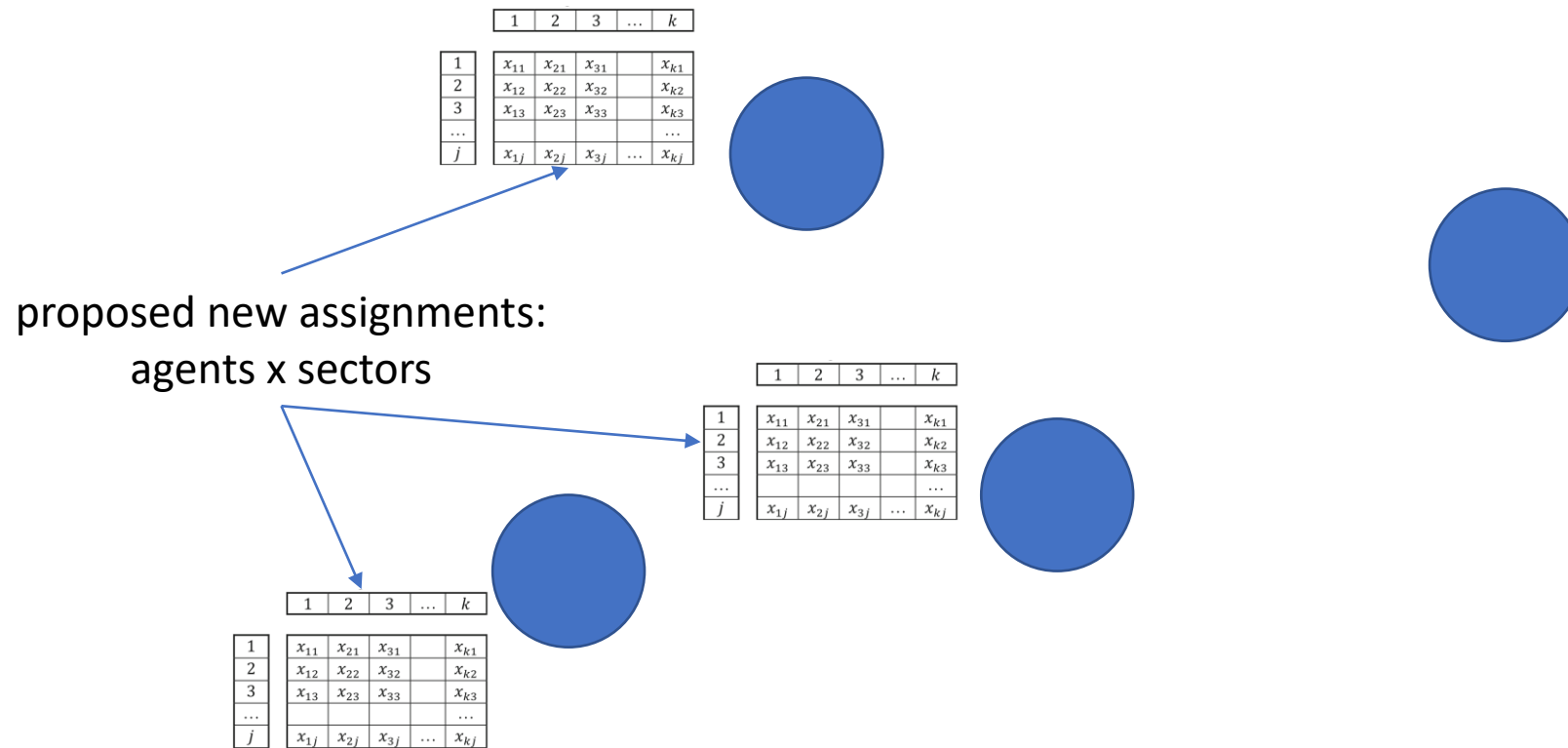
Consensus Model (example)



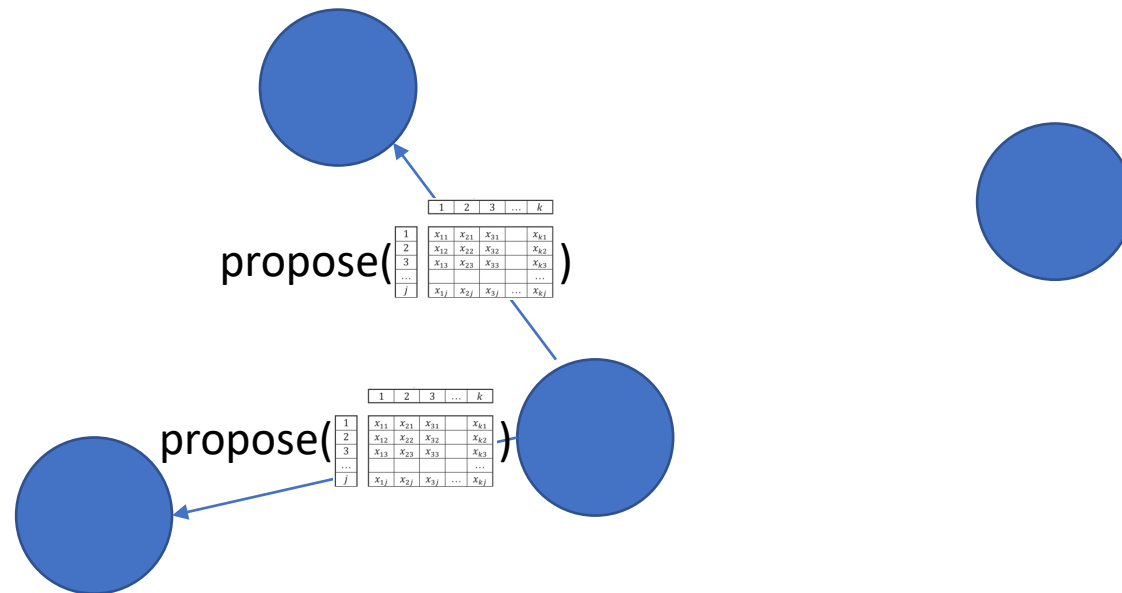
Consensus Model (example)



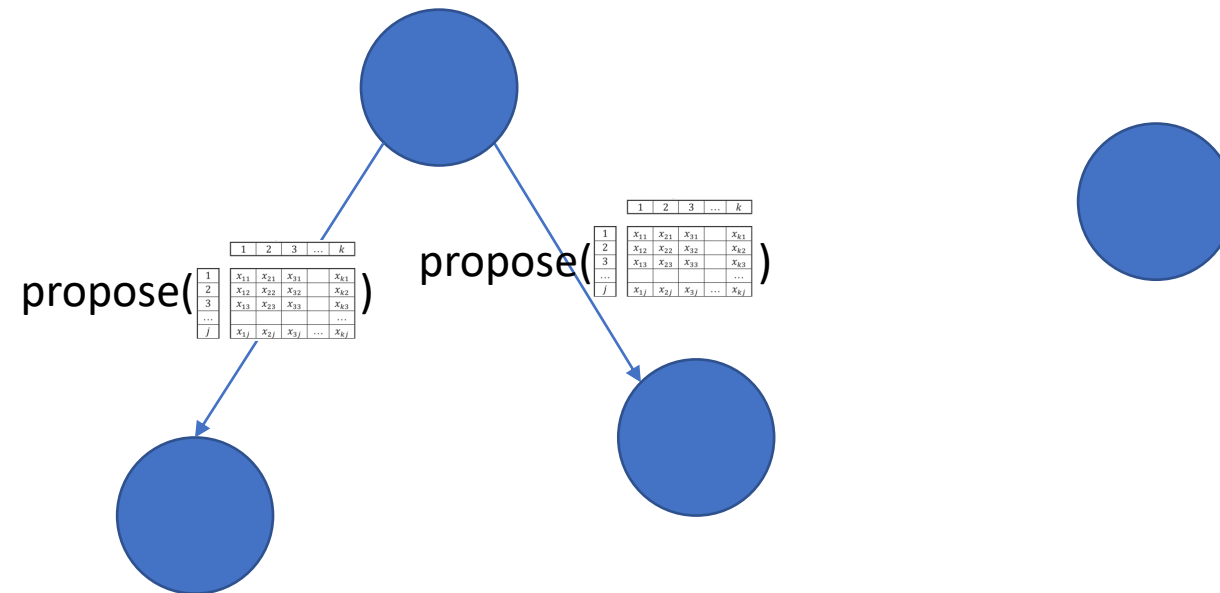
Consensus Model (example)



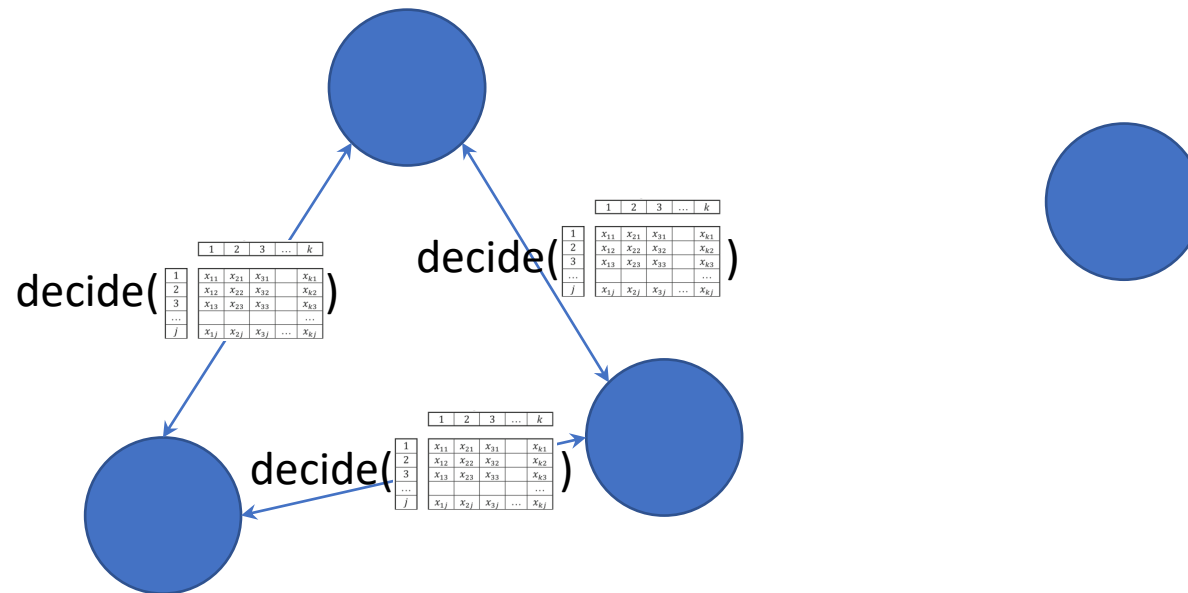
Consensus Model (example)



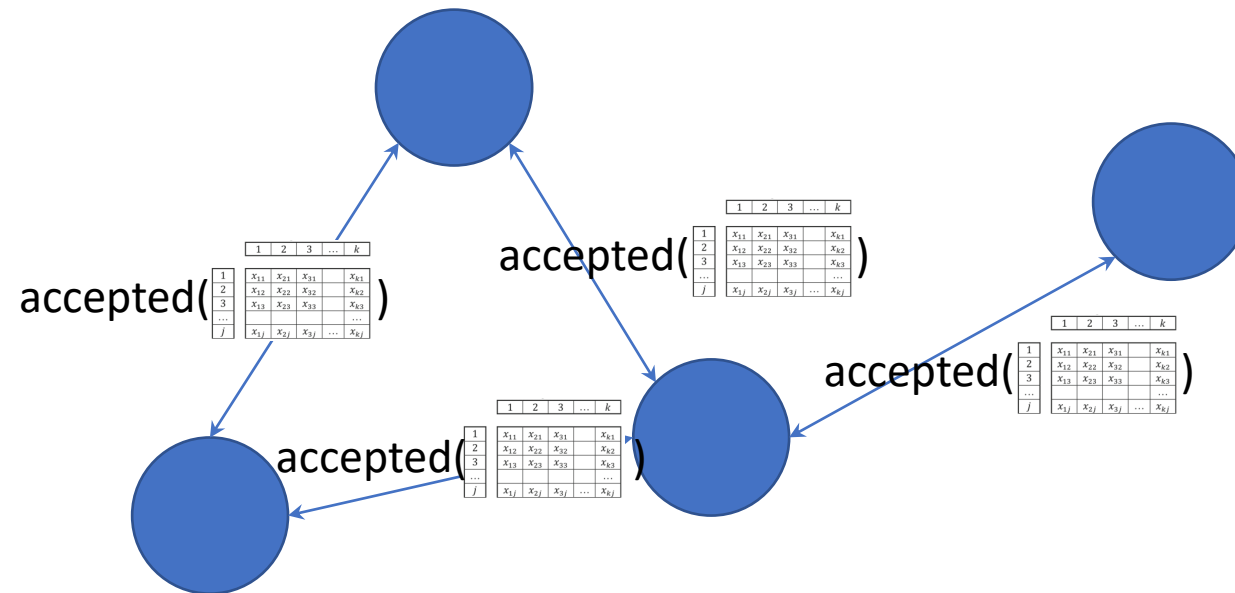
Consensus Model (example)



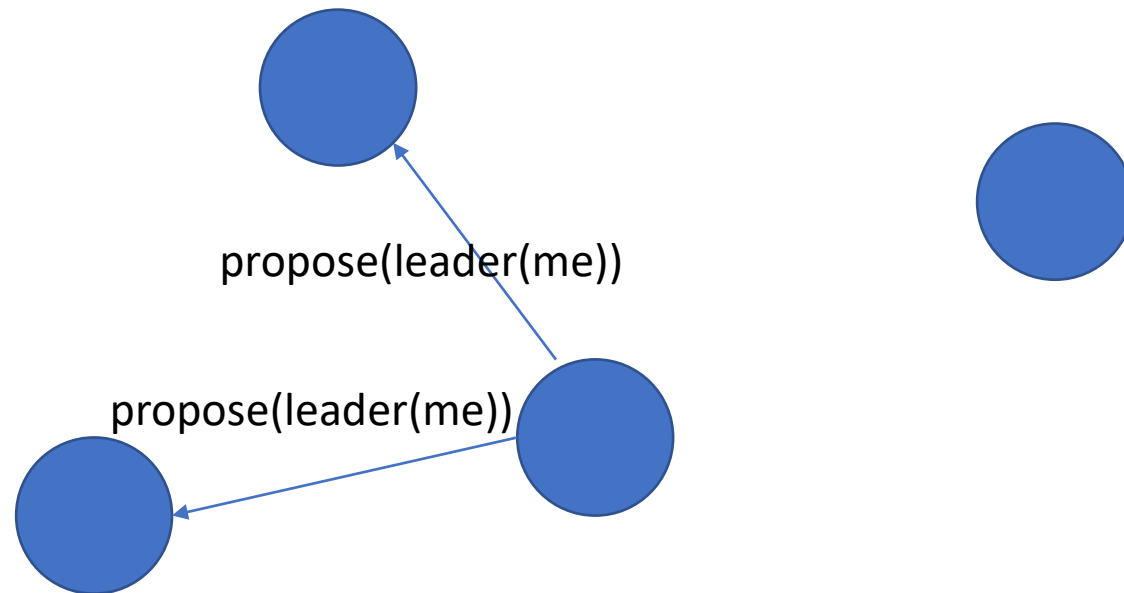
Consensus Model (example)



Consensus Model (example)



Consensus Model (example)



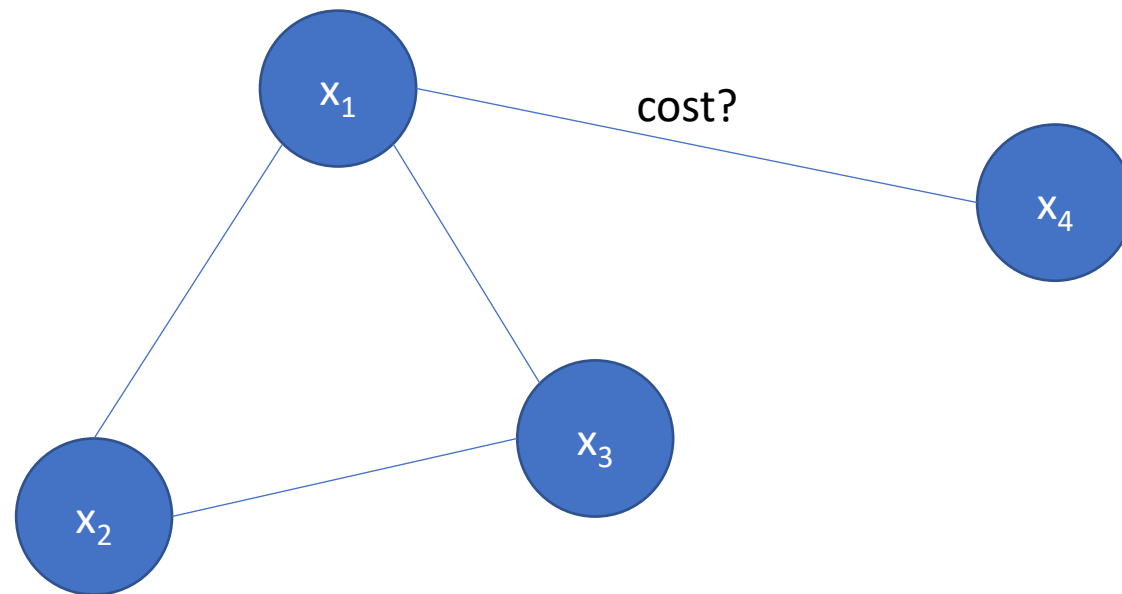
DCOP Model

- What are the variables?
- What are the domains of the variables?
- What is the cost function?
- What is the cost aggregation function?

DCOP Model

- What are the variables?
 - All the agents with computational and communication capabilities
- What are the domains of the variables?
 - Assignment of a specific sector
- What is the cost function?
 - Overlapping function w
- What is the cost aggregation function?
 - Sum / Average / Deviation / Entropy...

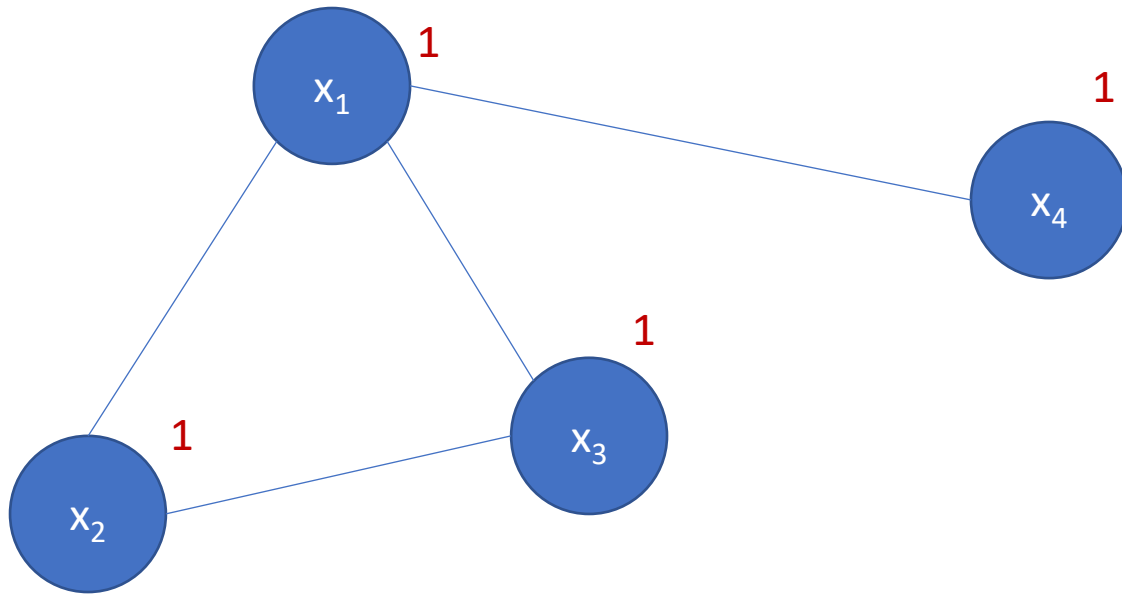
DCOP Model (example)



DCOP Model

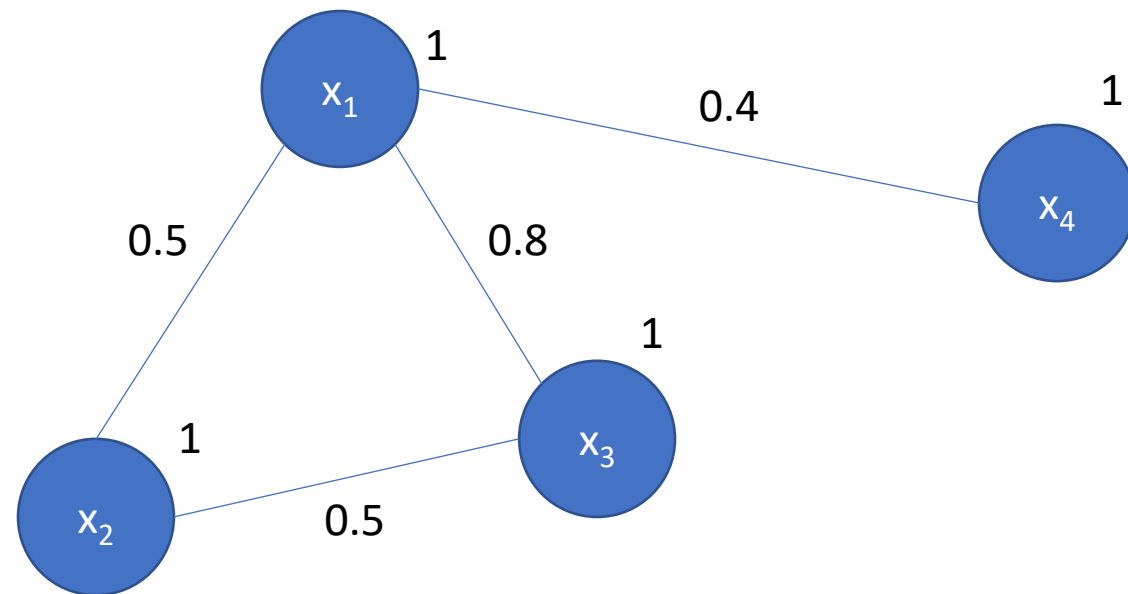
w	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	1	0	0	0.5	0.1	0.8	1	0.4
s_1^2	0	1	0	0	0.3	0	0	0.6
s_1^3	0	0	1	0.9	0.6	0.6	0.2	0.5
s_2^1	0.5	0	0.9	1	0	0.5	0	0.7
s_2^2	0.1	0.3	0.6	0	1	0.3	0.1	0.2
s_3^1	0.8	0	0.6	0.5	0.3	1	0	0.1
s_3^2	1	0	0.2	0	0.1	0	1	0
s_4^1	0.4	0.6	0.5	0.7	0.2	0.1	0	1

DCOP Model (example)



w	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	1	0	0	0.5	0.1	0.8	1	0.4
s_1^2	0	1	0	0	0.3	0	0	0.6
s_1^3	0	0	1	0.9	0.6	0.6	0.2	0.5
s_2^1	0.5	0	0.9	1	0	0.5	0	0.7
s_2^2	0.1	0.3	0.6	0	1	0.3	0.1	0.2
s_3^1	0.8	0	0.6	0.5	0.3	1	0	0.1
s_3^2	1	0	0.2	0	0.1	0	1	0
s_4^1	0.4	0.6	0.5	0.7	0.2	0.1	0	1

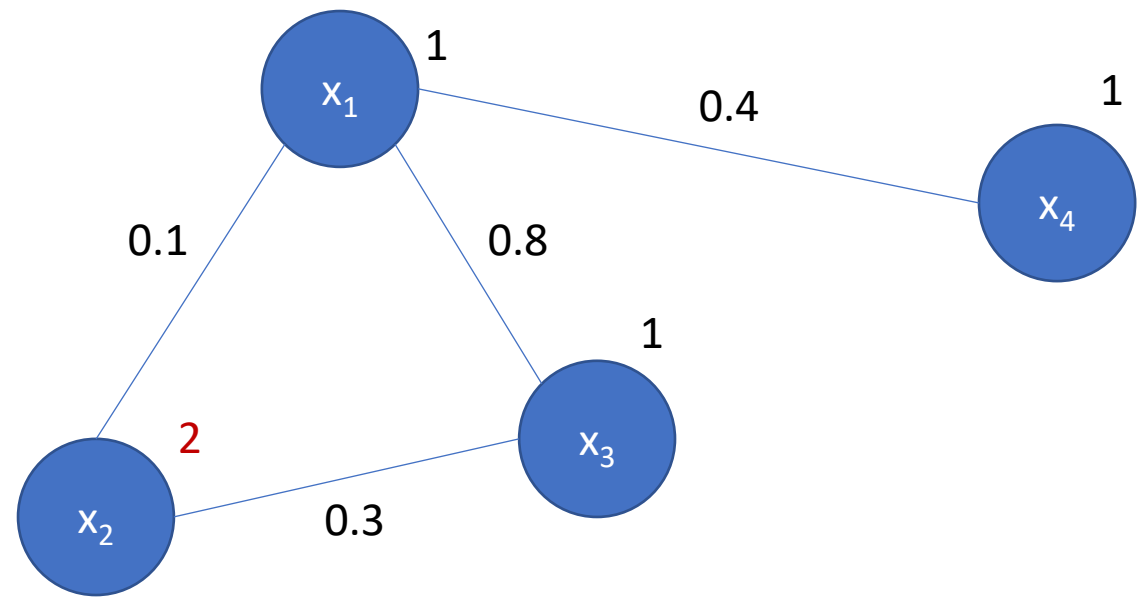
DCOP Model (example)



$F = 2.2$

w	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	1	0	0	0.5	0.1	0.8	1	0.4
s_1^2	0	1	0	0	0.3	0	0	0.6
s_1^3	0	0	1	0.9	0.6	0.6	0.2	0.5
s_2^1	0.5	0	0.9	1	0	0.5	0	0.7
s_2^2	0.1	0.3	0.6	0	1	0.3	0.1	0.2
s_3^1	0.8	0	0.6	0.5	0.3	1	0	0.1
s_3^2	1	0	0.2	0	0.1	0	1	0
s_4^1	0.4	0.6	0.5	0.7	0.2	0.1	0	1

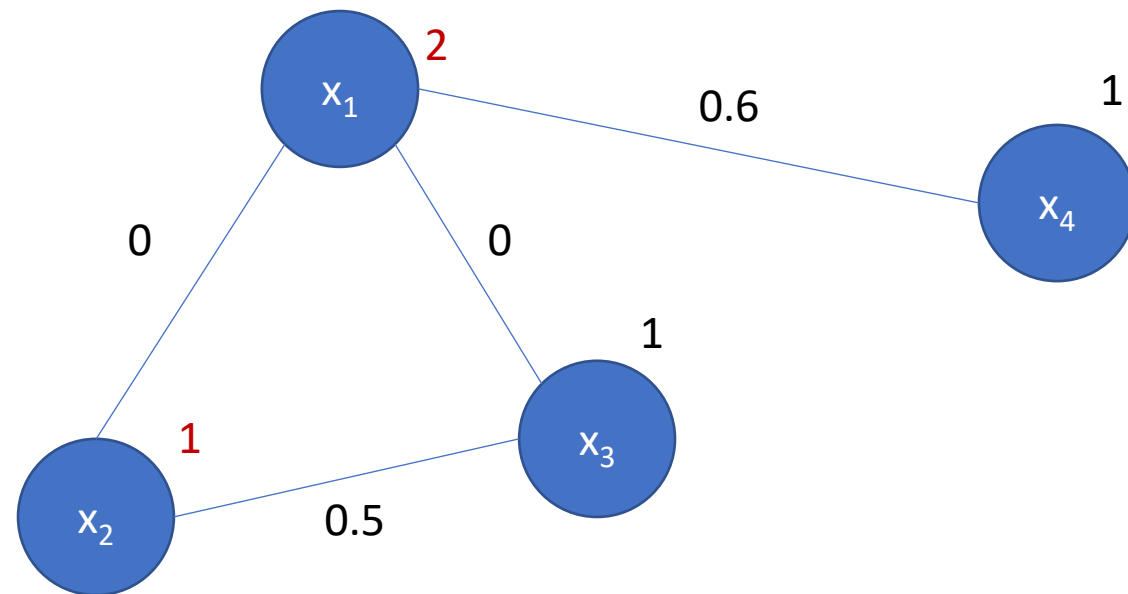
DCOP Model (example)



$F = 1.6$

w	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	1	0	0	0.5	0.1	0.8	1	0.4
s_1^2	0	1	0	0	0.3	0	0	0.6
s_1^3	0	0	1	0.9	0.6	0.6	0.2	0.5
s_2^1	0.5	0	0.9	1	0	0.5	0	0.7
s_2^2	0.1	0.3	0.6	0	1	0.3	0.1	0.2
s_3^1	0.8	0	0.6	0.5	0.3	1	0	0.1
s_3^2	1	0	0.2	0	0.1	0	1	0
s_4^1	0.4	0.6	0.5	0.7	0.2	0.1	0	1

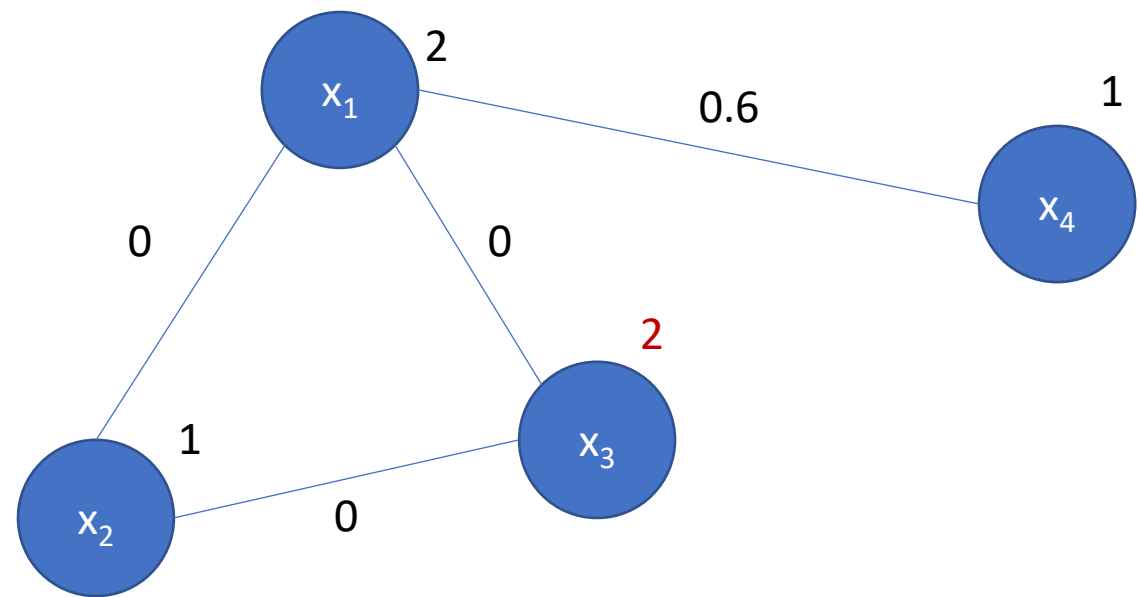
DCOP Model (example)



$F = 1.1$

w	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	1	0	0	0.5	0.1	0.8	1	0.4
s_1^2	0	1	0	0	0.3	0	0	0.6
s_1^3	0	0	1	0.9	0.6	0.6	0.2	0.5
s_2^1	0.5	0	0.9	1	0	0.5	0	0.7
s_2^2	0.1	0.3	0.6	0	1	0.3	0.1	0.2
s_3^1	0.8	0	0.6	0.5	0.3	1	0	0.1
s_3^2	1	0	0.2	0	0.1	0	1	0
s_4^1	0.4	0.6	0.5	0.7	0.2	0.1	0	1

DCOP Model (example)



$F = 0.6$

w	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	1	0	0	0.5	0.1	0.8	1	0.4
s_1^2	0	1	0	0	0.3	0	0	0.6
s_1^3	0	0	1	0.9	0.6	0.6	0.2	0.5
s_2^1	0.5	0	0.9	1	0	0.5	0	0.7
s_2^2	0.1	0.3	0.6	0	1	0.3	0.1	0.2
s_3^1	0.8	0	0.6	0.5	0.3	1	0	0.1
s_3^2	1	0	0.2	0	0.1	0	1	0
s_4^1	0.4	0.6	0.5	0.7	0.2	0.1	0	1

Distributed Planning Model

- What is the shared goal state?
- What are the types?
- What are the predicates?
- What predicates can/should be private?
- What are the operators?

Distributed Planning Model

- What is the shared goal state?
 - A target assignment for each agent
- What are the types?
 - Drone, sector
- What are the predicates?
 - Target assignment, sector overlapping, sector “ownership”
- What predicates can/should be private?
 - ?
- What are the operators?
 - Assign target

Distributed Planning Model

- What is the shared goal state?
 - A target assignment for each agent
- What are the types?
 - Drone, sector
- What are the predicates?
 - Target assignment, sector overlapping, sector “ownership”
 - Current sector, sector accessibility
- What predicates can/should be private?
 - Current sector, sector accessibility
- What are the operators?
 - Assign target, move to accessible sector

Distributed Planning Model (example)

- Assuming we don't have access to fluents

acc?	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	F	T	F	T	T	F	F	T
s_1^2	T	F	T	T	T	T	T	T
s_1^3	F	T	F	F	T	T	T	T
s_2^1	T	T	F	F	T	T	T	F
s_2^2	T	T	T	T	F	T	T	T
s_3^1	F	T	T	T	T	F	T	T
s_3^2	F	T	T	T	T	T	F	T
s_4^1	T	T	T	F	T	T	T	F

Distributed Planning Model (example)

(overlapping s11 s11)
(overlapping s12 s12)
(overlapping s13 s13)
(overlapping s21 s21)
(overlapping s22 s22)
(overlapping s31 s31)
(overlapping s32 s32)
(overlapping s41 s41)

(overlapping s11 s31)
(overlapping s11 s32)
(overlapping s13 s21)

acc?	s_1^1	s_1^2	s_1^3	s_2^1	s_2^2	s_3^1	s_3^2	s_4^1
s_1^1	F	T	F	T	T	F	F	T
s_1^2	T	F	T	T	T	T	T	T
s_1^3	F	T	F	F	T	T	T	T
s_2^1	T	T	F	F	T	T	T	F
s_2^2	T	T	T	T	F	T	T	T
s_3^1	F	T	T	T	T	F	T	T
s_3^2	F	T	T	T	T	T	F	T
s_4^1	T	T	T	F	T	T	T	F

Distributed Planning Model (example)

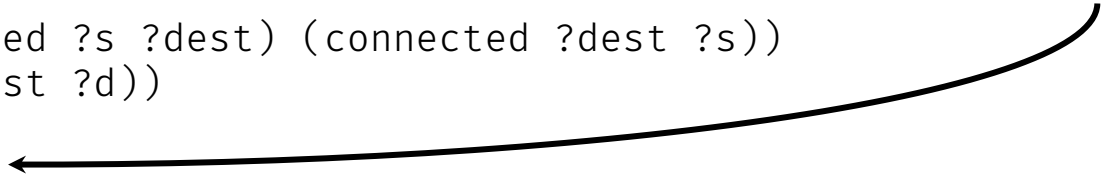
```
(define (domain drones)
  (:requirements :typing :multi-agent :unfactored-privacy)

  (:types
    sector drone - object)

  (:predicates
    (on-target ?d - drone)
    (target-assigned ?d - drone ?s - sector)
    (needs-assignment ?d1 - drone)
    (overlapping ?s1 - sector ?s2 - sector)
    (belongs ?s - sector ?d - drone)
    (:private ?agent - drone
      (at ?d - drone ?s - sector)
      (connected ?s1 - sector ?s2 - sector)))

  (:action go-to-sector
    :agent ?d - drone
    :parameters (?d - drone ?s - sector ?dest - sector)
    :precondition (and (at ?d ?s)
                      (or (connected ?s ?dest) (connected ?dest ?s))
                      (belongs ?dest ?d))
    :effect (and (not (at ?d ?s))
                 (at ?d ?dest)))
```

**At least one effect is
private so the action will
be private**



Distributed Planning Model (example)

```
(:action assign-target  
  :agent ?d1 - drone  
  :parameters (?d1 - drone ?s1 - sector)  
  :precondition (and (needs-assignment ?d1)  
                     (belongs ?s1 ?d1)  
                     (forall (?d2 - drone)  
                       (or (needs-assignment ?d2)  
                           (exists (?s2 - sector)  
                             (and (target-assigned ?d2 ?s2)  
                                  (not (overlapping ?s1 ?s2))  
                                  (not (overlapping ?s2 ?s1)))))))  
  :effect (and (not (needs-assignment ?d1))  
               (target-assigned ?d1 ?s1)))
```

```
(:action detect-on-target  
  :agent ?d - drone  
  :parameters (?d - drone ?s - sector)  
  :precondition (and (not (on-target ?d))  
                     (target-assigned ?d ?s)  
                     (at ?d ?s))  
  :effect (and (on-target ?d)))
```

Public actions

Distributed Planning Model (example)

```
(define (problem drones01)
  (:domain drones)
  (:objects
    x1 x2 x3 x4 - drone
    s11 s12 s13 s21 s22 s31 s32 s41 - sector)
  (:init
    (belongs s11 x1)
    (belongs s12 x1)
    (belongs s13 x1)
    (belongs s21 x2)
    (belongs s22 x2)
    (belongs s31 x3)
    (belongs s32 x3)
    (belongs s41 x4)

    (needs-assignment x1)
    (needs-assignment x2)
    (needs-assignment x3)
    (needs-assignment x4)

    (at x1 s11)
    (at x2 s22)
    (at x3 s31)
    (at x4 s41)

    ; .. Overlappings ..

    (:private x1
      (connected s11 s12)
      (connected s12 s13))

    (:private x2
      (connected s21 s22))

    (:private x3
      (connected s31 s32)))

  (:goal (and
    (on-target x1)
    (on-target x2)
    (on-target x3)
    (on-target x4))))
```

Distributed Planning Model (example)

(at x1 s11)
(at x2 s22)
(at x3 s31)
(at x4 s41)

Public domain +
Public problem



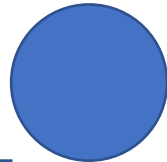
(at x1 s11)
(at x2 s22)
(at x3 s31)
(at x4 s41)

Public domain +
Public problem

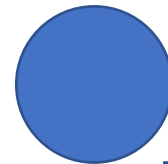


(at x1 s11)
(at x2 s22)
(at x3 s31)
(at x4 s41)

Public domain +
Public problem

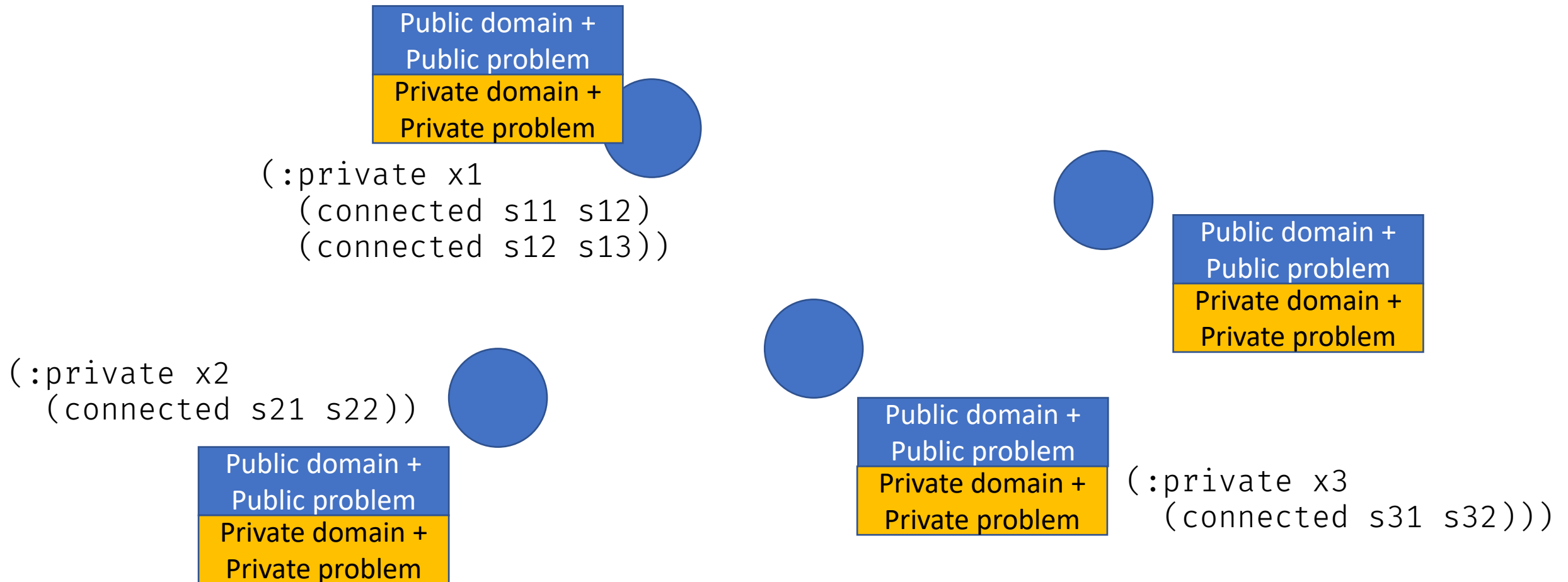


Public domain +
Public problem

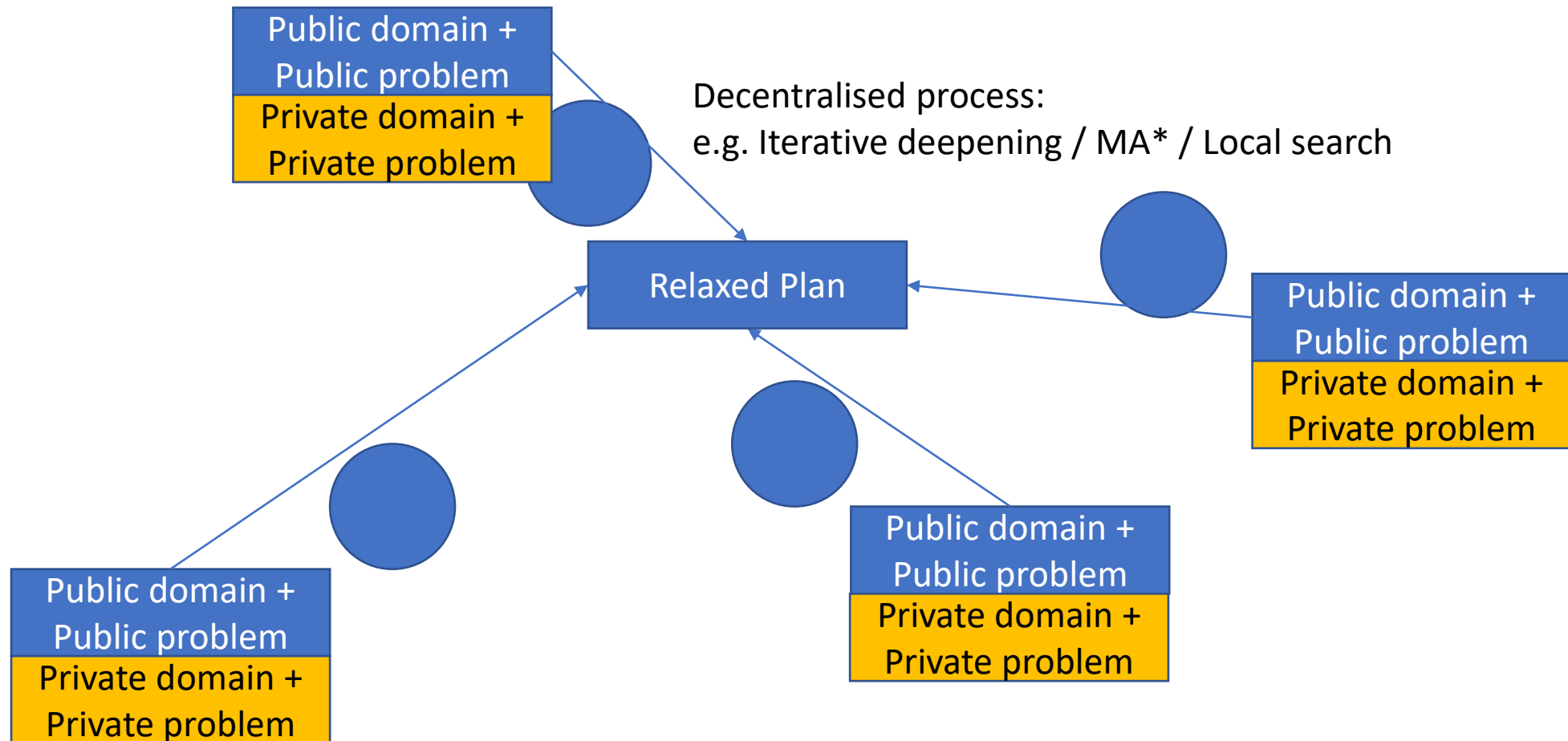


(at x1 s11)
(at x2 s22)
(at x3 s31)
(at x4 s41)

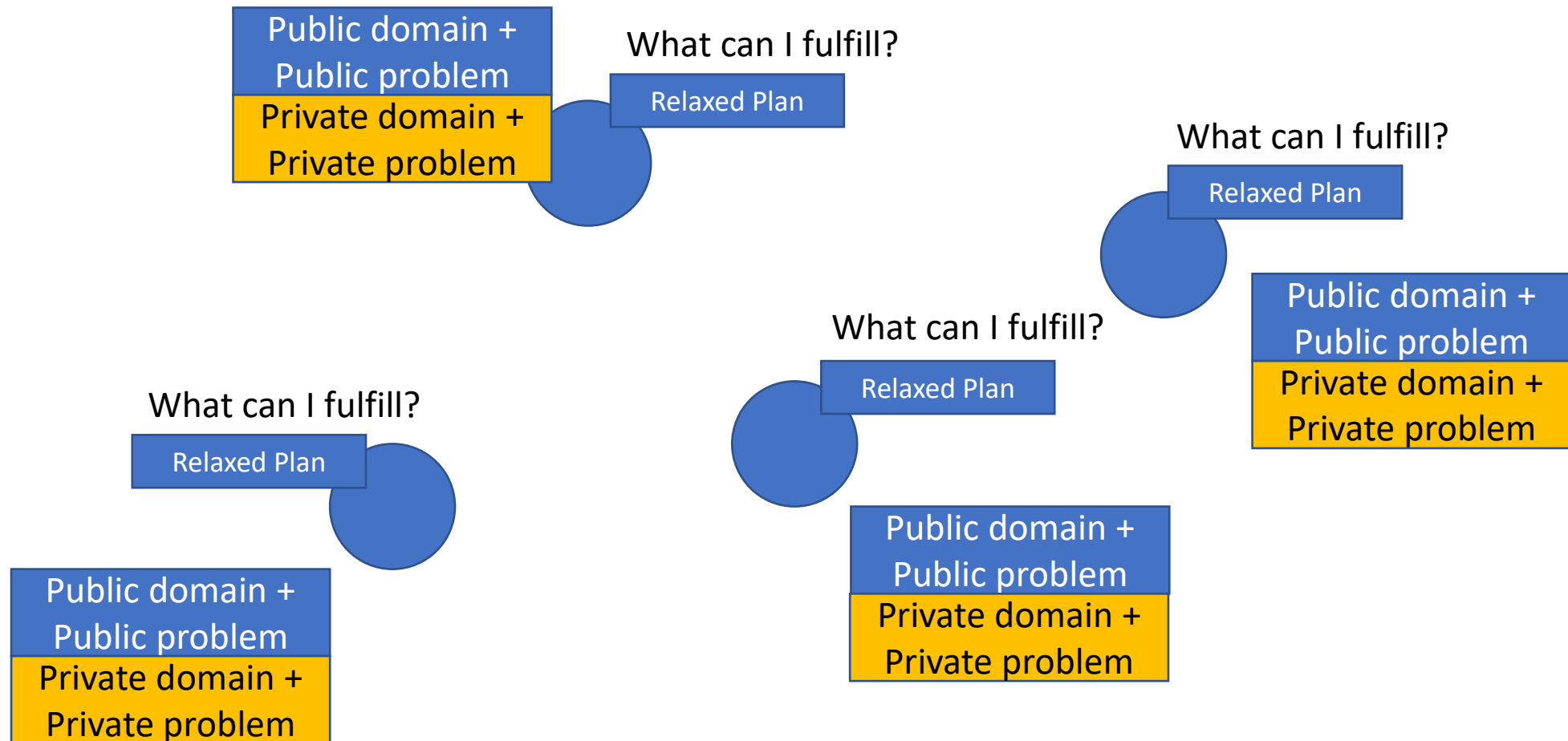
Distributed Planning Model (example)



Distributed Planning Model (example)



Distributed Planning Model (example)



Negotiation

Auction

Scenario

- We have a system of N drones with wireless communication capabilities up to a certain range of x kilometers
 - Individual units can be specialized (e.g. better camera, higher water tank capacity, faster speed, ...)
 - Each action (including communication) requires an amount of energy
 - Each drone has a variable amount of available energy



- There is additionally a regular computer at a fixed location that may be used or not
- They have to coordinate to fight fire spreading on a forest (simplified by a discrete map)
- Each agent has their own preferences (tasks they would prefer to accomplish)

Problem definition

- We want the drones to **agree** on a task allocation
- Let $A = \{A_1, \dots, A_n\}$ be the set of **n drones**
 - each agent A_i can select the area to extinguish among **k sectors** $\{s^1, \dots, s^k\}$
 - A_i , at a given point of time, will be assigned a particular s^j or no task at all
- Let f_i be the utility function of A_i
 - $f_i(s_j)$ is the payoff (value) for agent A_i of being assigned objective s_j
 - f can be dynamic over time and can be seen as **expected reward** (e.g. based on personal preferences, beliefs, intentions)
minus
cost (e.g. distance, time of arrival, energy, extinguishing capabilities)

- Communication might fail dynamically
- We want to reach an equilibrium (if possible, close to Pareto optimality)



How to model the problem?

- It is a mixed motives game:
 - Collective objective: to extinguish the fires
 - Individual objective: to fulfill their own's preferences
- Can it be modelled as
 - a cooperation problem?
 - a negotiation problem?

How to model the problem?

- It is a mixed motives game:
 - Collective objective: to extinguish the fires
 - Individual objective: to fulfill their own's preferences
- Can it be modelled as
 - a cooperation problem?
 - Yes: the agents are not antagonists and there can be a formulation based on a joint goal
 - a negotiation problem?
 - Yes, if we can guarantee convergence and that the agents will adhere to the decisions
- We will see an example based on negotiation (market auctions)

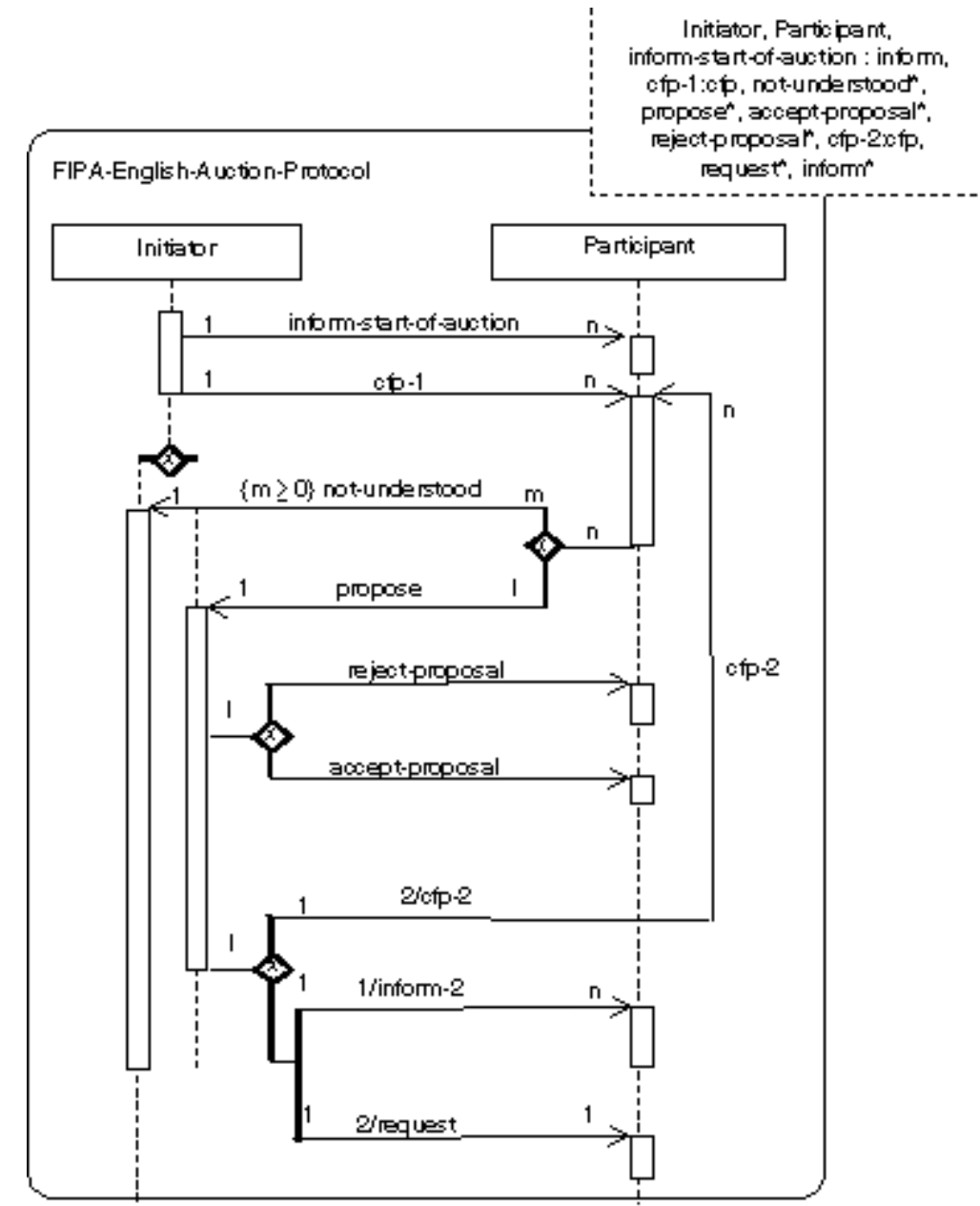
Market Models

- Formalisation of market mechanisms for self-interested agents
- Basic form: seller and buyer negotiating a transaction for a good g
 - The seller has a reserve price R_s , and will accept selling for no less than R_s
 $price(g) \geq R_s$
 - The buyer has a reserve price R_b , and will accept buying for no more than R_b
 $price(g) \leq R_b$
 - Any proposed price p such that $R_s \leq p \leq R_b$ is a valid solution

Auction Models

- Specific form of market mechanism
 - Multi-agent systems of arbitrary size
 - Strict protocol based on asynchronous communication (e.g. FIPA/JADE)
 - Bidding policies are **private** and are determined by each agent
- General form ([more info](#)):
 - We have n agents and n objects
 - For each agent i and object j , a_{ij} is the reward for A_i
 - We want to find a one-to-one assignment $(1, j_1), \dots, (n, j_n)$ such that we maximize:

$$\sum_{i=1}^n a_{ij_i}$$



Auction Models

- Specific form of market mechanism
 - Multi-agent systems of arbitrary size
 - Strict protocol based on asynchronous communication (e.g. FIPA/JADE)
 - Bidding policies are **private** and are determined by each agent
- General form ([more info](#)):
 - We have n agents and n objects
 - For each agent i and object j , a_{ij} is the reward for A_i
 - We want to find a one-to-one assignment $(1, j_1), \dots, (n, j_n)$ such that $\sum_{i=1}^n a_{ij_i}$

- In this general form it is possible to calculate the equilibrium
- An agent is *happy* if the assignment maximizes value where *value* = (*reward* - *price*)

$$a_{ij_i} - p_{j_i} = \max_{j=1, \dots, n} \{a_{ij} - p_j\}.$$

- There is equilibrium when all agents are *happy*

Auction Models

- Basic algorithm for task allocation: *naive auction*

```
function naive_auction(agents, objects)
  prices :- random_prices()
  assignment :- random_assignment(agents, objects, prices)
  while not everyone_happy(assignment) do
     $a_i$  :- select_non_happy_agent(assignment)
     $j_i$  :- select_maximal_value_object( $a_i$ , objects, prices)
     $a_j$  :- current_assignee(assignment,  $j_i$ )
    assignment :- exchange_assignments(assignment,  $a_i$ ,  $a_j$ )
    assignment :- raise_price(assignment,  $j_i$ , ?)
  end_while
  return assignment
end_function
```

Usually the increment is for an amount between:

- 0, and
- the difference between the value of the best object for a_i (j_i) and the value of the second best object for a_i
(otherwise it doesn't make sense to bid for j_i anymore)

Auction Model

- How do we model the utility function?
 - Expected reward
 - Cost function
- How do agents decide how much to bid?
- How do we make sure the system converges into a decision?

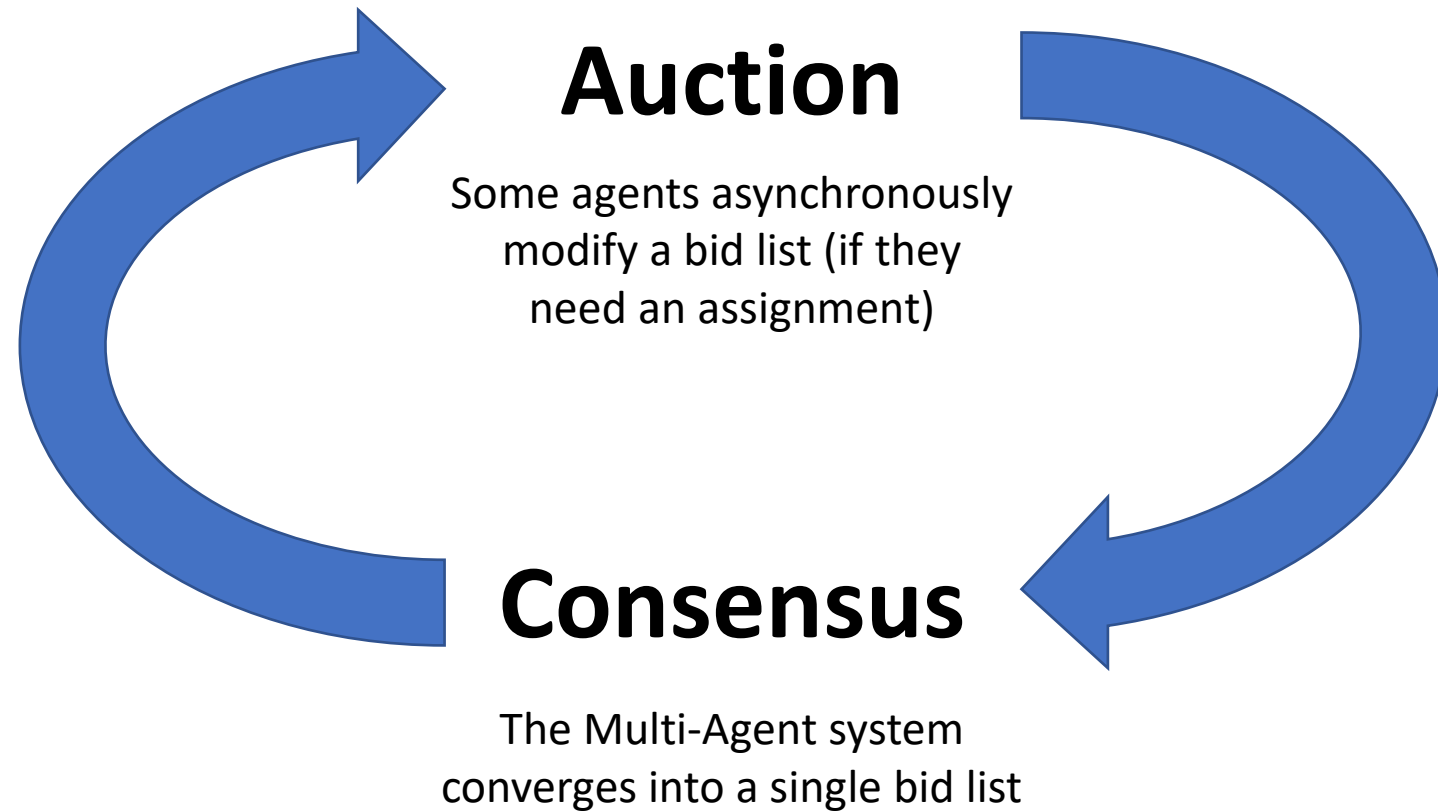
Auction Model

- How do we model the utility function?
 - Expected reward
 - Alignment to individual preferences, priorities, adherence to norms, ...
 - Cost function
 - Energy, time, distance, ...
- How do agents decide how much to bid?
 - Depends on individual strategy
 - This might impact convergence
- How do we make sure the system converges into a decision?
 - Implemented by design
 - Timeouts
 - Maximum number of bids / rounds

Auction Model: CBAA

- Source: Choi, H. L., Brunet, L., & How, J. P. (2009). **Consensus-based decentralized auctions for robust task allocation**. *IEEE transactions on robotics*, 25(4), 912-926. [[pdf](#)]
- CBAA: Consensus-based bundle algorithm
- Objective: to coordinate (in a decentralized fashion) a fleet of autonomous vehicles
- Combination of two decentralized algorithms:
 - Market auction
 - Consensus
- Guarantees (under certain modelling constraints):
 - Convergence
 - Conflict-free assignment
 - Global payoff $\geq (0.5 * \text{theoretical optimal})$

Auction Model: CBAA



Auction Model: CBAA

```
function select_task(ai, assignment, bid_list)
    local_value_function = get_local_value_function(ai)
    if not have_assignment(assignment, ai)
        good_tasks = filter_tasks_by_positive_difference(bid_list, local_value_function)
        if not empty(good_tasks)
            best_task = best_task_by_value(good_tasks, local_value_function)
            assignment = assign_task(assignment, best_task, ai)
            current_bid = get_current_bid(bid_list, best_task)
            new_bid = choose_value_between(current_bid, local_value_function(best_task))
            bid_list = add_bid(bid_list, best_task, ai, new_bid)
        end_if
    end_if
    return assignment, bid_list
end_function
```

reward - cost

value > current_bid

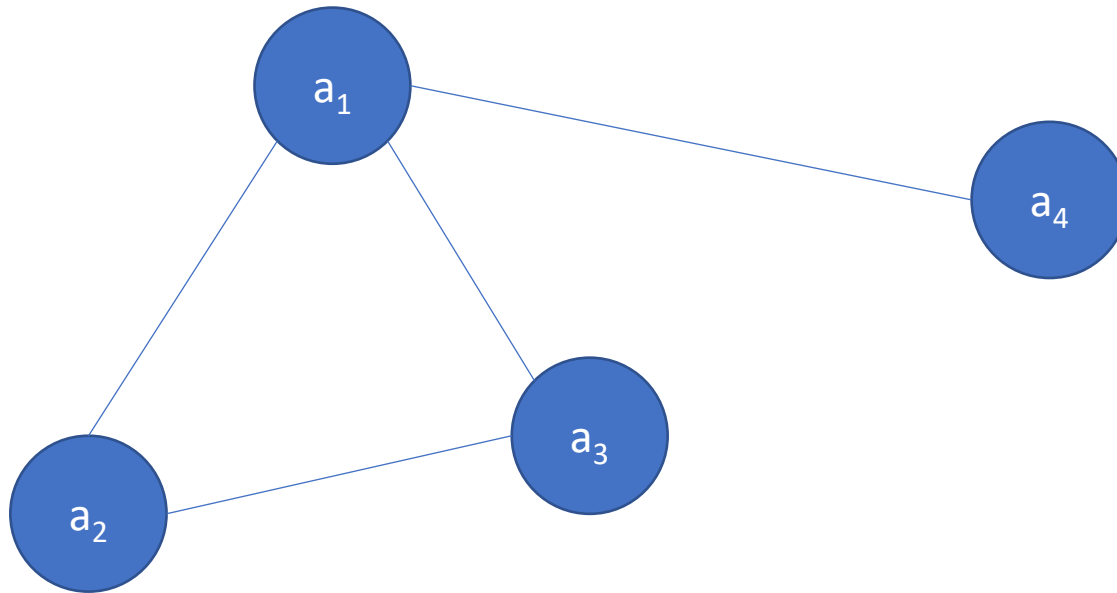
Auction Model: CBAA

```
function select_task(ai, assignment, bid_list)
  local_value_function = get_local_value_function(ai)
  if not have_assignment(assignment, ai)
    good_tasks = filter_tasks_by_positive_difference(bid_list, local_value_function)
    if not empty(good_tasks)
      best_task = best_task_by_value(good_tasks, local_value_function)
      assignment = assign_task(assignment, best_task, ai)
      current_bid = get_current_bid(bid_list, best_task)
      new_bid = choose_value_between(current_bid, local_value_function(best_task))
      bid_list = add_bid(bid_list, best_task, ai, new_bid)
    end_if
  end_if
  return assignment, bid_list
end_function
```

reward - cost

value > current_bid

Auction Model (example)



r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

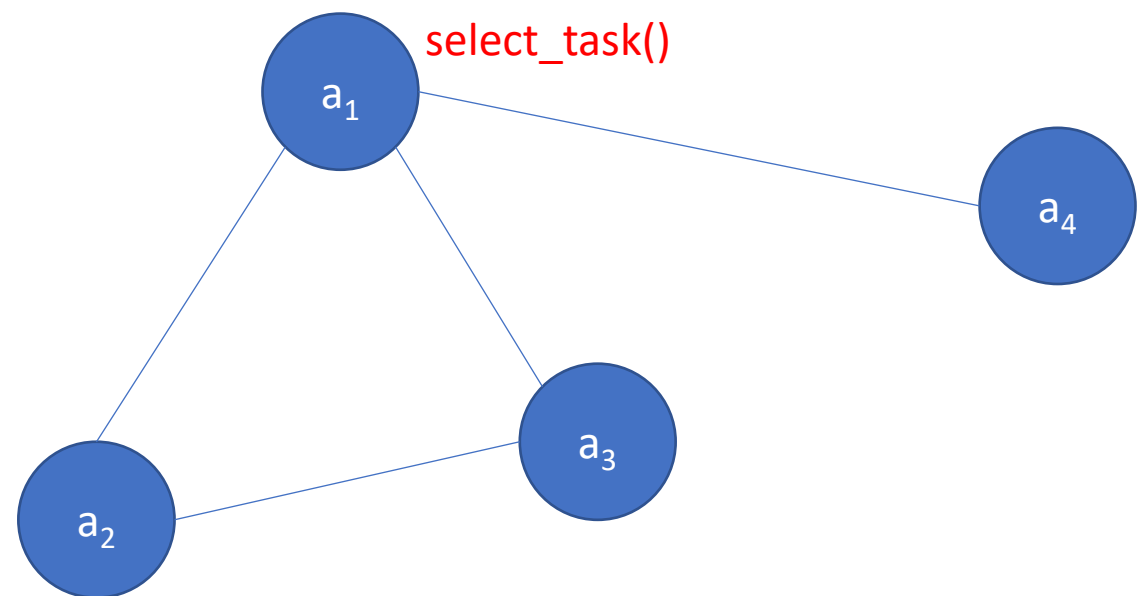
c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

Assumptions:

$s_1 - s_2 - s_3 - s_4$ form a physical line and the cost of moving from s_i to s_{i+1} is 1. Initially, for all i , a_i is at s_i . Agents distribute their preferences so that the sum of their rewards is 10. Agents always bid the maximum. The consensus algorithm circumvents the topology and we run it after every bid list modification.

	s ₁	s ₂	s ₃	s ₄
assignment				
bids	0	0	0	0

Auction Model (example)

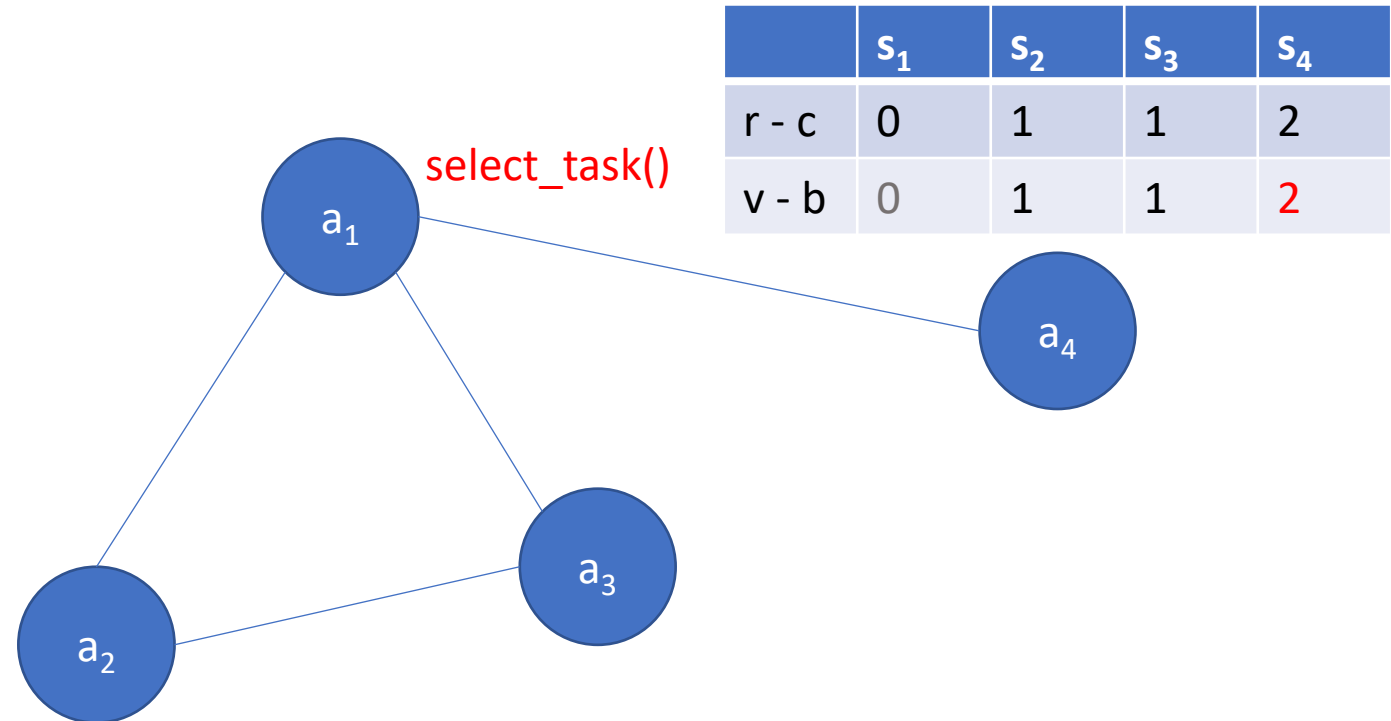


r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

	s ₁	s ₂	s ₃	s ₄
assignment				
bids	0	0	0	0

Auction Model (example)



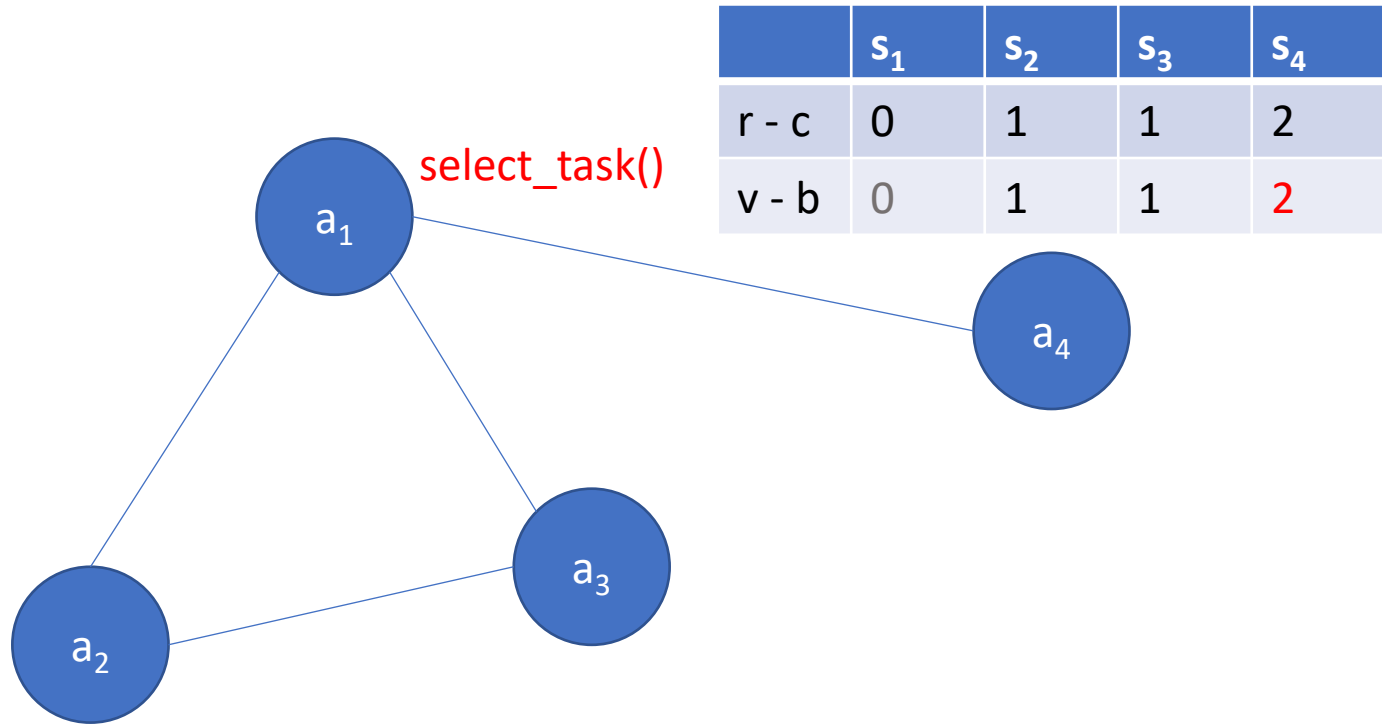
	s ₁	s ₂	s ₃	s ₄
r - c	0	1	1	2
v - b	0	1	1	2

r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

	s ₁	s ₂	s ₃	s ₄
assignment				
bids	0	0	0	0

Auction Model (example)



	s ₁	s ₂	s ₃	s ₄
r - c	0	1	1	2
v - b	0	1	1	2

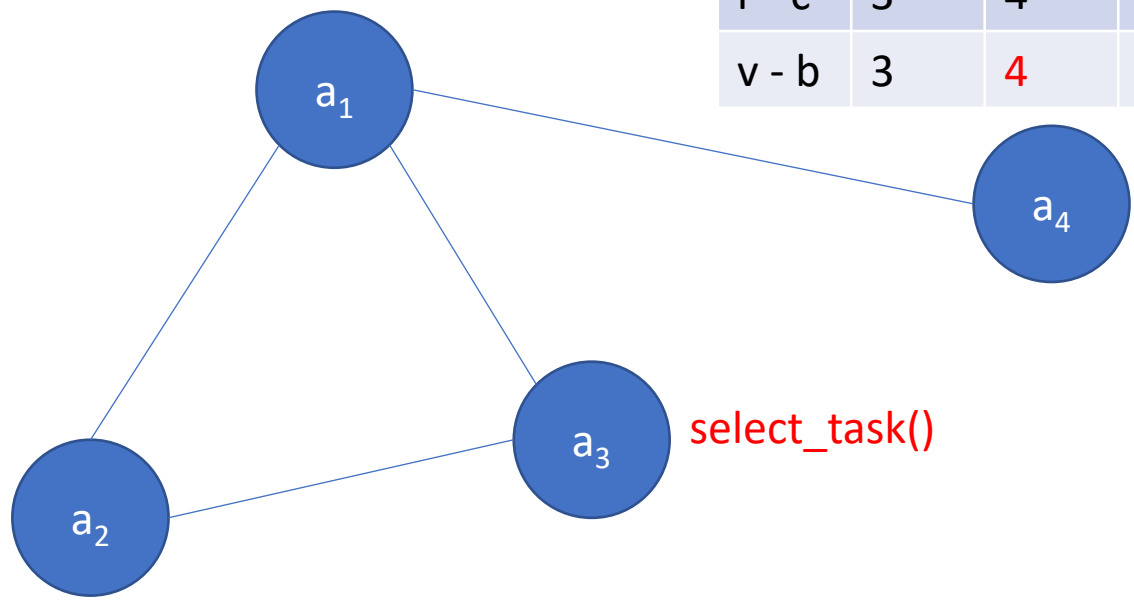
r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

	s ₁	s ₂	s ₃	s ₄
assignment				a ₁
bids	0	0	0	2

Could have been any value in (0, 2]

Auction Model (example)



	s ₁	s ₂	s ₃	s ₄
r - c	3	4	0	-1
v - b	3	4	0	-3

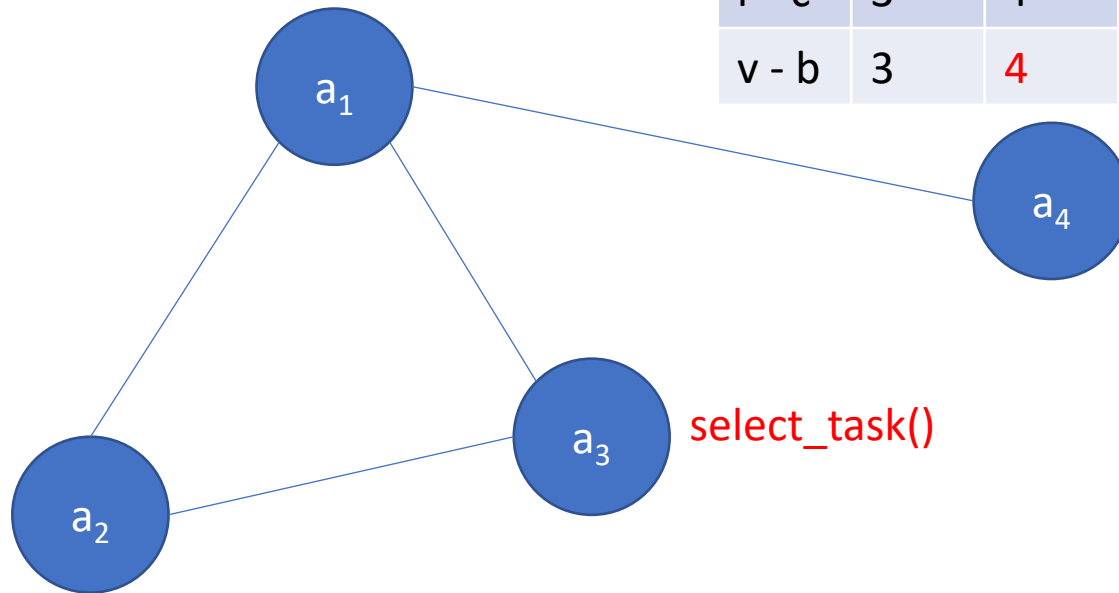
r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

select_task()

	s ₁	s ₂	s ₃	s ₄
assignment				a ₁
bids	0	0	0	2

Auction Model (example)



	s_1	s_2	s_3	s_4
$r - c$	3	4	0	-1
$v - b$	3	4	0	-3

`select_task()`

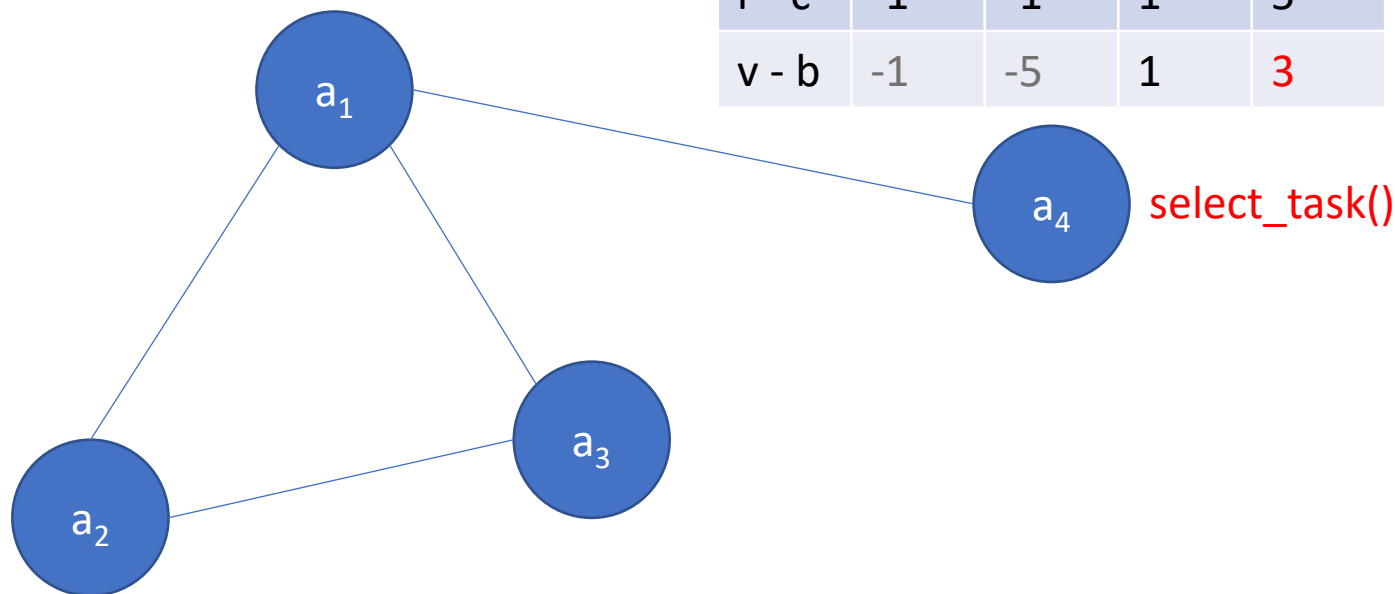
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment		a_3		a_1
bids	0	4	0	2

Could have been any value in $(0, 4]$

Auction Model (example)



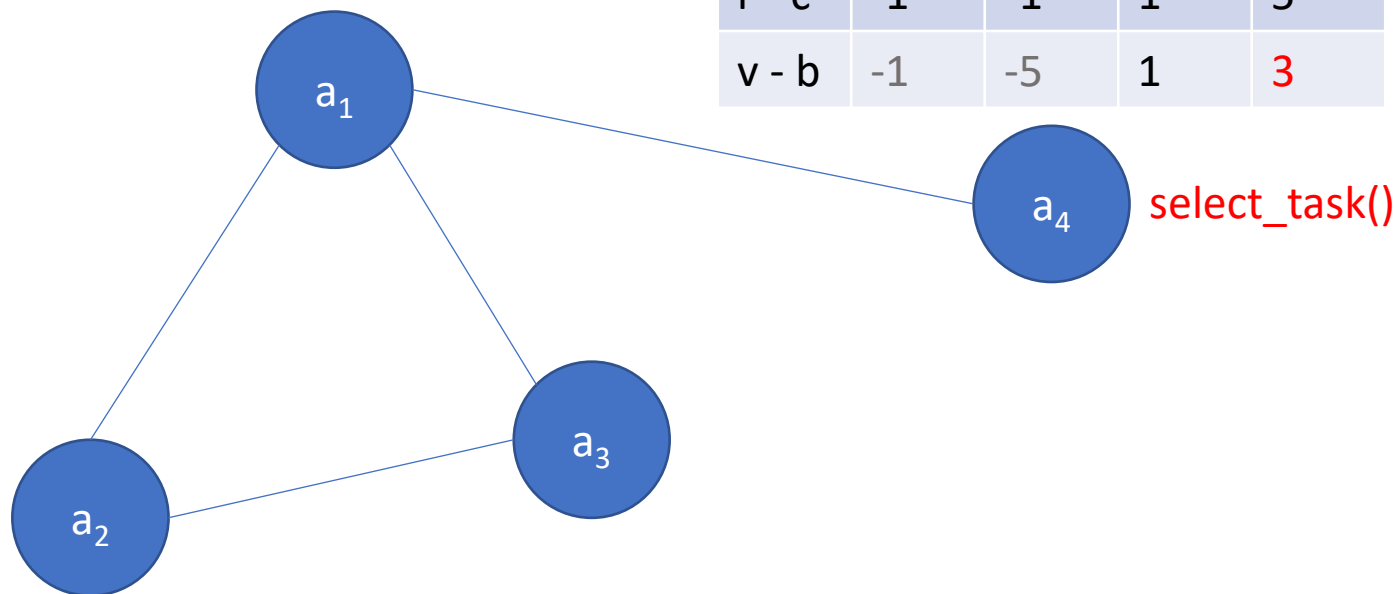
	s_1	s_2	s_3	s_4
$r - c$	-1	-1	1	5
$v - b$	-1	-5	1	3

r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment		a_3		a_1
bids	0	4	0	2

Auction Model (example)



	s ₁	s ₂	s ₃	s ₄
r - c	-1	-1	1	5
v - b	-1	-5	1	3

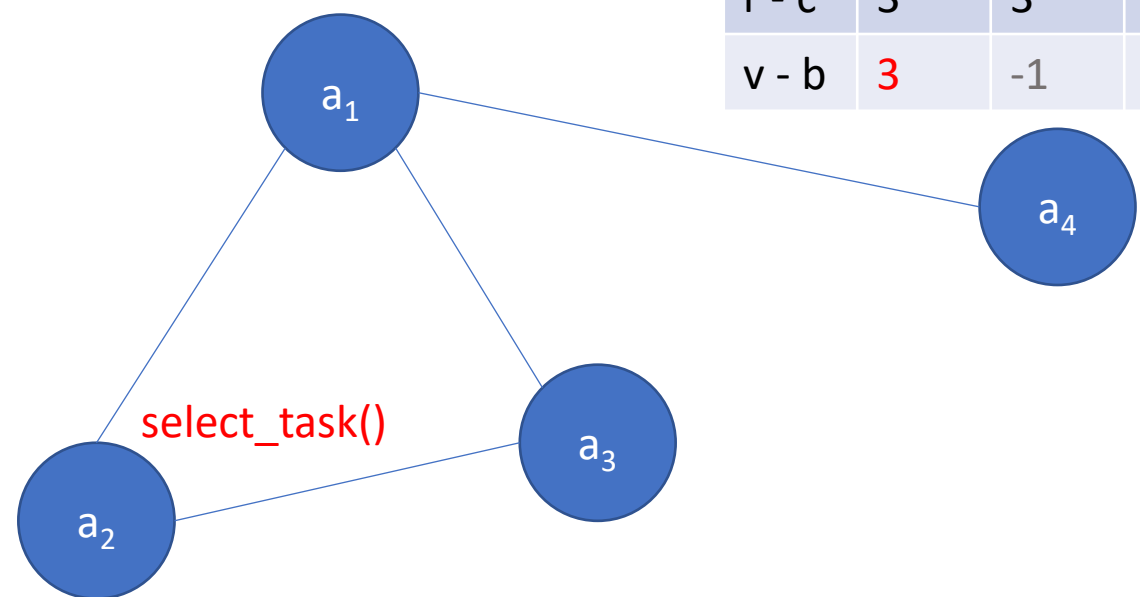
r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

	s ₁	s ₂	s ₃	s ₄
assignment		a ₃		a ₄
bids	0	4	0	5

Could have been any value in (2, 5]

Auction Model (example)



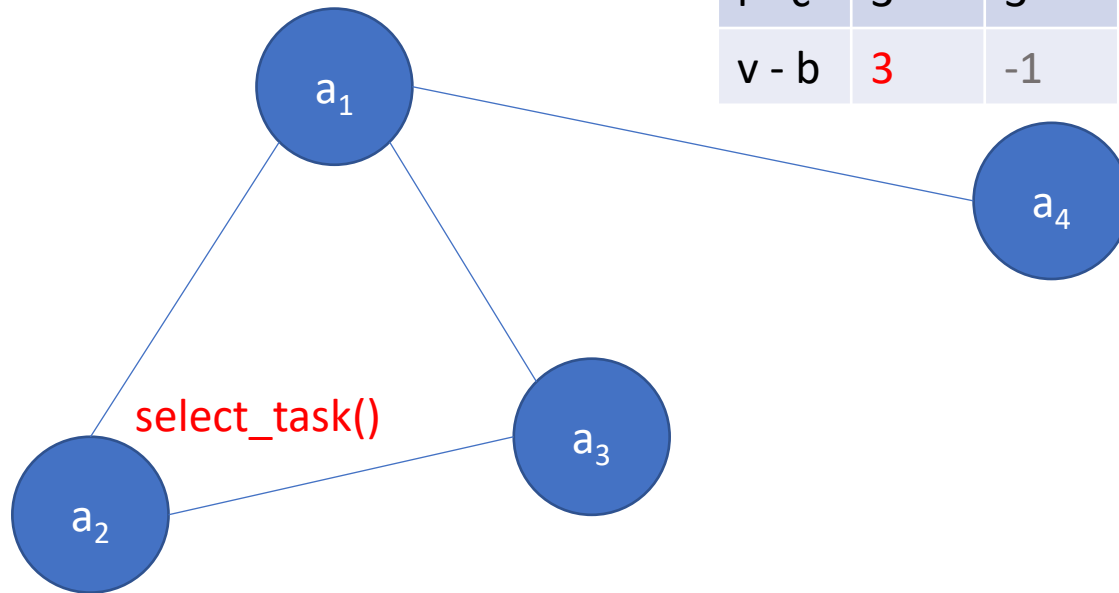
	s_1	s_2	s_3	s_4
$r - c$	3	3	1	-1
$v - b$	3	-1	1	-6

r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment		a_3		a_4
bids	0	4	0	5

Auction Model (example)



	s_1	s_2	s_3	s_4
$r - c$	3	3	1	-1
$v - b$	3	-1	1	-6

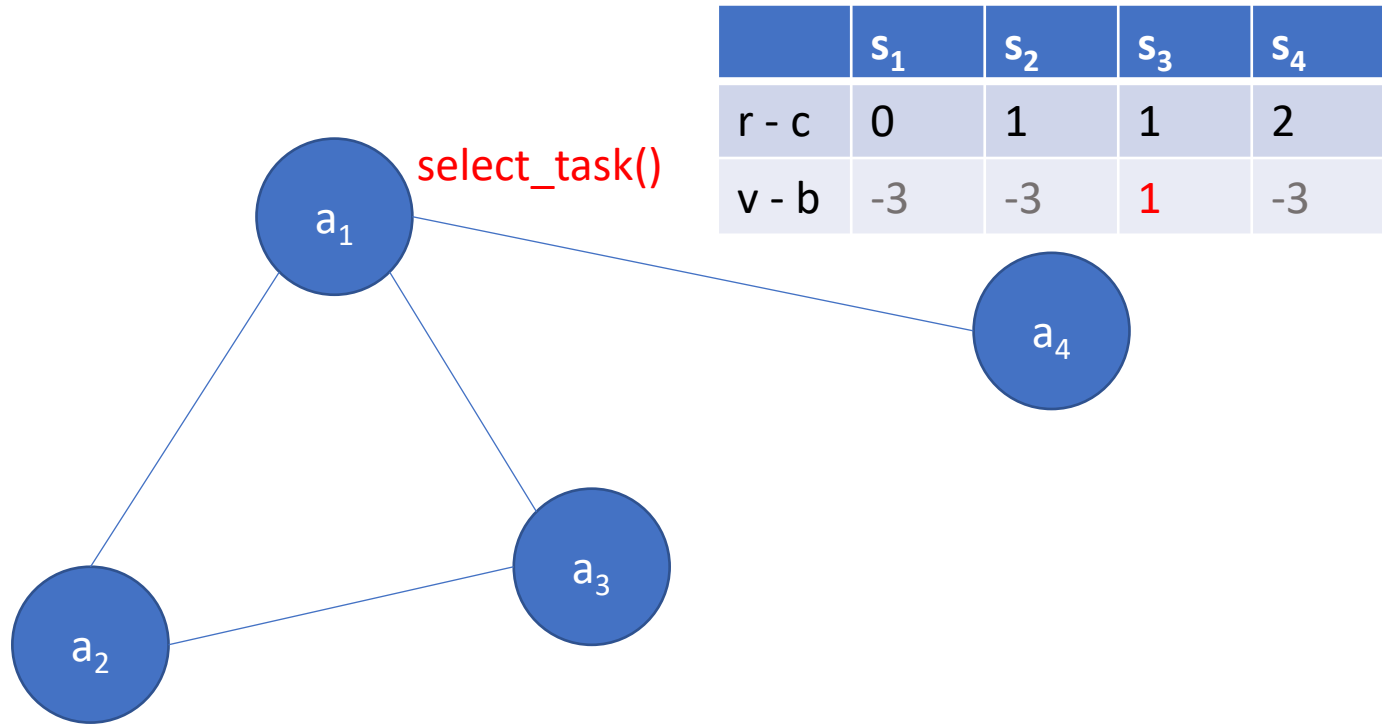
r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	0	1	2	3
a_2	1	0	1	2
a_3	2	1	0	1
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment	a_2	a_3		a_4
bids	3	4	0	5

Could have been any value in $(0, 3]$

Auction Model (example)



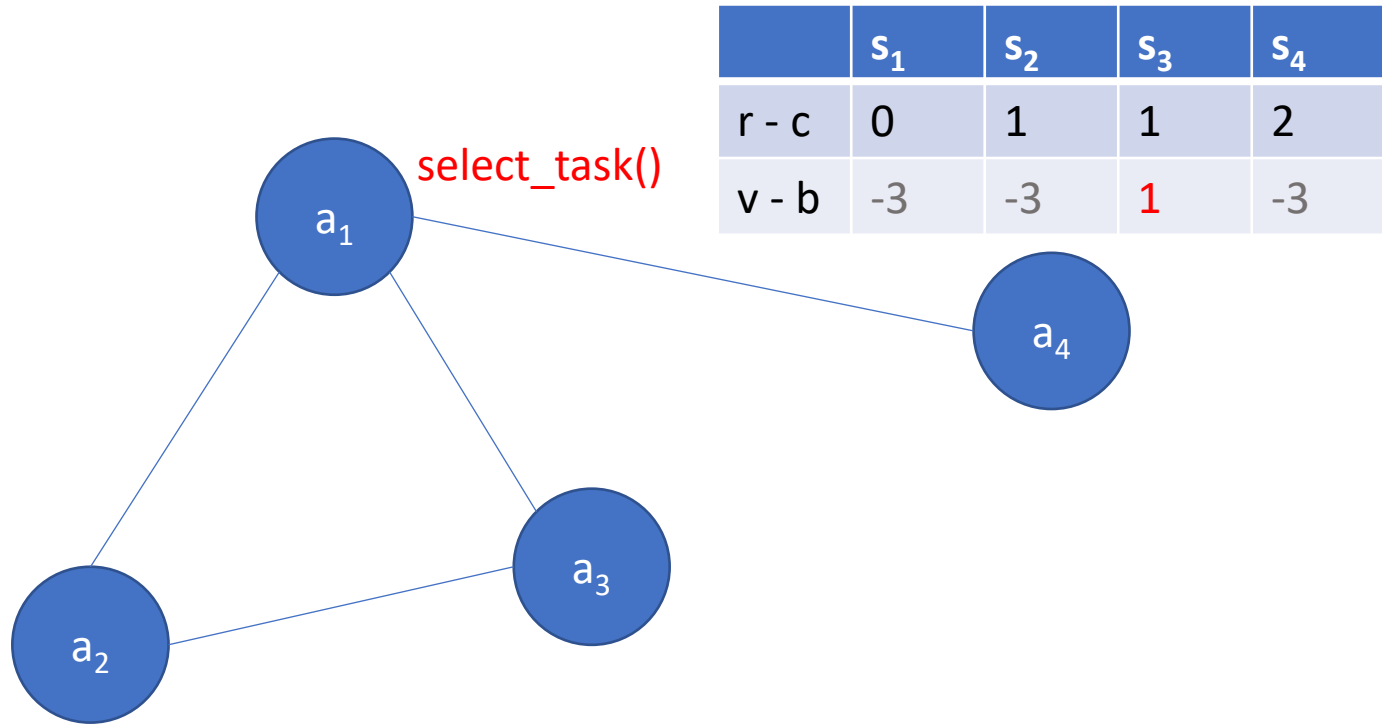
	s ₁	s ₂	s ₃	s ₄
r - c	0	1	1	2
v - b	-3	-3	1	-3

r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

	s ₁	s ₂	s ₃	s ₄
assignment	a ₂	a ₃		a ₄
bids	3	4	0	5

Auction Model (example)



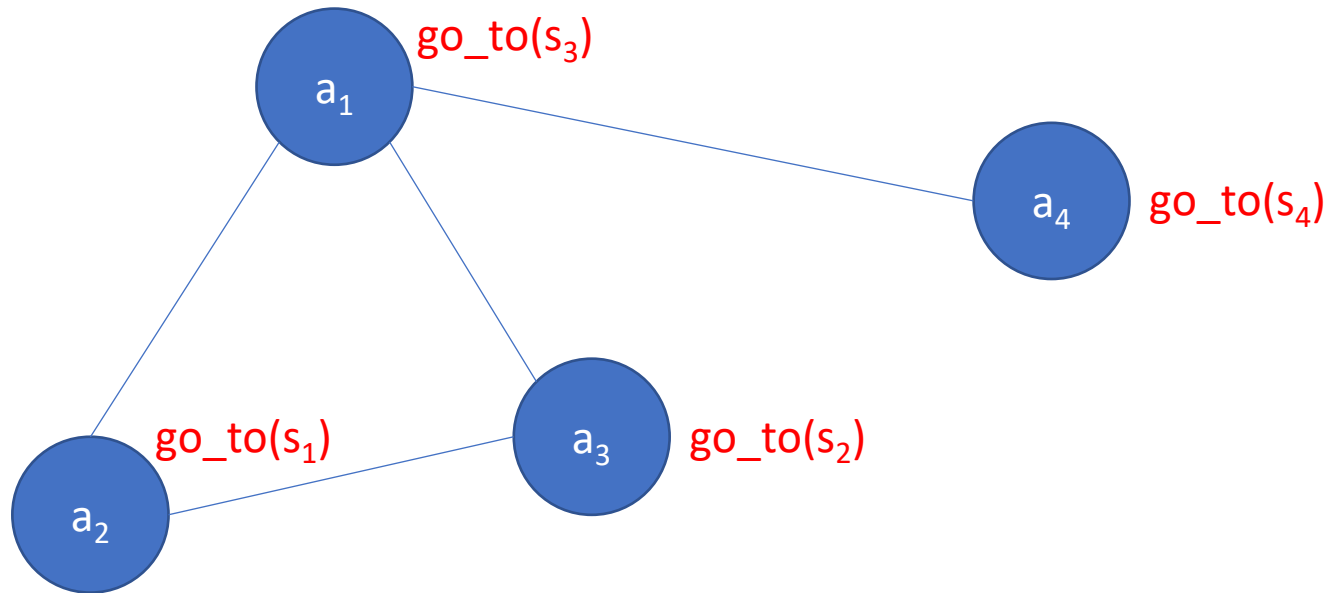
	s ₁	s ₂	s ₃	s ₄
r - c	0	1	1	2
v - b	-3	-3	1	-3

r	s ₁	s ₂	s ₃	s ₄
a ₁	0	2	3	5
a ₂	4	3	2	1
a ₃	5	5	0	0
a ₄	2	1	2	5

c	s ₁	s ₂	s ₃	s ₄
a ₁	0	1	2	3
a ₂	1	0	1	2
a ₃	2	1	0	1
a ₄	3	2	1	0

	s ₁	s ₂	s ₃	s ₄
assignment	a ₂	a ₃	a ₁	a ₄
bids	3	4	1	5

Auction Model (example)



r	s_1	s_2	s_3	s_4
a_1	0	2	3	5
a_2	4	3	2	1
a_3	5	5	0	0
a_4	2	1	2	5

c	s_1	s_2	s_3	s_4
a_1	2	1	0	1
a_2	0	1	2	3
a_3	1	0	1	2
a_4	3	2	1	0

	s_1	s_2	s_3	s_4
assignment	a_2	a_3	a_1	a_4
bids	3	4	1	5

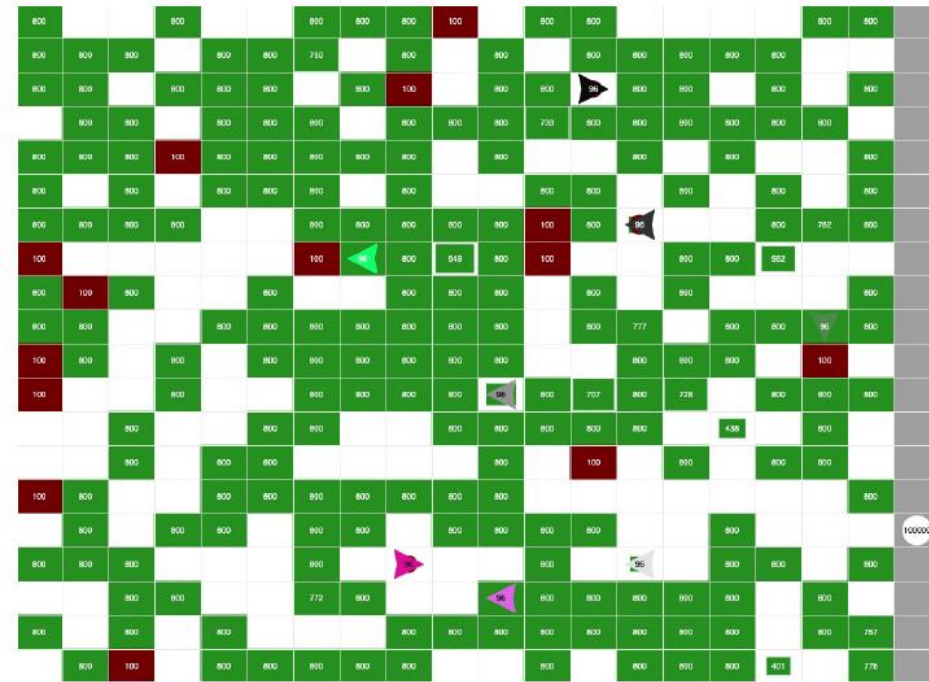
Communication Models

Blackboard

Actor Model

Toy Example

- We have a system of N drones with wireless communication capabilities up to a certain range of x kilometers
 - Individual units can be specialized (e.g. better camera, higher water tank capacity, faster speed, ...)
 - Each action (including communication) requires an amount of energy
 - Each drone has a variable amount of available energy



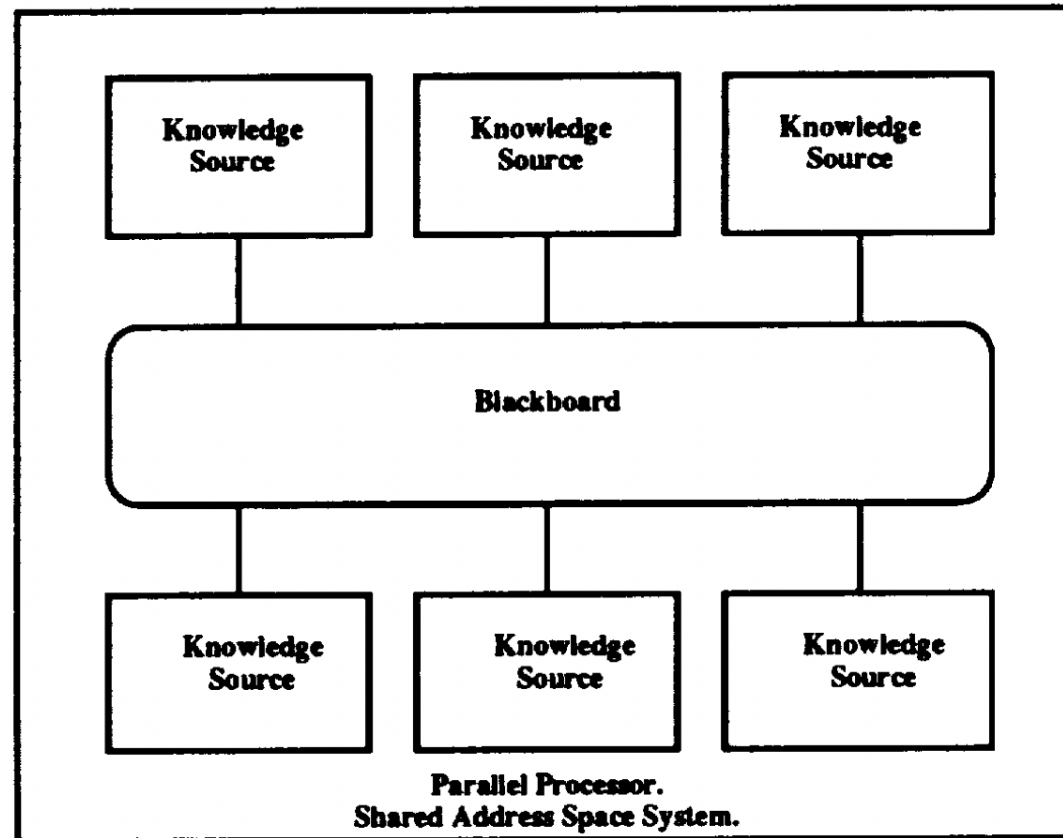
The Cooperative Negotiation and Coordination Approach in a Multi-Agent System for a Dynamic Real-Time Environment, Pedro Llanos Arroyo, TFG (UPC), 2021.

- There is additionally a regular computer at a fixed location that may be used or not
- They have to coordinate to fight fire spreading on a forest (simplified by a discrete map)

The Blackboard Model

- H. Penny Nii (1986):
 - *“a collection of intelligent agents (knowledge sources) who are gathered around a blackboard,*
 - *looking at pieces of information written on it,*
 - *thinking about the current state of the solution,*
 - *and writing their conclusions on the blackboard as they generate them”*
- Allen Newell (1962):
 - *“This conception is just that of Selfridge's Pandemonium: a set of demons, each independently looking at the total situation and shrieking in proportion to what they see that fits their natures...”*

The Blackboard Model



McManus, J. W., & Bynum, W. L. (1996). Design and analysis techniques for concurrent blackboard systems. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 26(6), 669-680.

The Blackboard Model: Properties

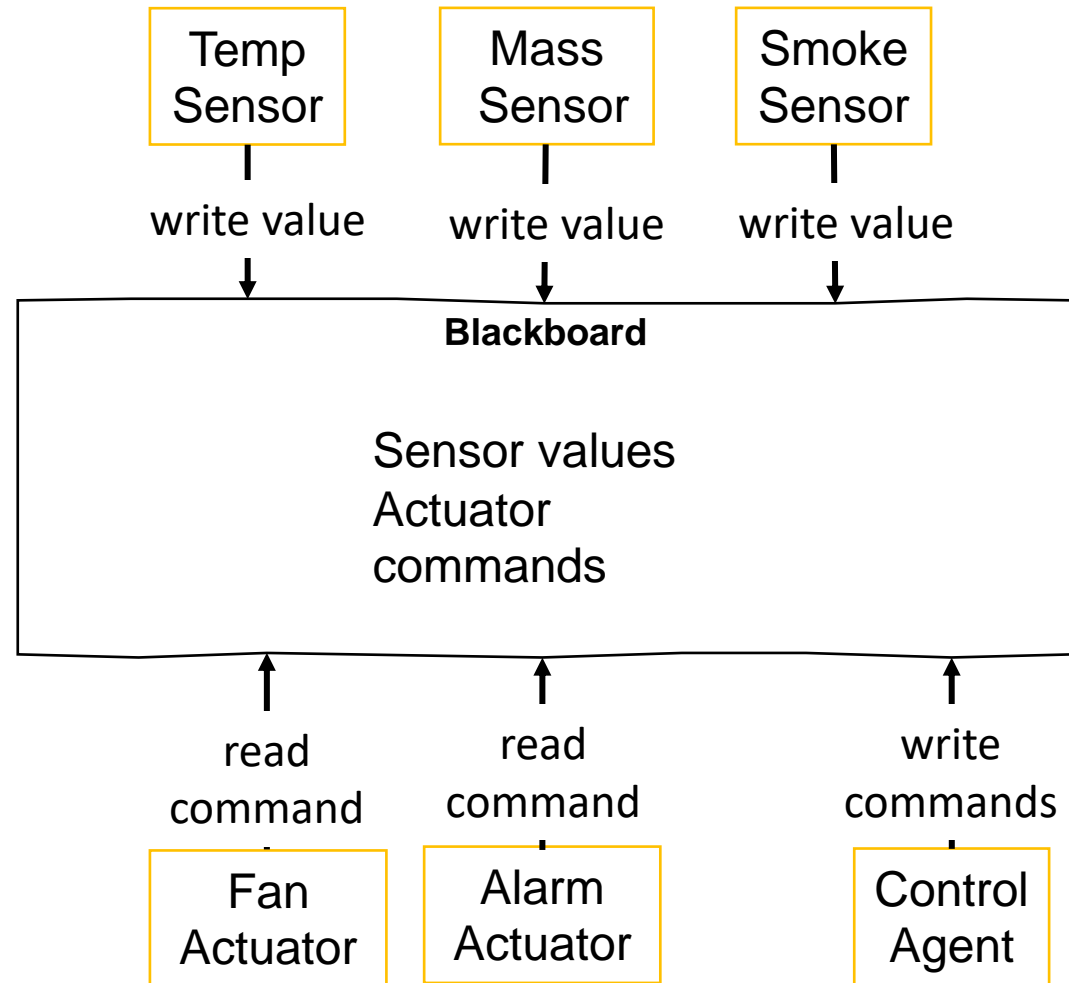
- **Centralized state**, single source of truth
- **Indirect communication**
- Agents **do not need to know each other**
- All agents can write and have access to the full state at any time:
concurrent asynchronous reads/writes
- Agents are **independent and specialized**
 - Mixture of implementations/algorithms/topics of concern
- **Modular protection**
- **Conflict resolution** is managed by the blackboard
- **Parallel execution** is possible at the agent level

Nii, H. P. (1986). The blackboard model of problem solving and the evolution of blackboard architectures. AI magazine, 7(2).

The Blackboard Model: Properties

- Disadvantages:
 - High dependency over data structure
 - Low tolerance to failures (bottleneck)
 - Potentially high overhead in communication

The Blackboard Model: Example



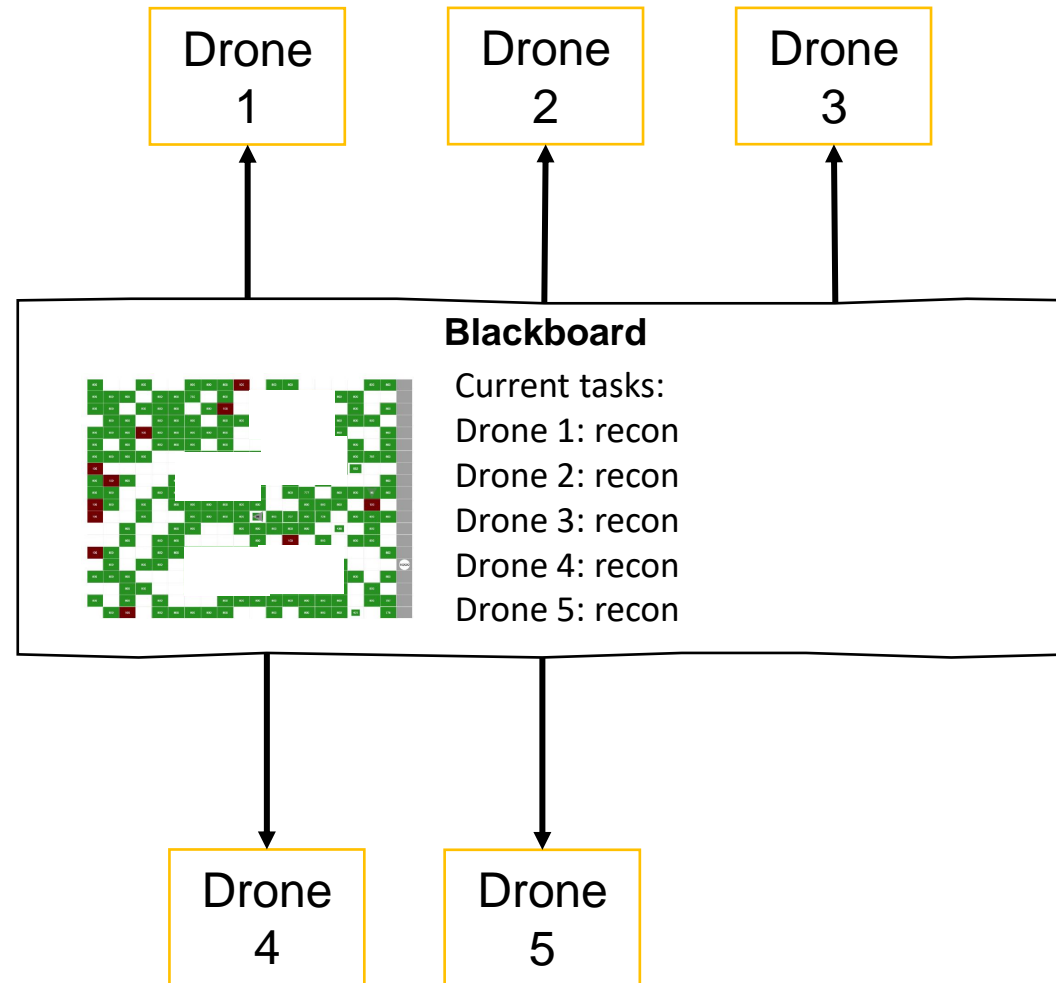
The Blackboard Model: Example

- How do we model a problem in this communication model?
 - What are the Knowledge Sources?
 - What knowledge do they provide?
 - What is the content of the Blackboard?
 - Where is the Blackboard located?
 - What conflicts will the Blackboard have to deal with?

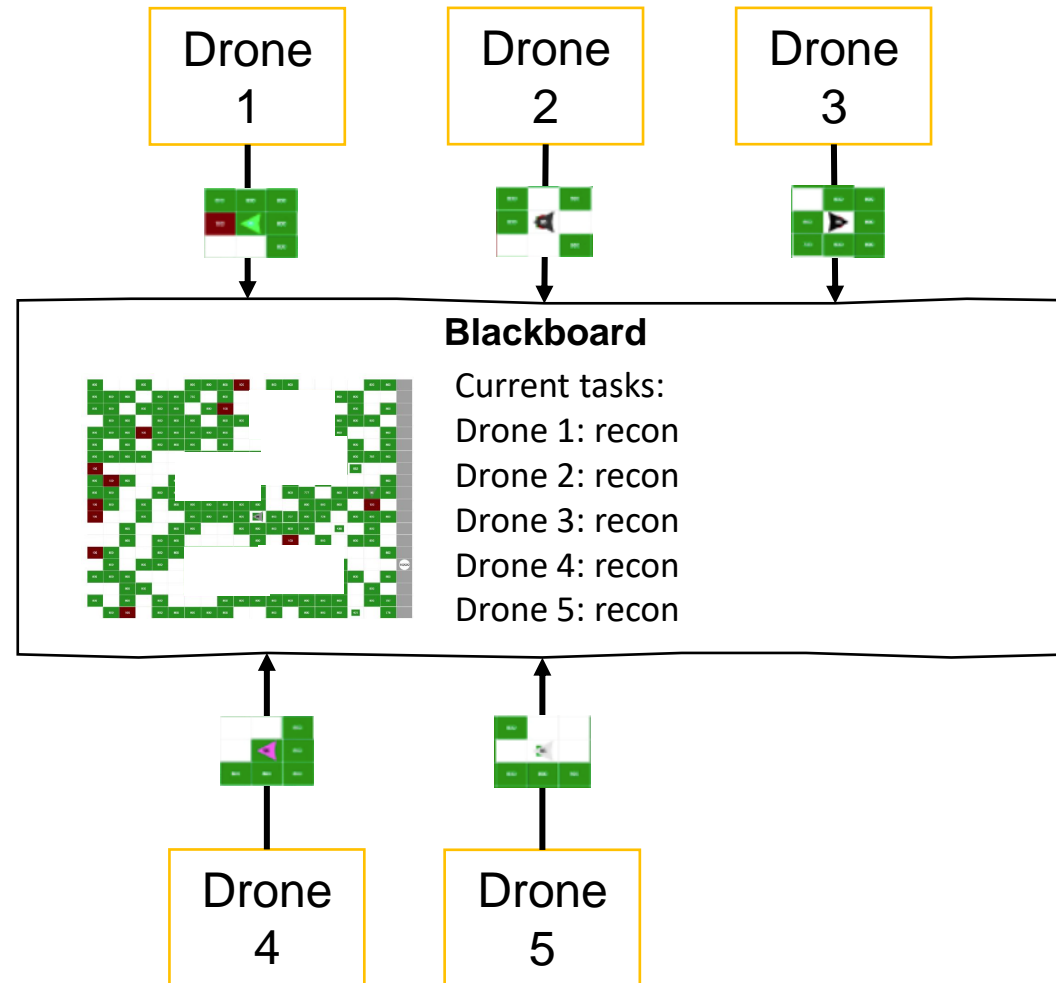
The Blackboard Model: Example

- How do we model a problem in this communication model?
 - What are the Knowledge Sources?
 - Drones
 - What knowledge do they provide?
 - Partial map, GPS location, current action, plans of action...
 - What is the content of the Blackboard?
 - Global map, current assignments
 - Where is the Blackboard located?
 - Could be the computer (depending on comms range)
 - Could be a drone or replicated among the drones
 - What conflicts will the Blackboard have to deal with?
 - Conflicting partial maps/GPS info, duplicate targets/actions

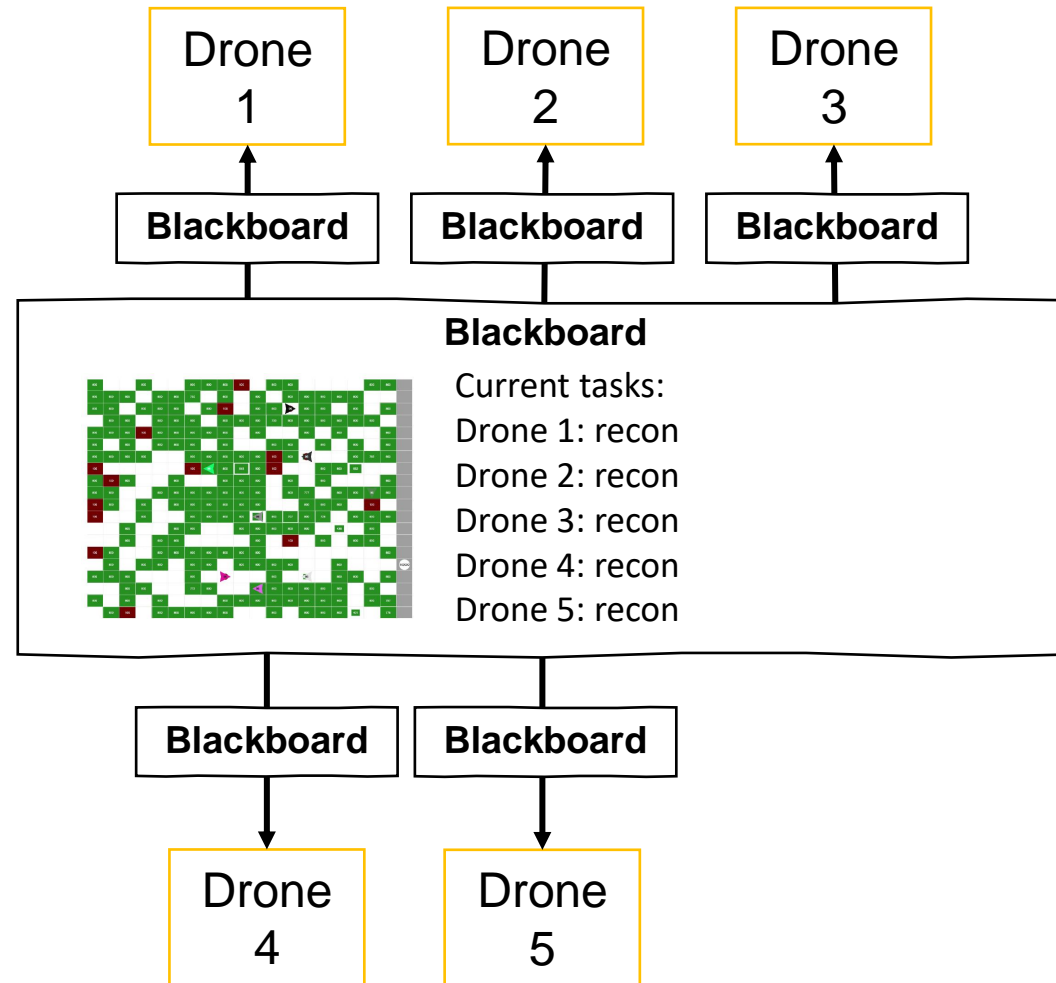
The Blackboard Model: Example (I)



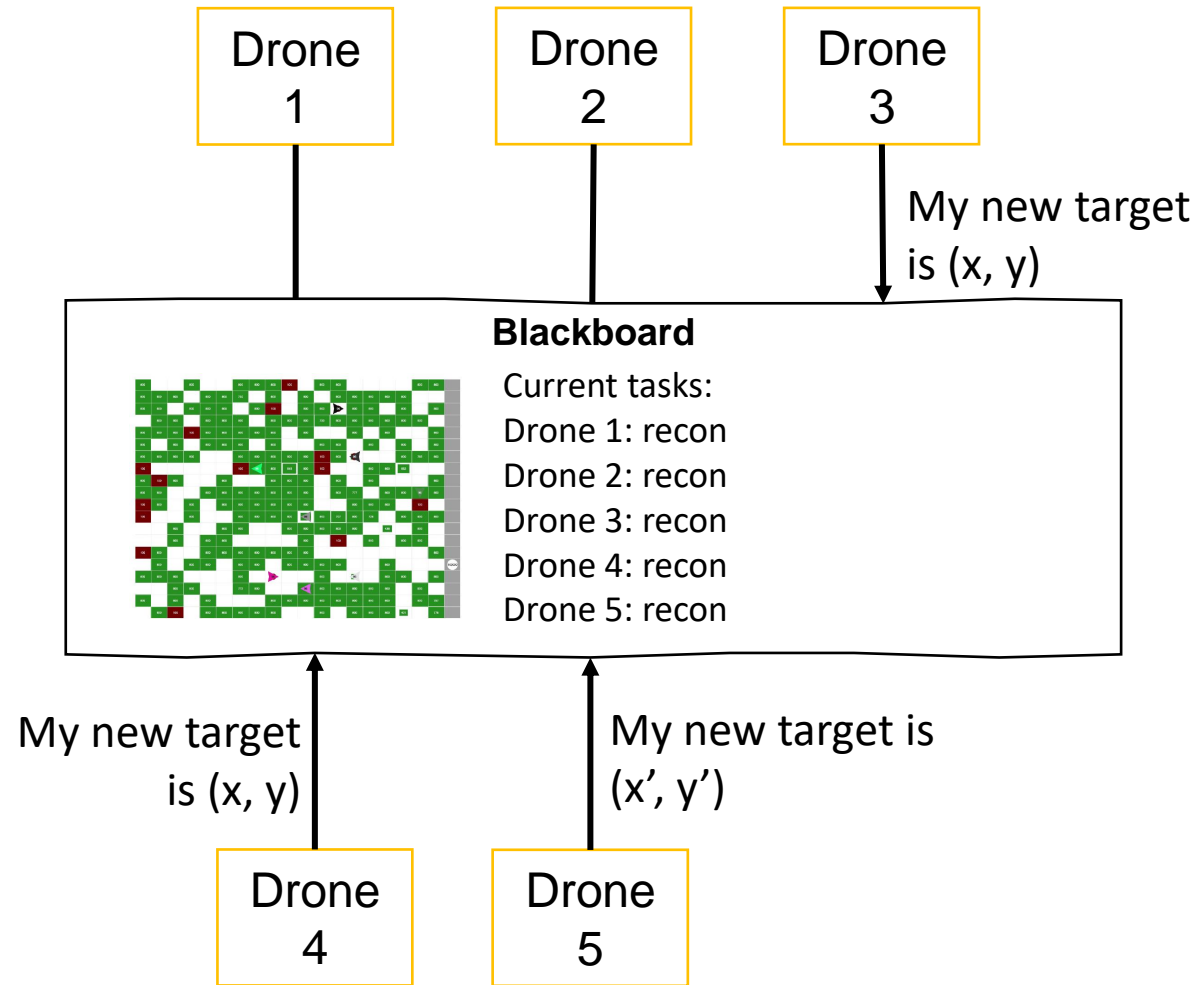
The Blackboard Model: Example (I)



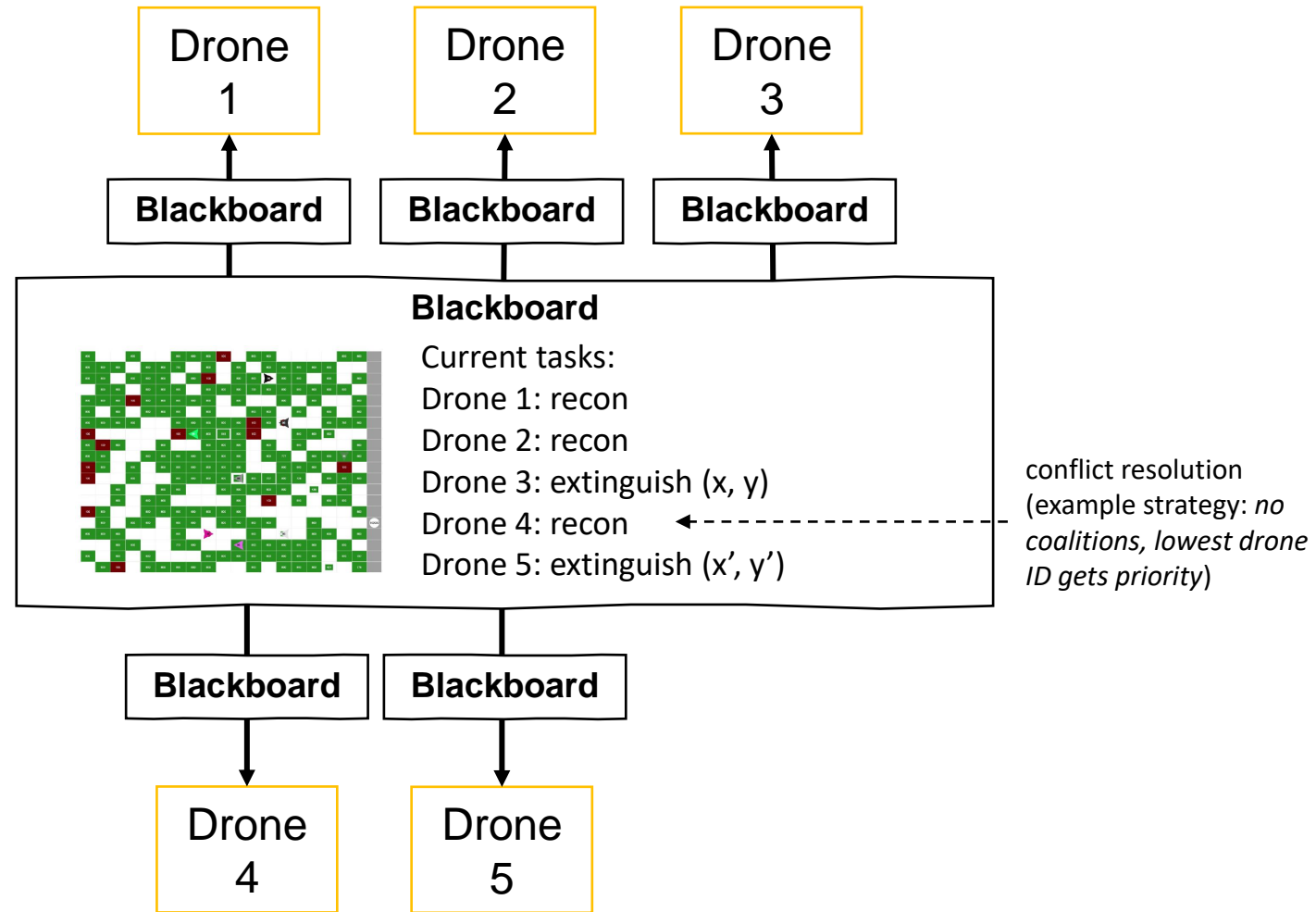
The Blackboard Model: Example (I)



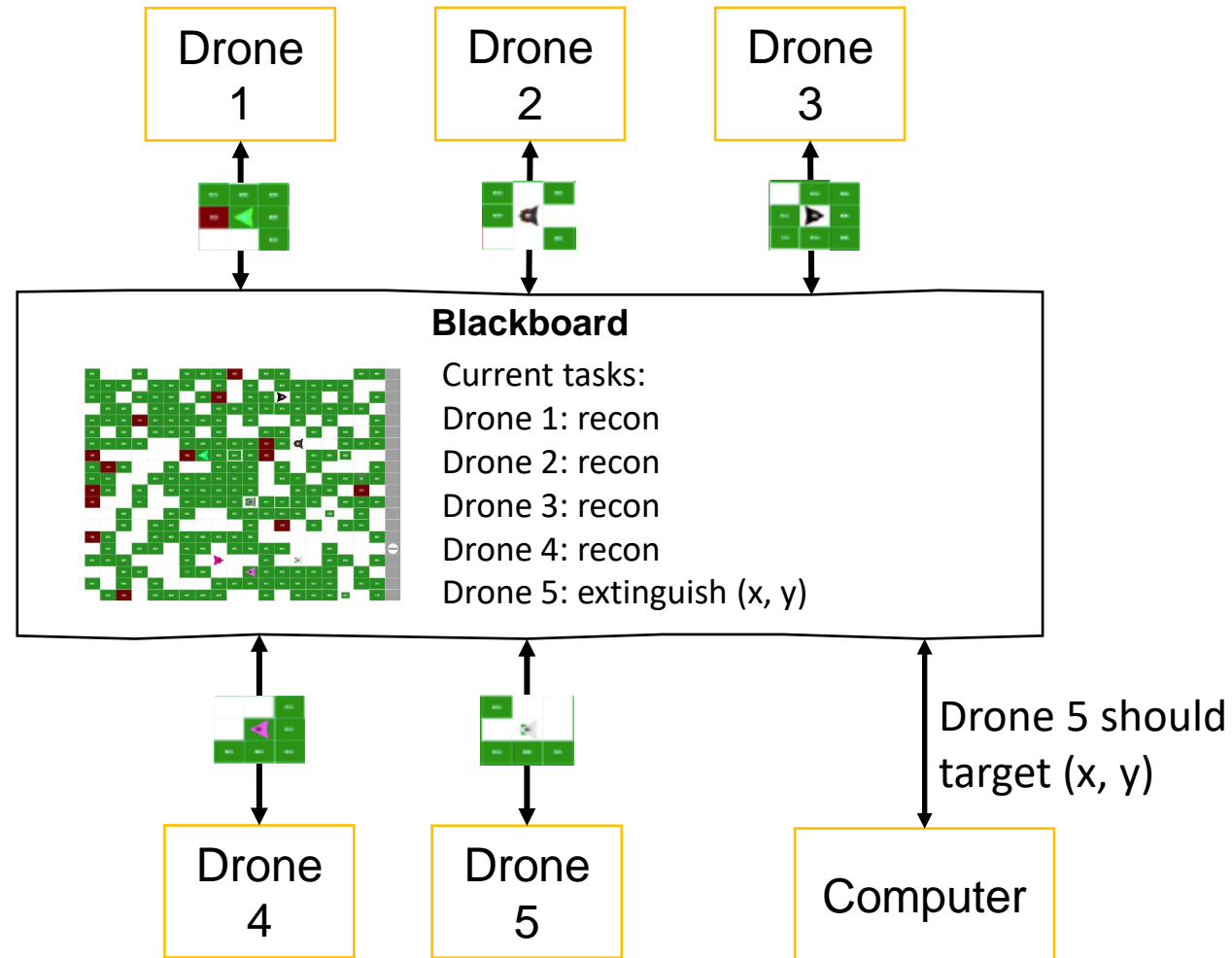
The Blackboard Model: Example (I)



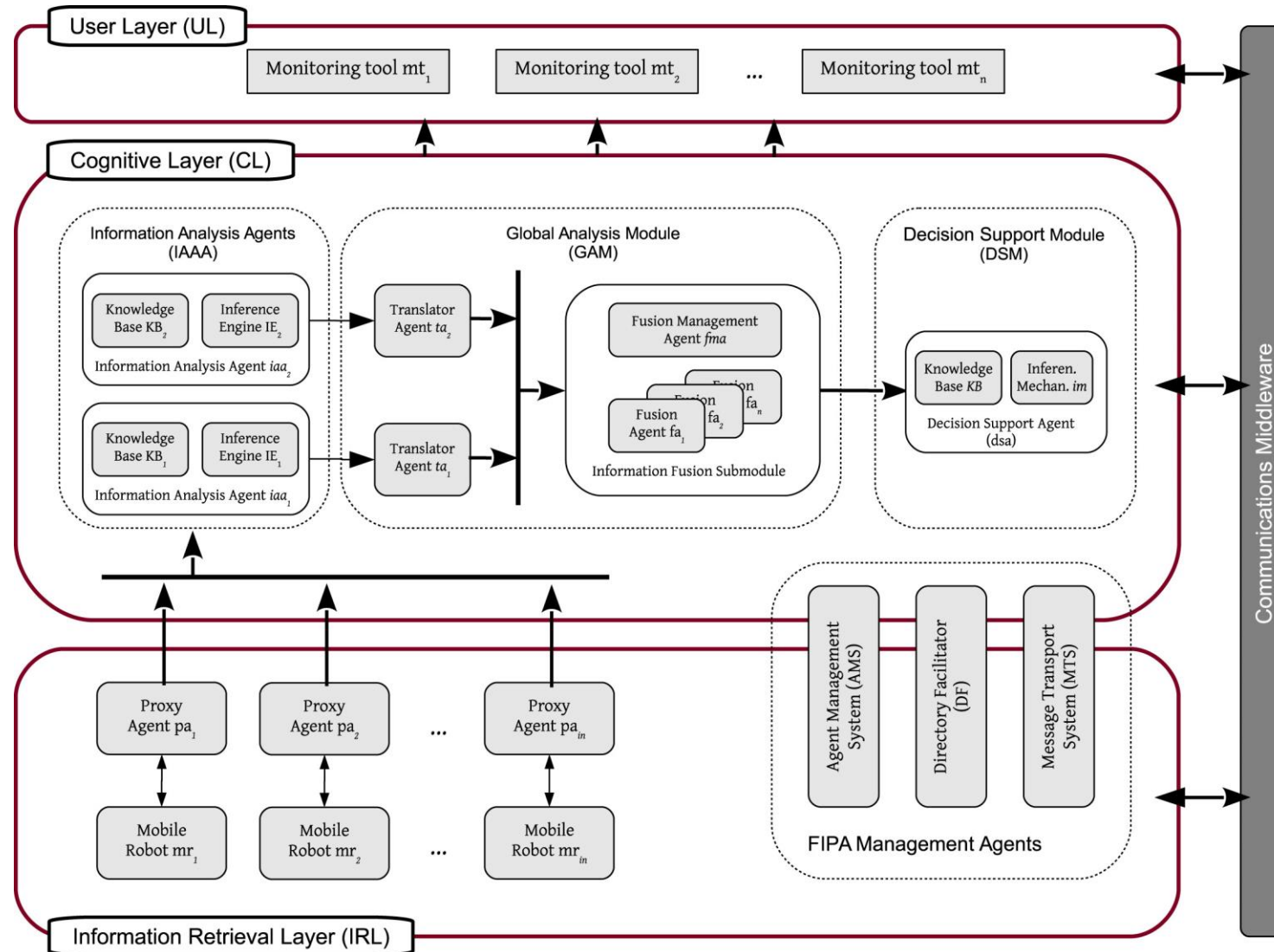
The Blackboard Model: Example (I)



The Blackboard Model: Example (II)

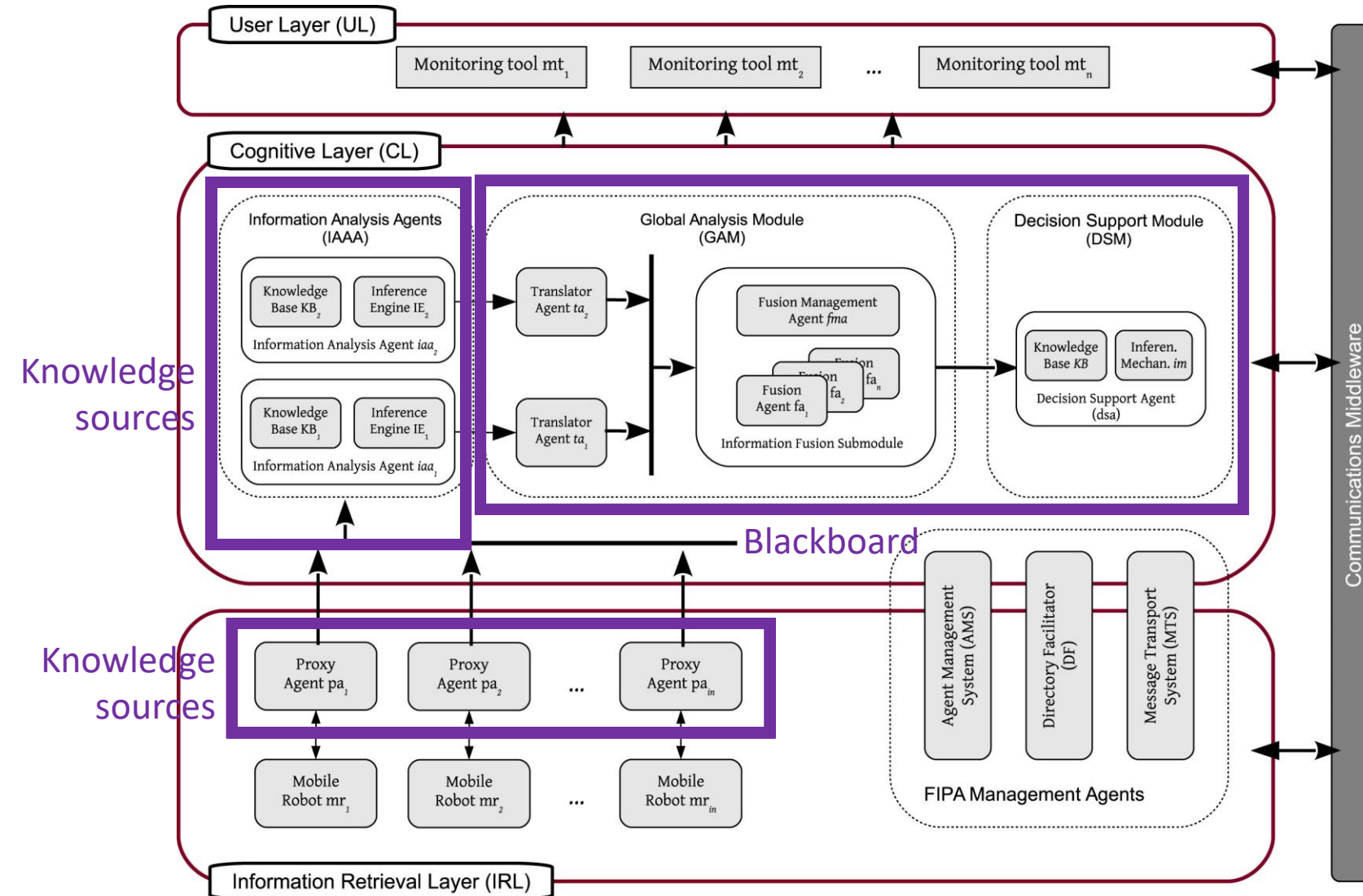


The Blackboard Model: Example (III)



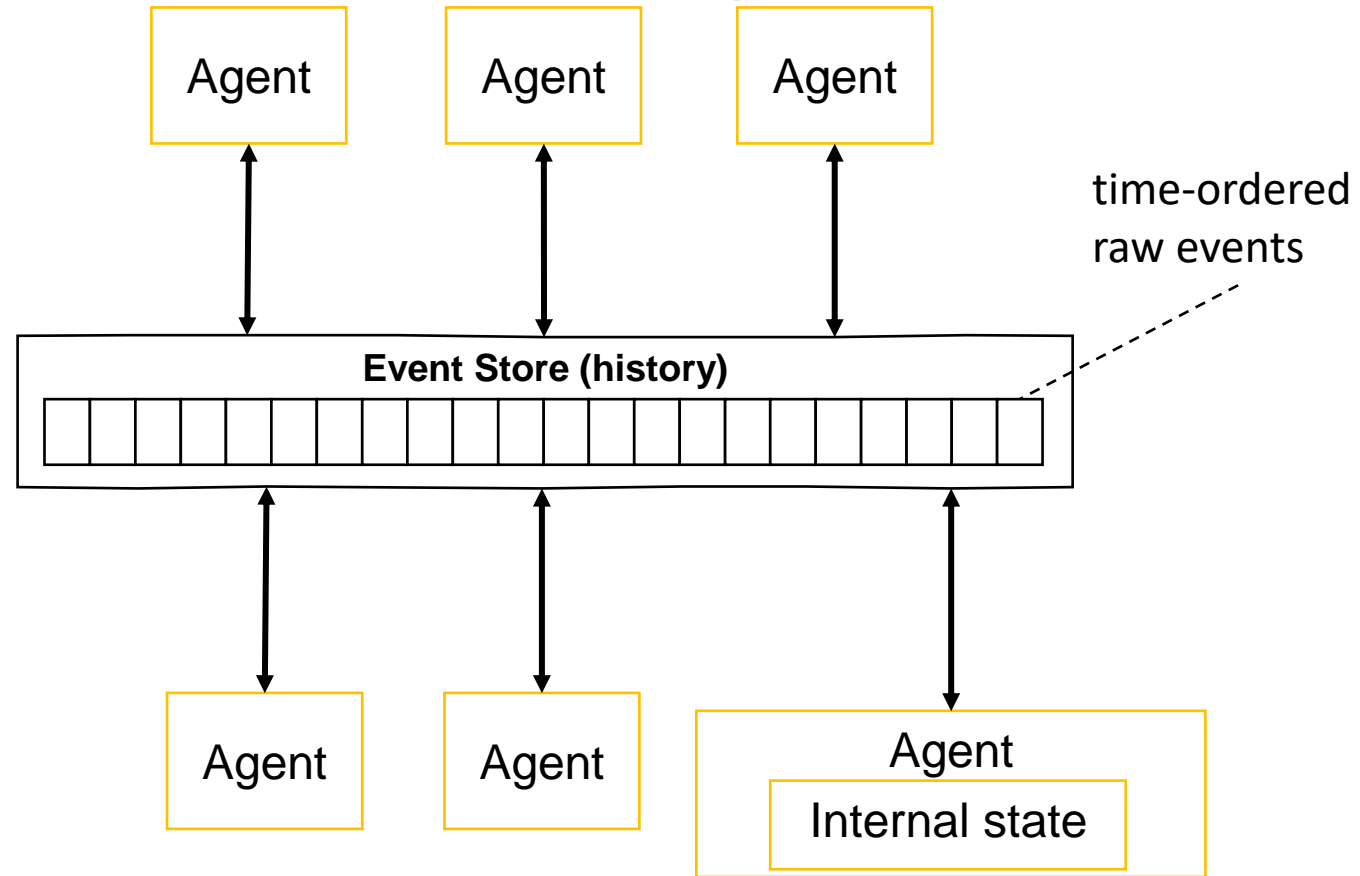
Vallejo, D., Castro-Schez, J. J., Glez-Morcillo, C., & Albusac, J. (2020). Multi-agent architecture for information retrieval and intelligent monitoring by UAVs in known environments affected by catastrophes. *Engineering Applications of Artificial Intelligence*, 87, 103243.

The Blackboard Model: Example (III)



Vallejo, D., Castro-Schez, J. J., Glez-Morcillo, C., & Albusac, J. (2020). Multi-agent architecture for information retrieval and intelligent monitoring by UAVs in known environments affected by catastrophes. *Engineering Applications of Artificial Intelligence*, 87, 103243.

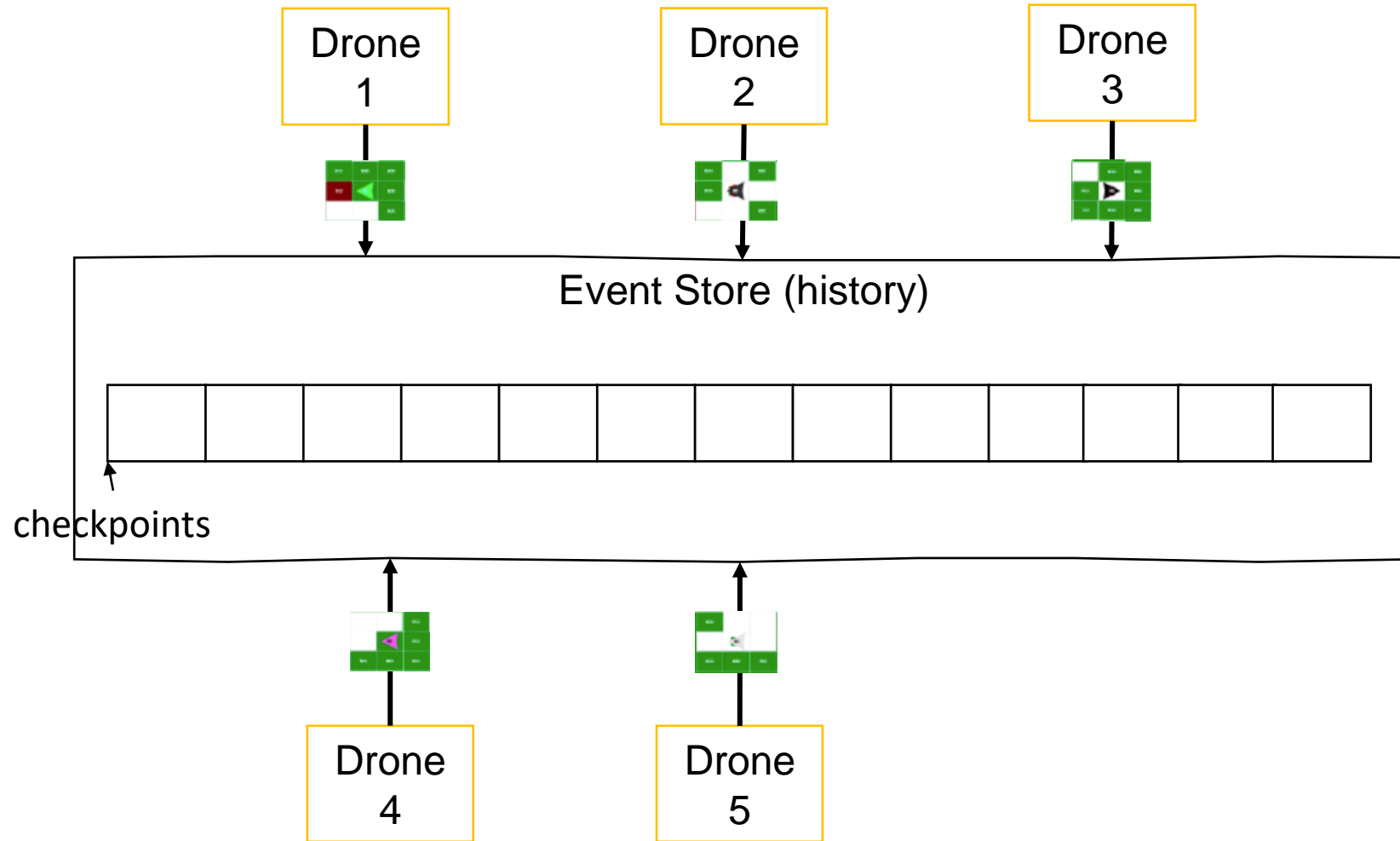
A variation: Event Sourcing



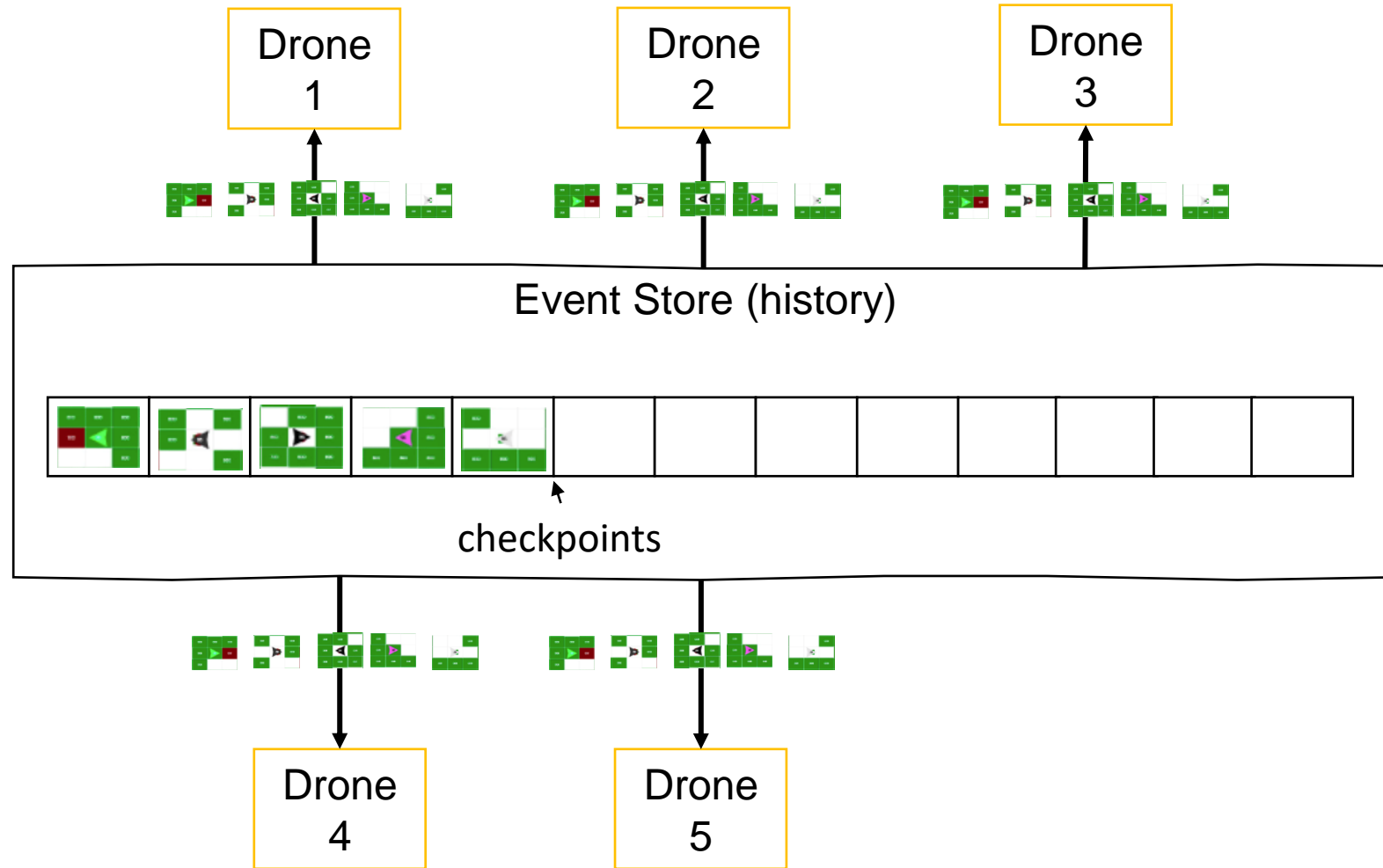
A variation: Event Sourcing

- The content of the Blackboard is the **history of raw events**
- **Atoms** in concurrent read/writes are **the individual events**
- **The internal state is a *replay* of the history** based on each specific agent logic
 - The agent logic can change at any time
 - Conflicts are delegated to the individual agents
 - **State is decoupled from the history**
- There are variations (CQRS) that segregate between
 - commands: updates (events added)
 - queries: continuously updated materialized views

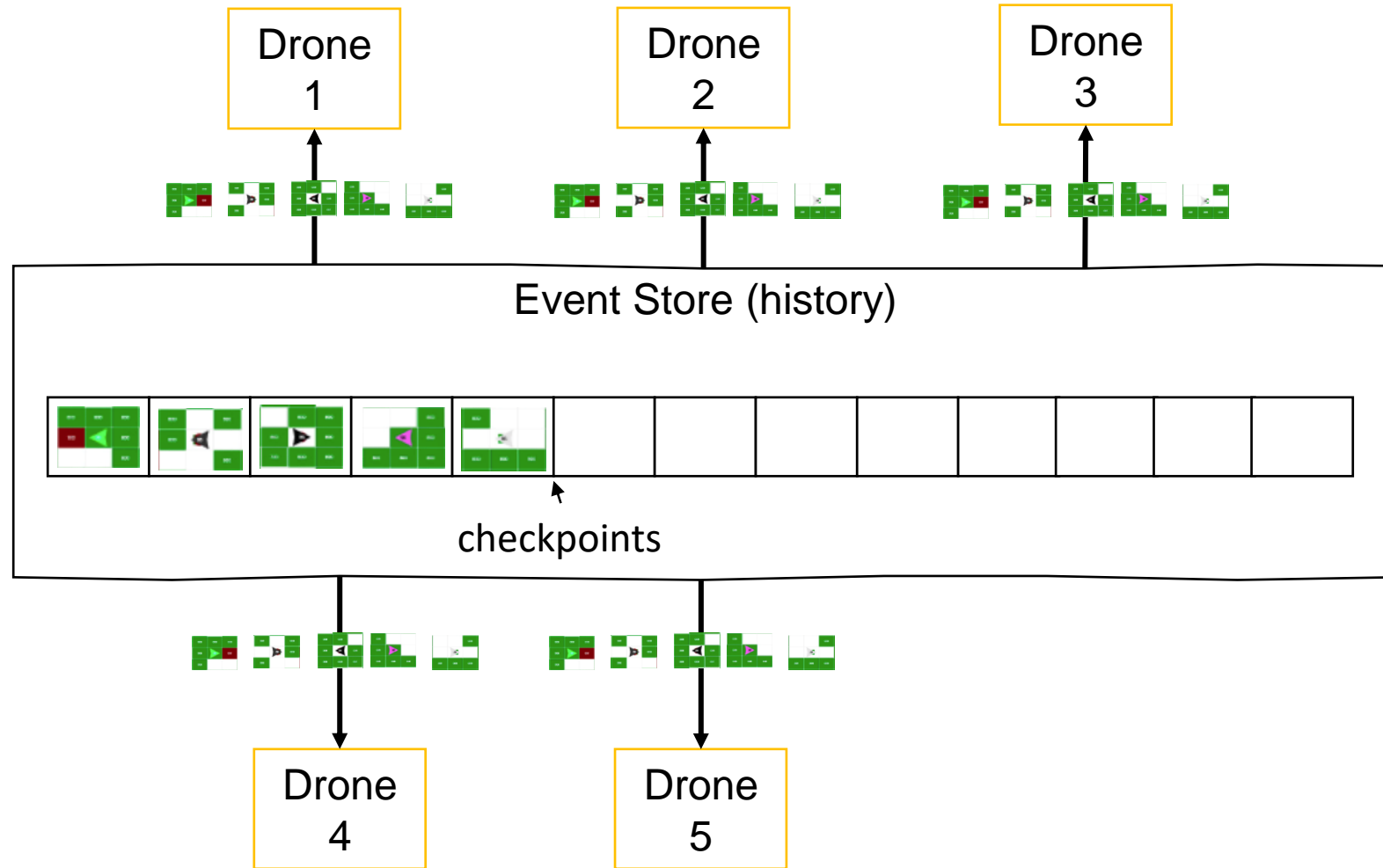
A variation: Event Sourcing Example



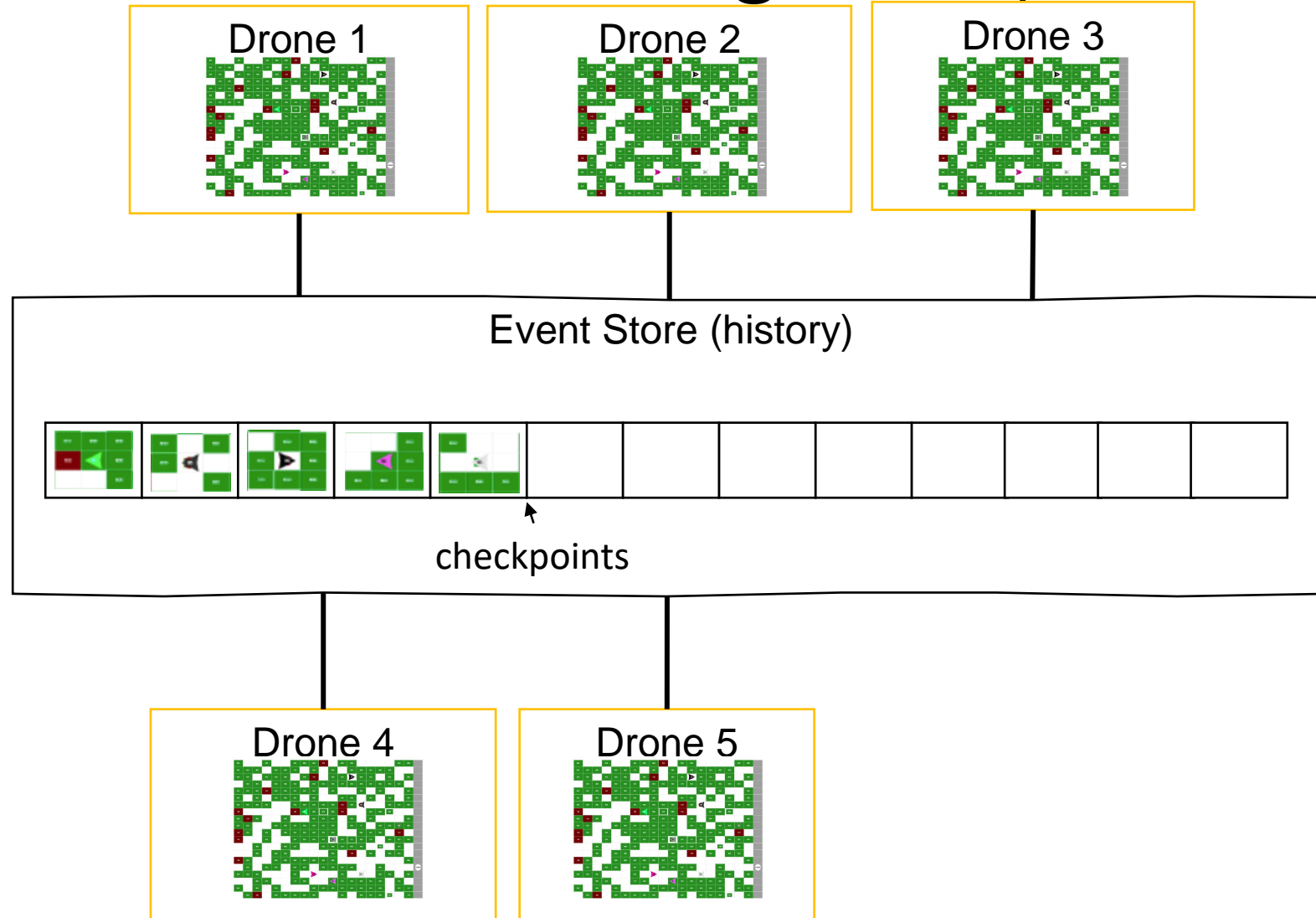
A variation: Event Sourcing Example



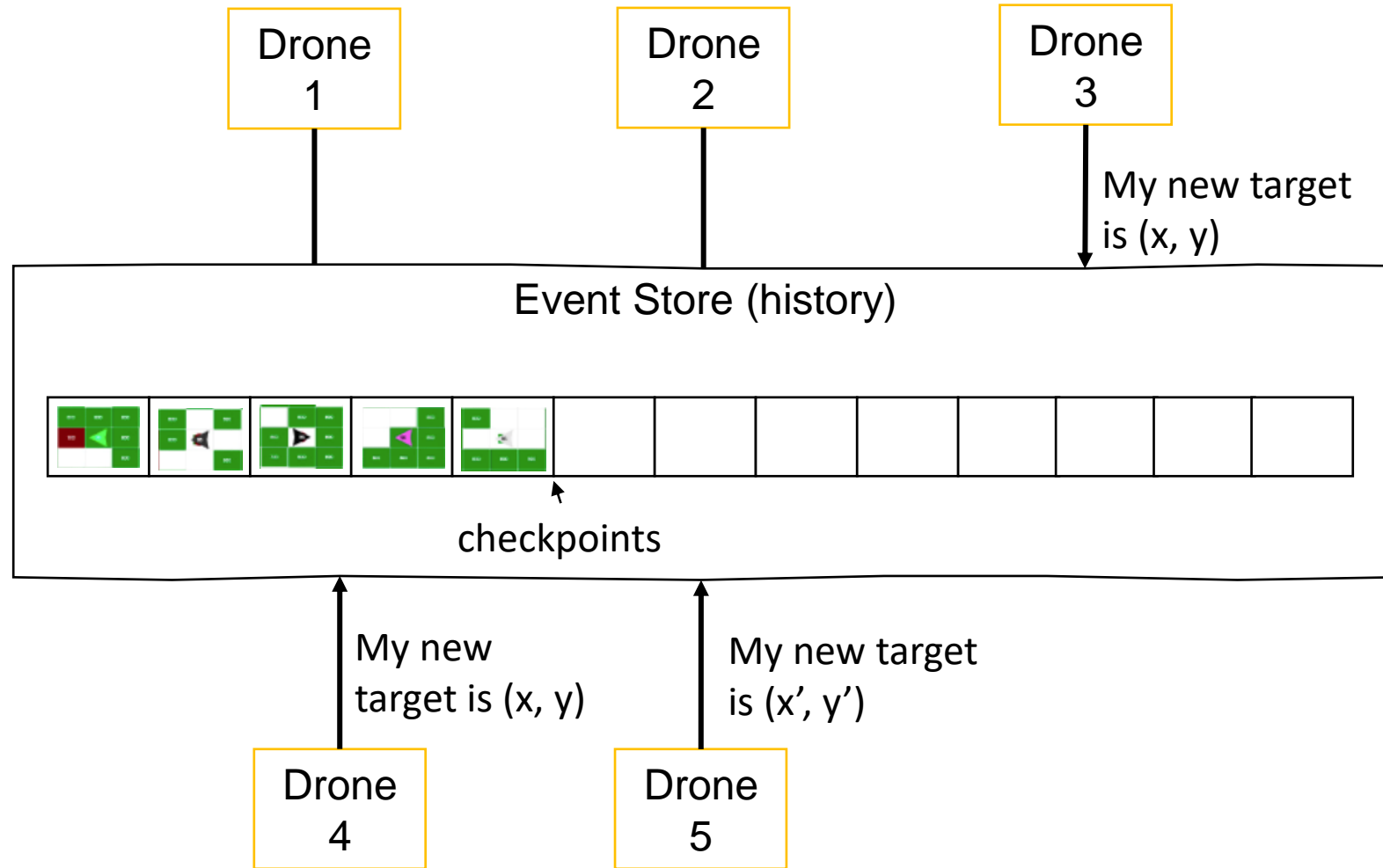
A variation: Event Sourcing Example



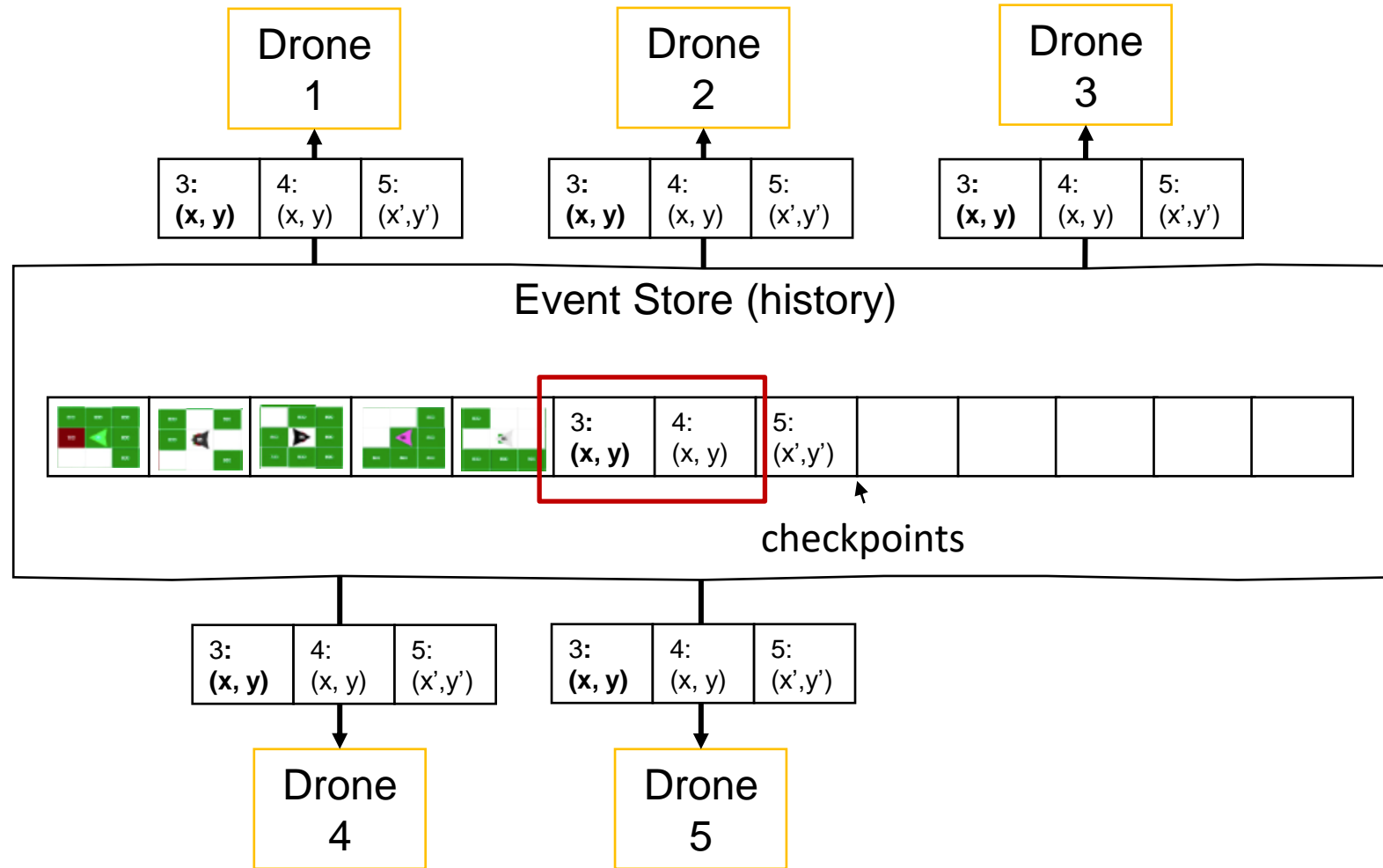
A variation: Event Sourcing Example



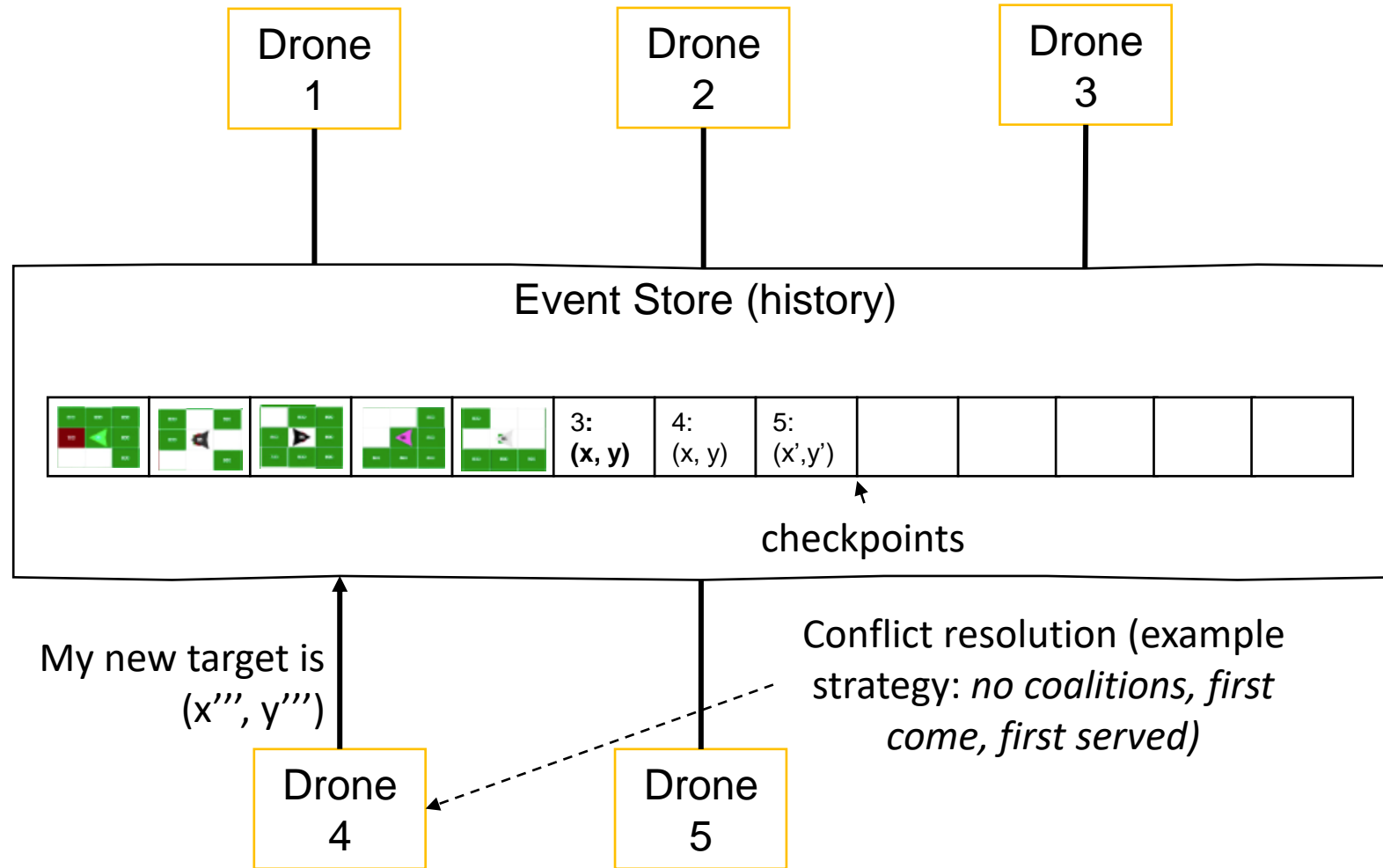
A variation: Event Sourcing Example



A variation: Event Sourcing Example

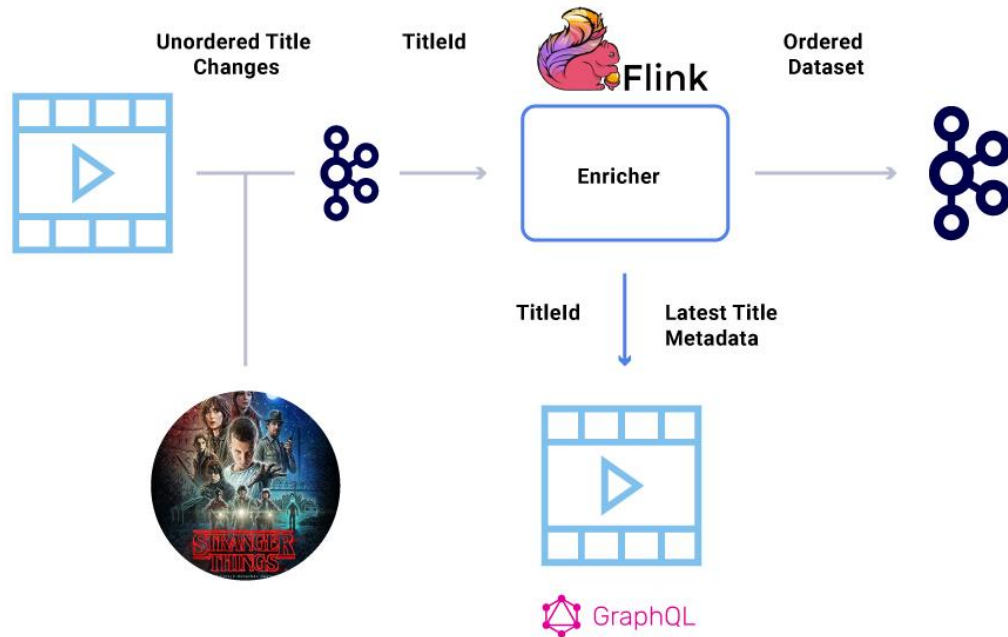


A variation: Event Sourcing Example



A variation: Event Sourcing in practice

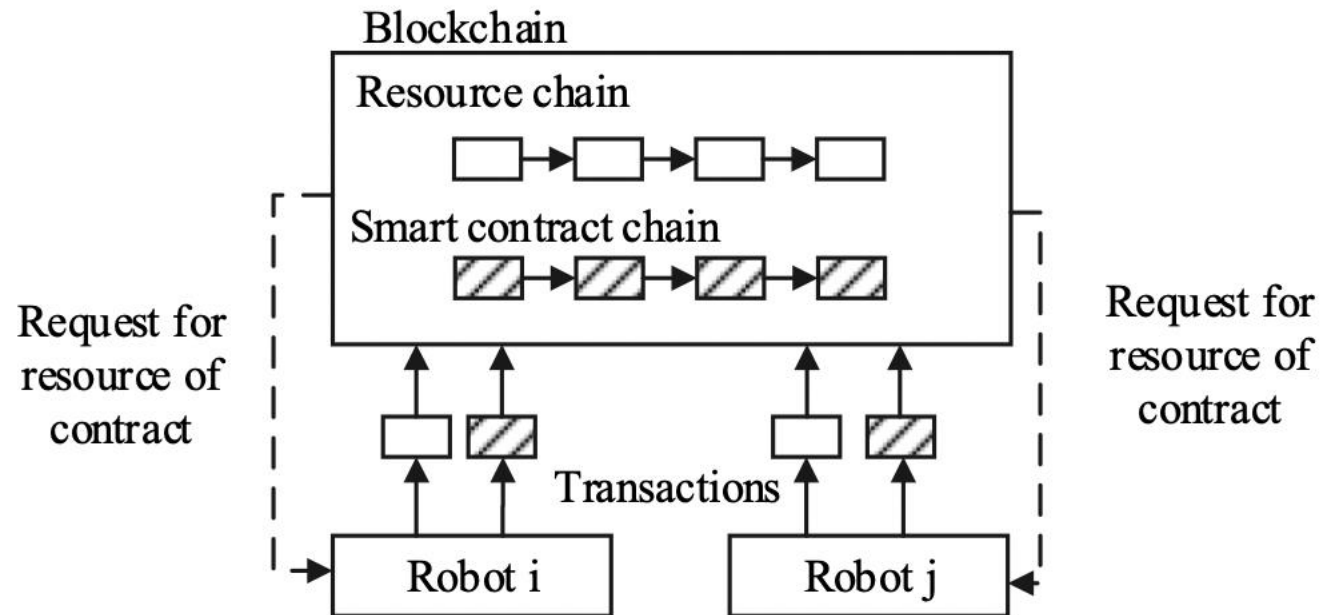
- It is a fundamental communication model in modern distributed systems
 - Microservices
 - Serverless functions
 - Lambda architecture
- Some tools used for event sourcing:
 - Apache Kafka
 - Redis
 - EventStore.com



Featuring Apache Kafka in the Netflix Studio and Finance World
<https://www.confluent.io/blog/how-kafka-is-used-by-netflix/>

Another variation: Blockchain

- **Centralized state**



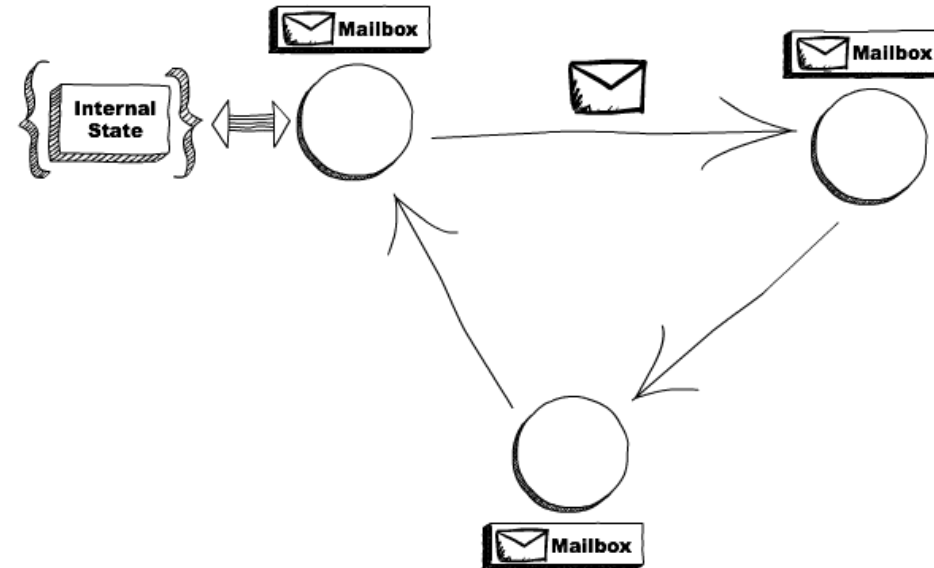
Teslya, N., & Smirnov, A. (2018). Blockchain-based framework for ontology-oriented robots' coalition formation in cyberphysical systems. In MATEC Web of Conferences (Vol. 161, p. 03018). EDP Sciences.

Message Passing

- There is **direct communication** between two agents
- There is **no concept of shared state**
- Many formal models (some examples):
 - π -calculus
 - Communicating sequential processes (CSP)
 - Petri Nets
 - Actor model
 - Bulk synchronous parallel (BSP)
- Some pragmatic approaches based on parts or combinations of formal models:
 - Rx* (Reactive programming)
 - Async IO (channels)
 - Software Transactional Memory (STMs)

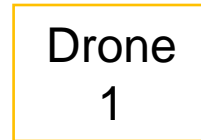
Message Passing: the Actor Model

- An actor **sends a message** directly to another actor through public addresses
- A reply is not necessary
 - *At most*, confirmation of reception
- Each agent has a **mailbox** of messages
 - Each message is treated sequentially (FIFO)
- Each agent has a **predefined behaviour** for each *message type*



- Send a finite number of messages to actors, and/or
- Spawn copies of the same actor, and/or
- Schedule a different behaviour for the next event

The Actor Model (example)



The Actor Model (example)

`addresses = findPeers()`

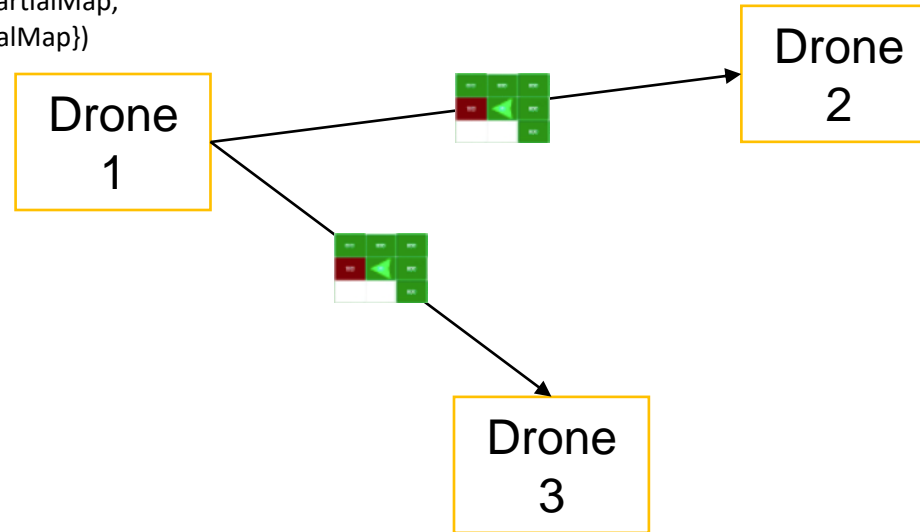
Drone
1

Drone
2

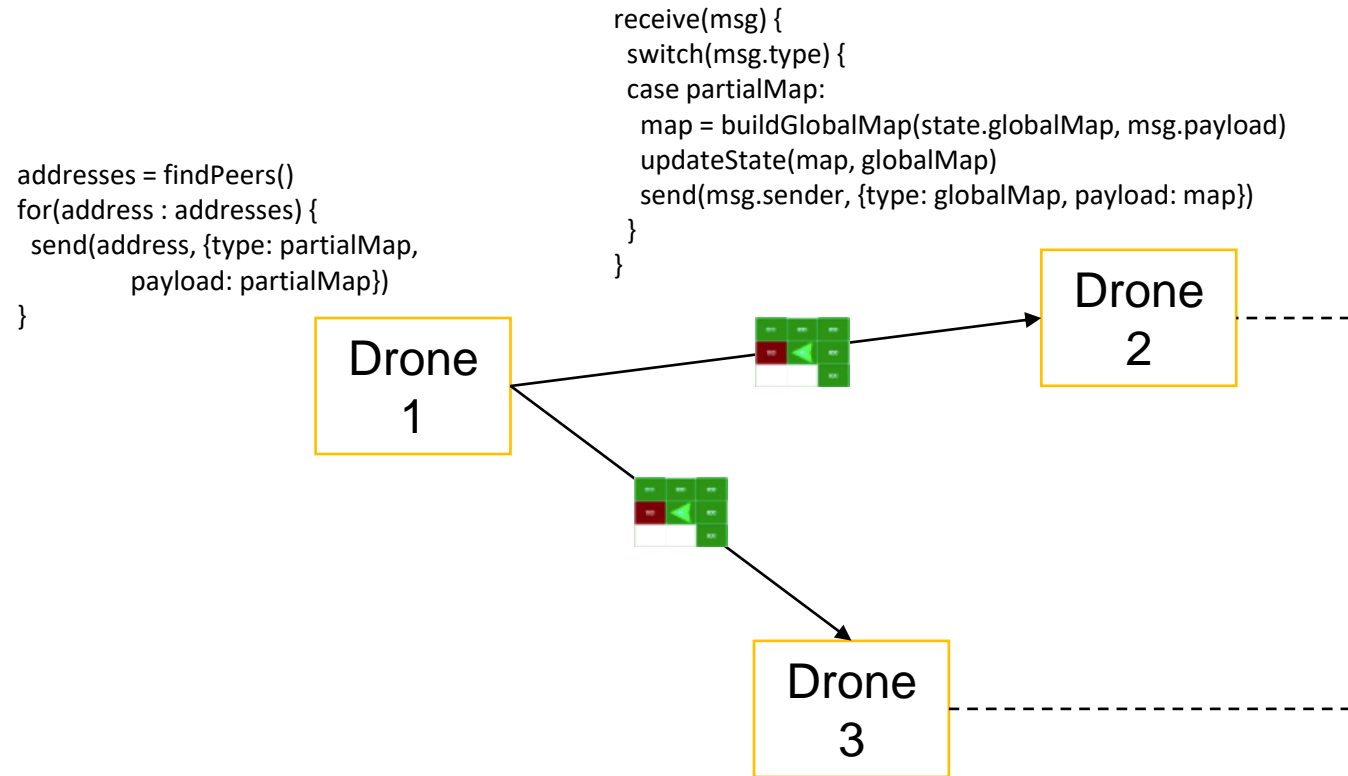
Drone
3

The Actor Model (example)

```
addresses = findPeers()
for(address : addresses) {
  send(address, {type: partialMap,
                  payload: partialMap})
}
```

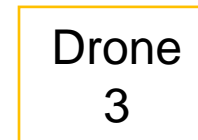
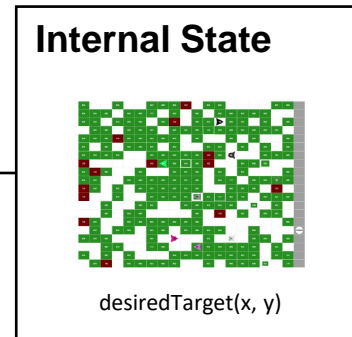
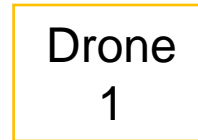


The Actor Model (example)

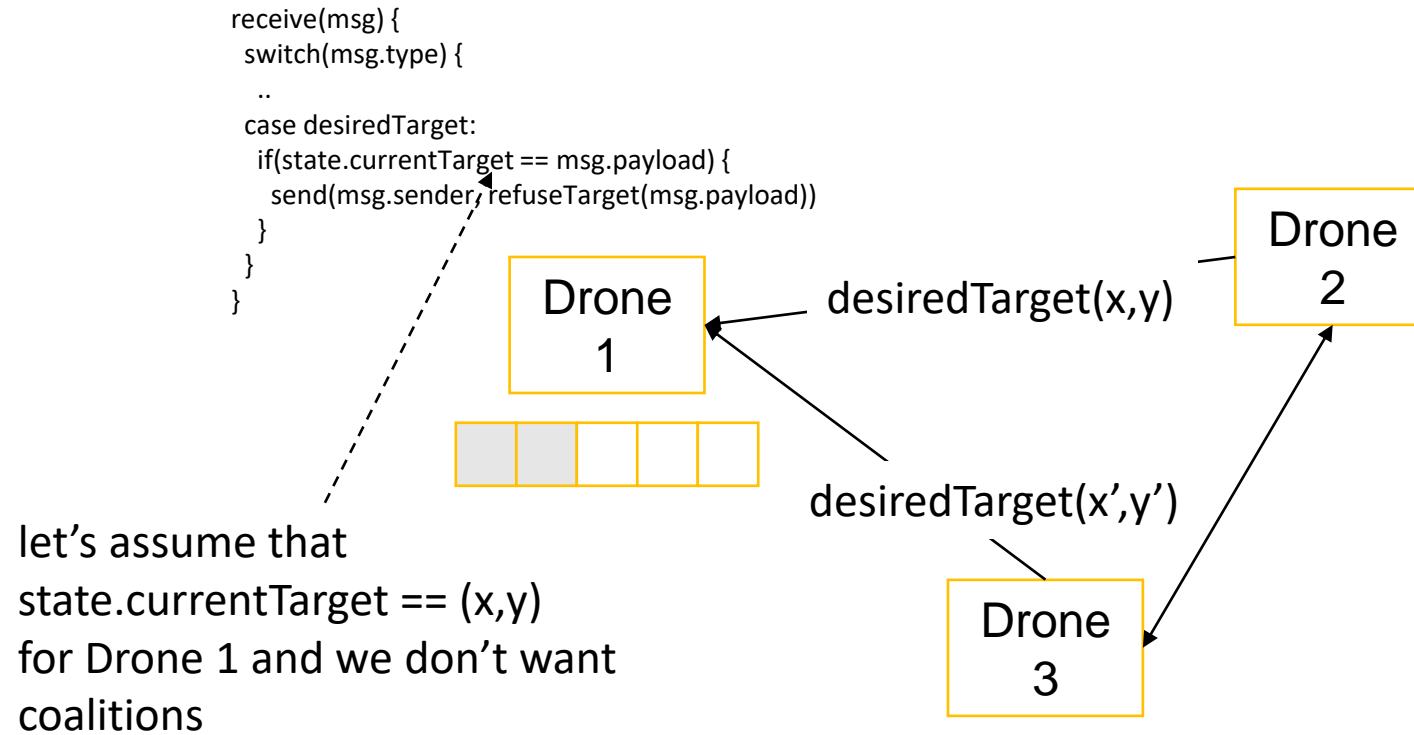


The Actor Model (example)

```
receive(msg) {  
  switch(msg.type) {  
    case partialMap:  
      map = buildGlobalMap(state.globalMap, msg.payload)  
      updateState(map, globalMap)  
      send(msg.sender, {type: globalMap, payload: map})  
    }  
  }  
}
```

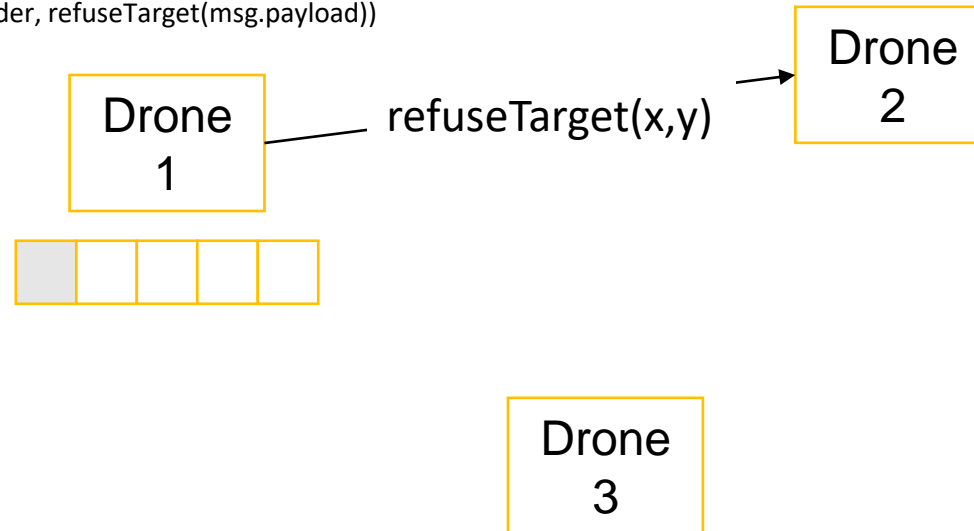


The Actor Model (example)



The Actor Model (example)

```
receive(msg) {  
  switch(msg.type) {  
    ..  
    case desiredTarget:  
      if(state.currentTarget == msg.payload) {  
        send(msg.sender, refuseTarget(msg.payload))  
      }  
    }  
  }  
}
```



The Actor Model (example)

```
receive(msg) {  
  switch(msg.type) {  
    ..  
    case desiredTarget:  
      if(state.currentTarget == msg.payload) {  
        send(msg.sender, refuseTarget(msg.payload))  
      }  
    }  
  }  
}
```

Drone
1

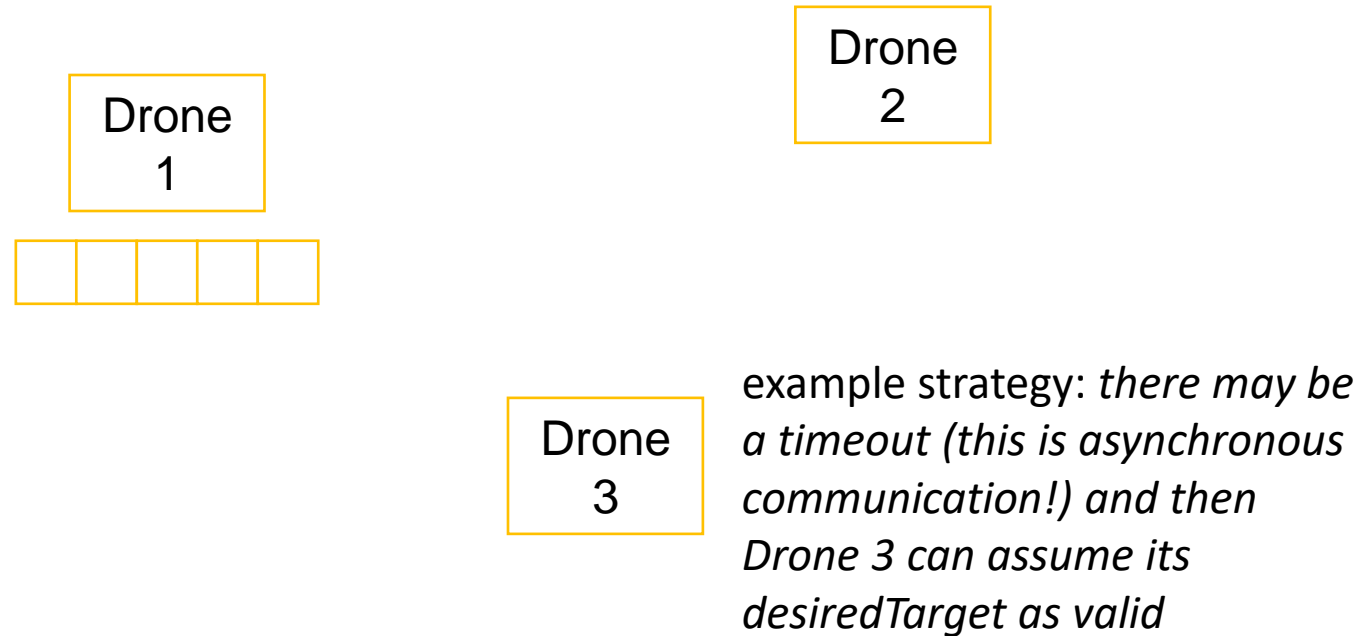


message from Drone 3 is
read next, what if Drone
1 takes no action?

Drone
2

Drone
3

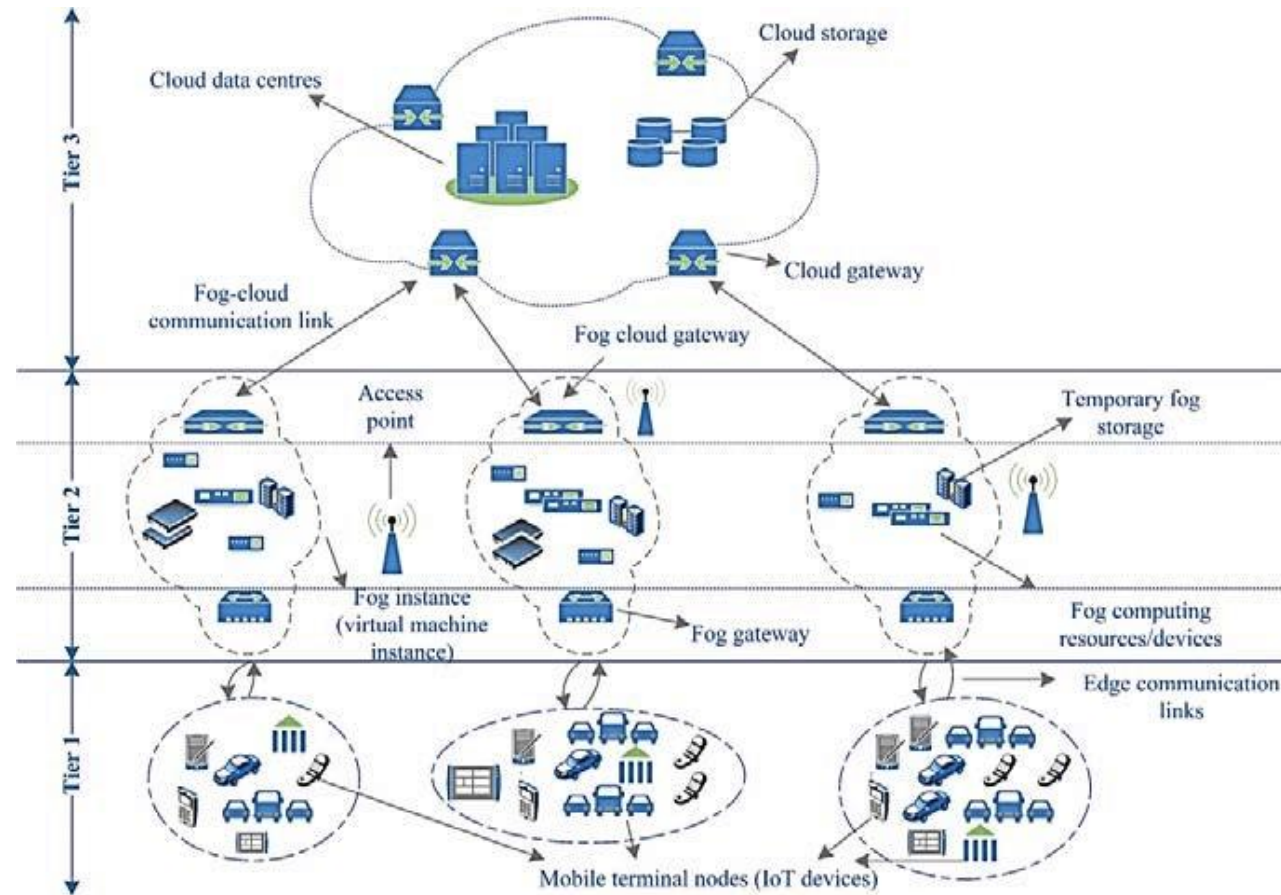
The Actor Model (example)



The Actor Model in practice

- Benefits
 - Low overhead
 - Fault tolerance (modular protection)
 - Very high scalability and portability
- Disadvantages
 - States may be inconsistent (many sources of truth)
 - Favors push (reactive) patterns
- It is a main construct in several languages
 - Scala (Akka)
 - Erlang
 - Smalltalk
- It is available as a library in almost every language
- JADE implements a communication model that can be reduced (*with some caveats*) to the Actor Model

A recent model: edge computing



Taneja, M., & Davy, A. (2016). Resource aware placement of data analytics platform in fog computing. *Procedia Computer Science*, 97, 153-156.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5), 637-646.