HSLU Lucerne University of Applied Sciences and Arts

Multi Agent Reinforcement Learning

Reinforcement Learning
December 22, 2022

FH Zentralschweiz

Multi Agent Reinforcement Learning

Multi-Agent Systems have a long history:

"A multi-agent system is a group of autonomous, interacting entities sharing a common environment, which they perceive with sensors and upon which they act with actuators"

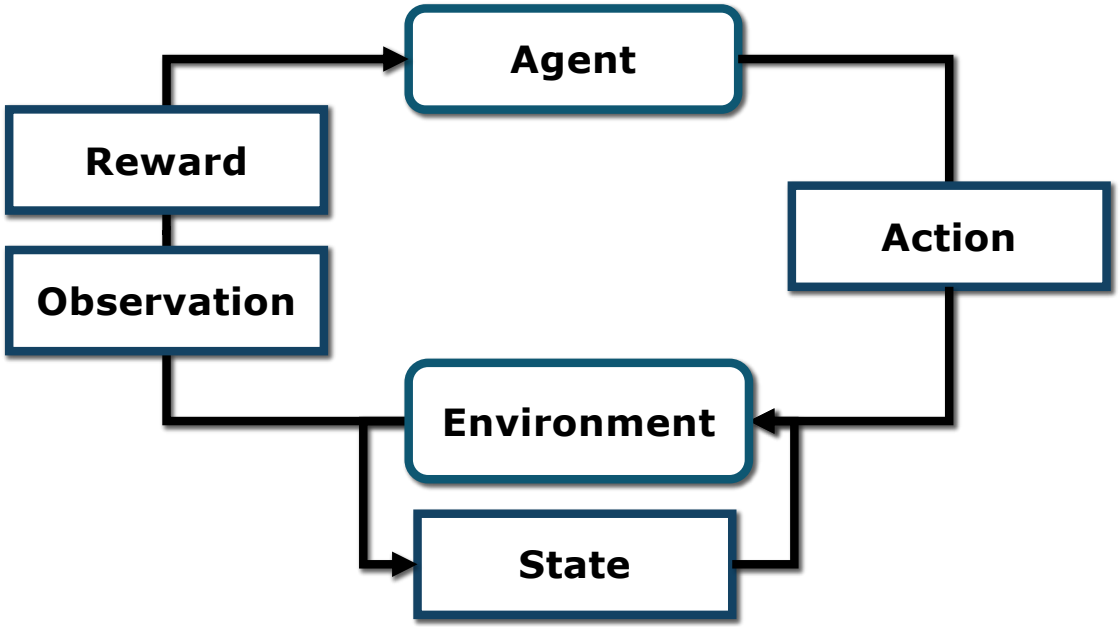
In Reinforcement Learning

- Shared Environment
- Each agents has its own observations, rewards, actions and goals
- (Agents might need to coordinate to achieve their own goals)

MARL Agents

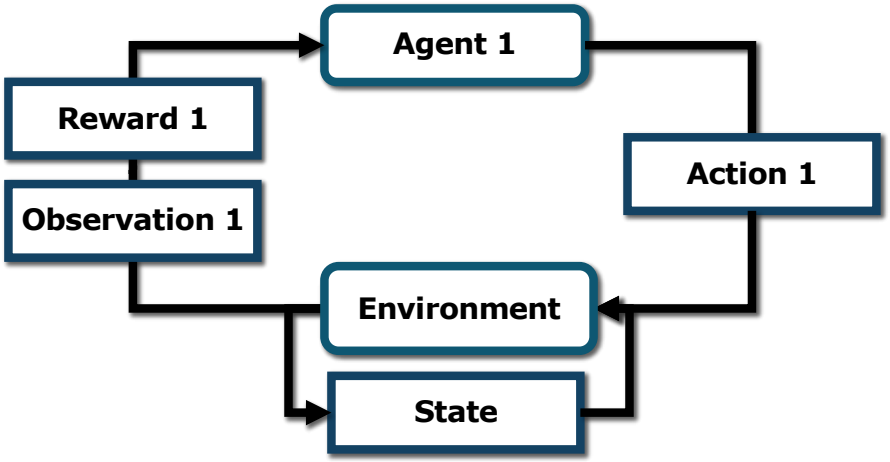
- Agents could be **collaborative** with the same goal and only receive a joint rewards.
- Agents could be collaborative with same or different goals and receive individual and/or a joint reward:
  - For example if multiple steps are necessary to reach the goal, the agents could get rewards for each step and for the goal
- Agents could be **competitive** with different goals (and rewards), for example in competitive games

Single Agent RL Environment

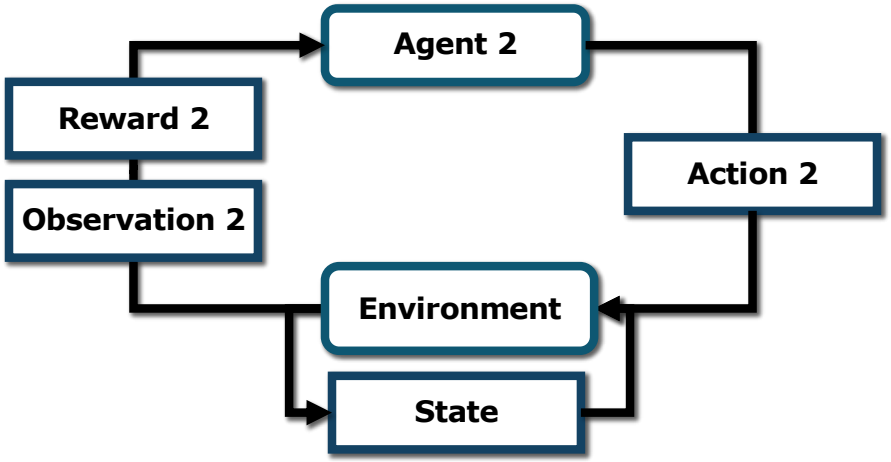


Turn based environments

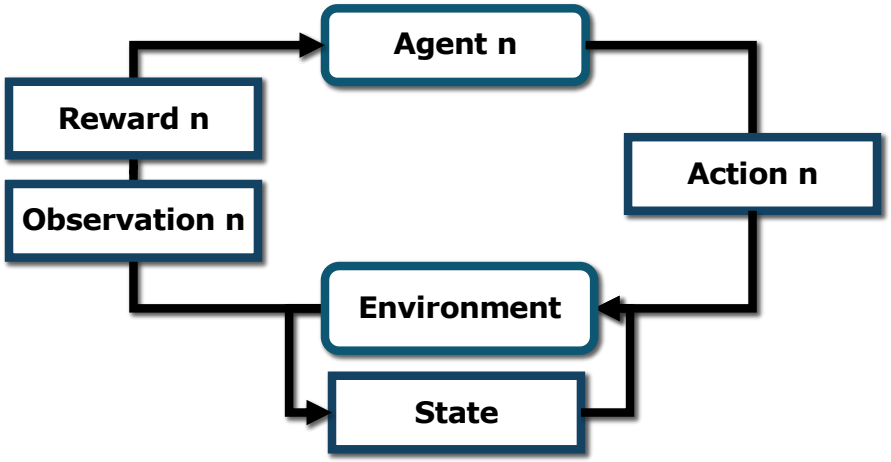
Time Step 1



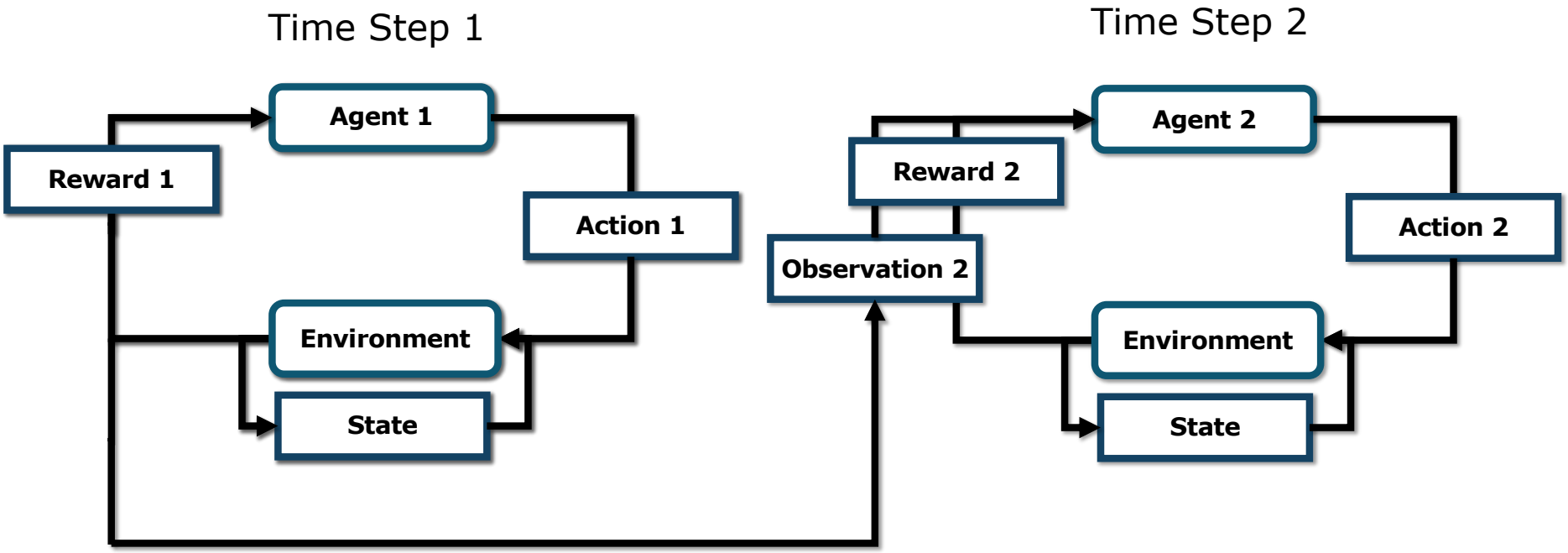
Time Step 2



Each agents decides its action
based on the current observation



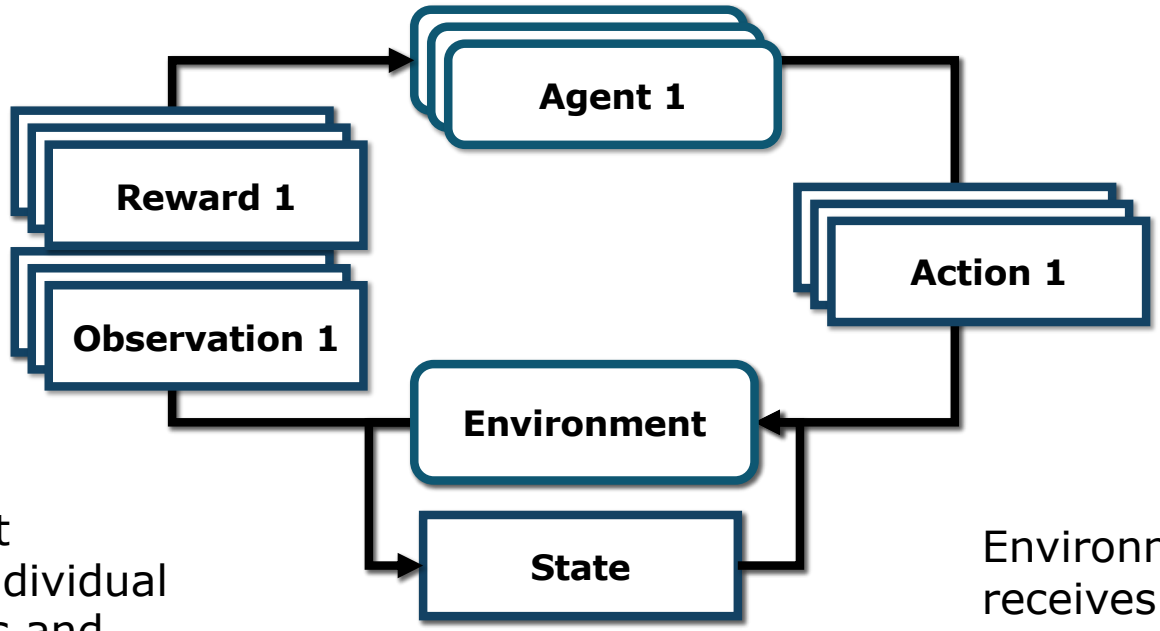
Turn based environments



- The action from one player will change the environment, which then can generate the observation for the next player
- Rewards could be available immediately or only after all players have done an action
- For example in a turn based game such as Nine Mens Morris (Mühle), Jass or Poker

Simultanous, Pseudo-Real Time

Agents decide actions simultanously

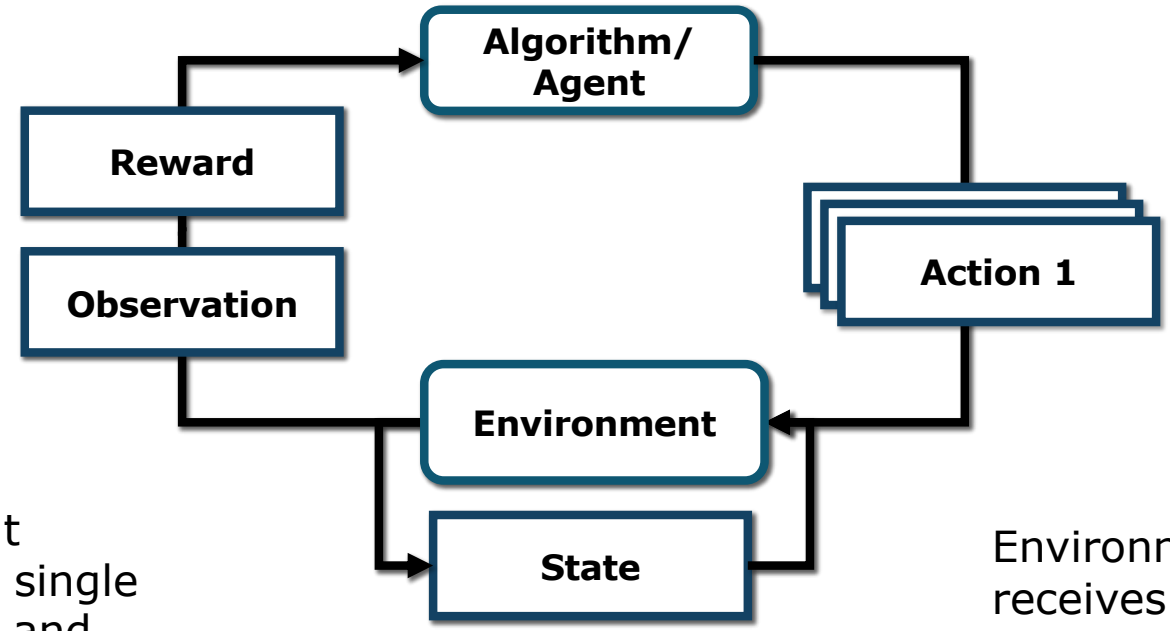


Environment calculates individual observations and rewards

Environment receives all the actions for this step

Joint Actions by a single Algorithm / Agent

A single algorithm or agent decides all actions



Environment calculates a single observation and reward

Environment receives all the actions for this step

Challenges in Multi Agent Reinforcement Learning

- Non-Stationary environment:
  - For a single agent, the other agents act as a change in the environment
- Scalability (observations, joint action spaces)
  - Action spaces may grow non-linearly with the number of agents, making it infeasable to directly learn them
- Sparsity of Rewards
  - Complicated environments and tasks may only give rewards sparingly, for example at the end of a game. This makes it hard to train an agent.

Examples / Applications

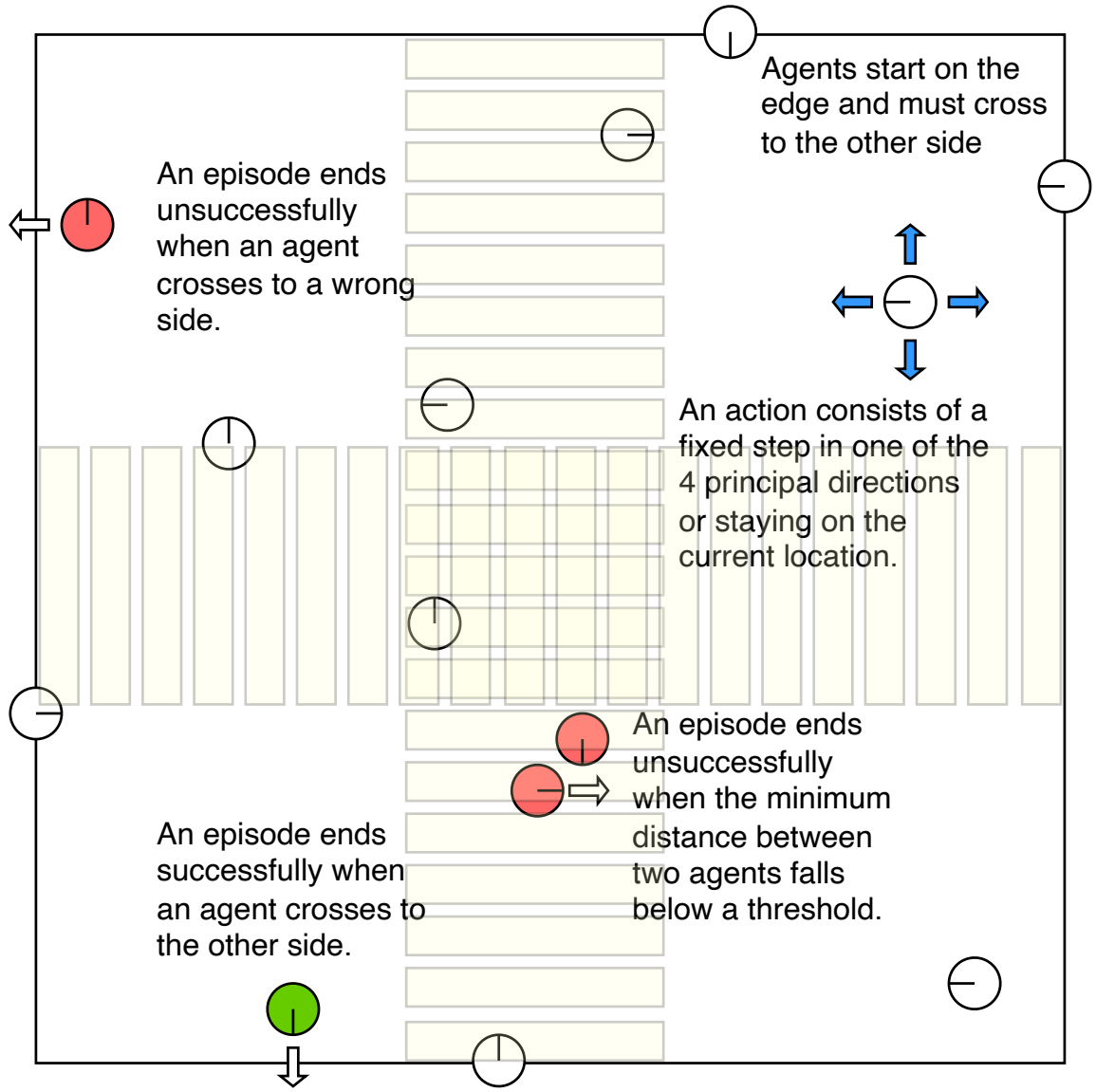
Games (Chess, Go, League of Legends, StarCraft)

Robotic Championsships

Market Negotiations

Tokyo Shibuya Crossing

Crossing Simulation



Crossing Simulation

- Each bot is controlled by one agent.
- All agents have identical tasks and identical goals
- All agents use the same algorithm / policy
- The environment is unknown, there is no model that can be used to try actions and the exits are not known (but modelled by some rewards)

Observations

Observation	Type	Remarks
Position	[2]	Position of bot
Velocity	[2]	Velocity vector of bot (due to last action)
Goal	[4]	One-hot encoded (N, E, S, W)
Distance to goal	[1]	Distance to goal

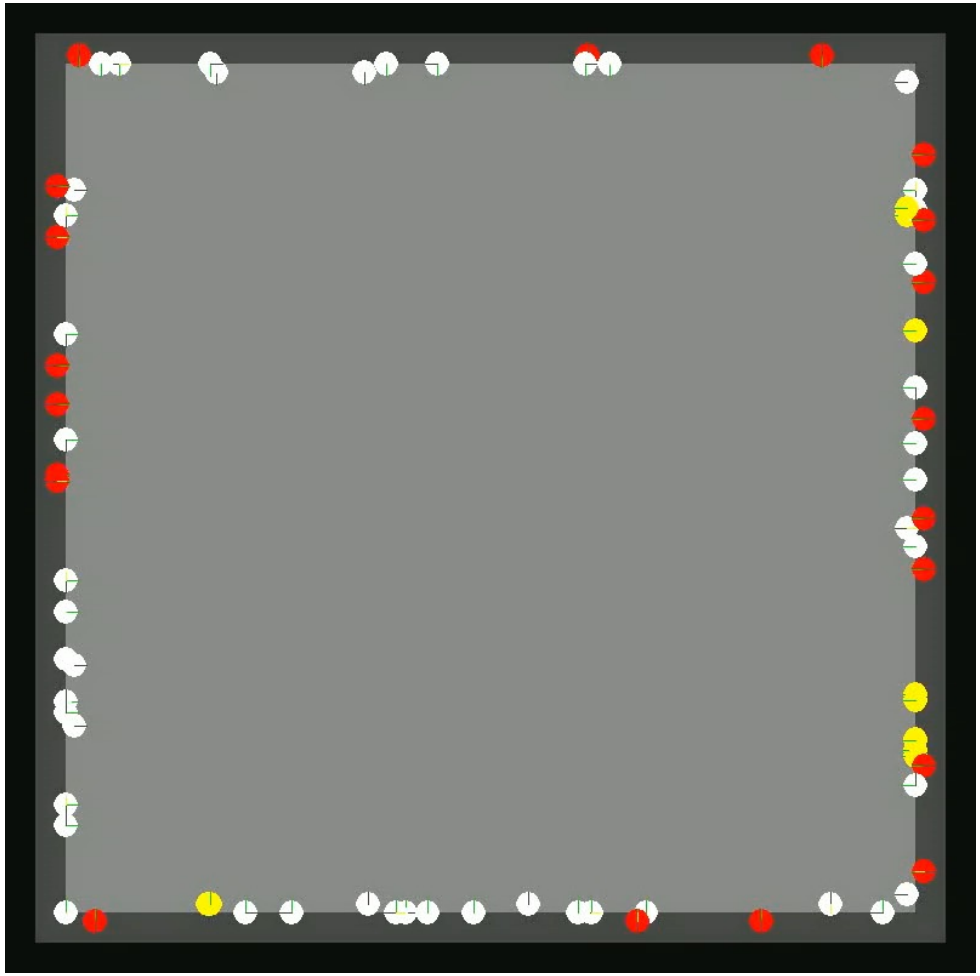
Edge Observation	Type	Remarks
Direction to neighbor	[2]	Direction to other bot
Distance to neighbor	[2]	Distance to other bot

Rewards

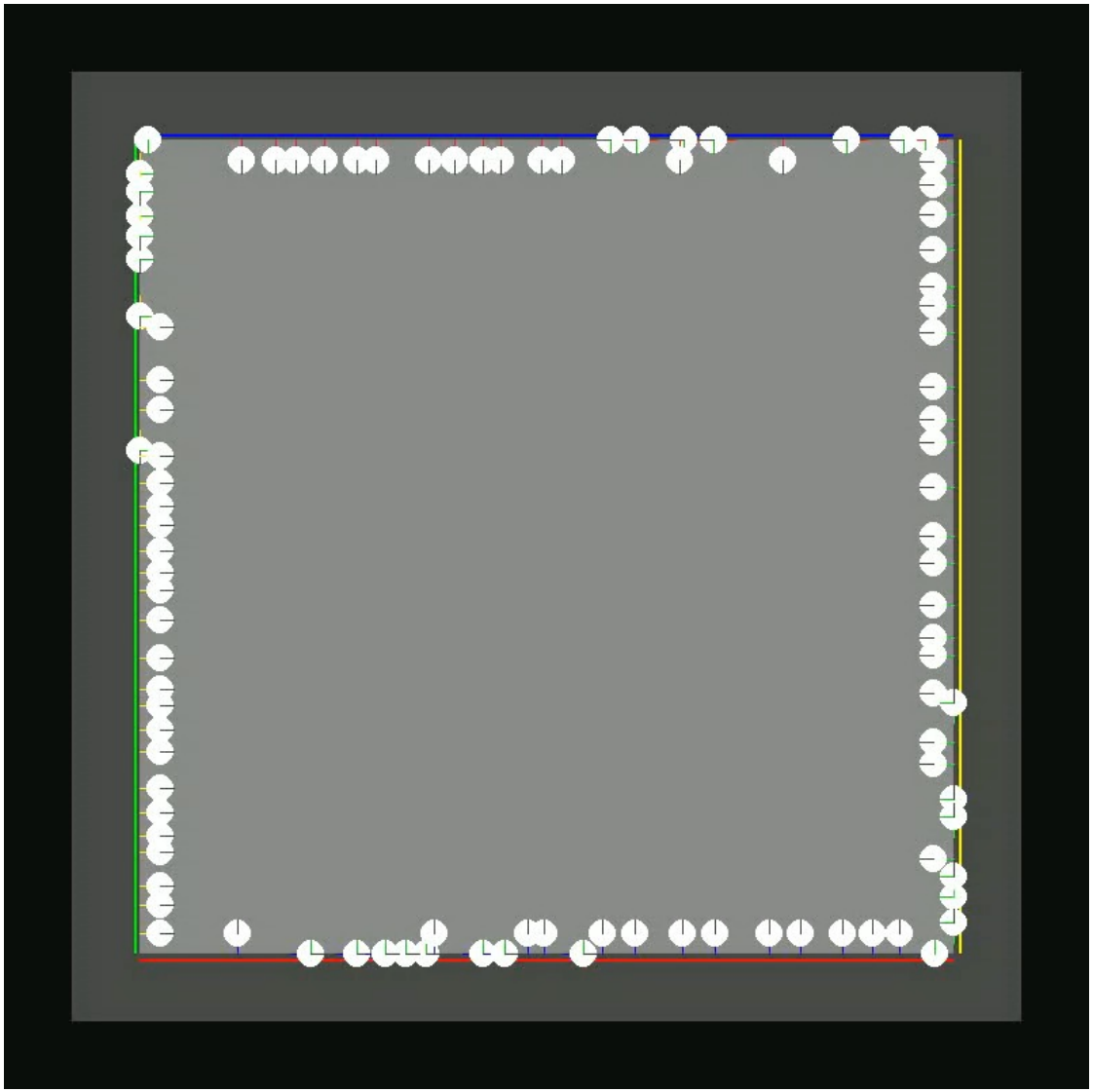
Reward	Default Value	Remarks
Timestep	-0.01	Reward at each time step
Distance to goal	0.01	Linearly decreases with distance from goal
Distance to side	0.002	Linearly increases with distance from wrong sides until a maximal distance
Distance to other bots		Linearly increases with distance from other bots up to maximal distance
Fall	-1.0	Reset to start position
Exit wrong side	-1.0	Reset to start position
Exit correct side	1.0	Goal reached, episode ends

Crossing: Metrics for Reinforcement Learning

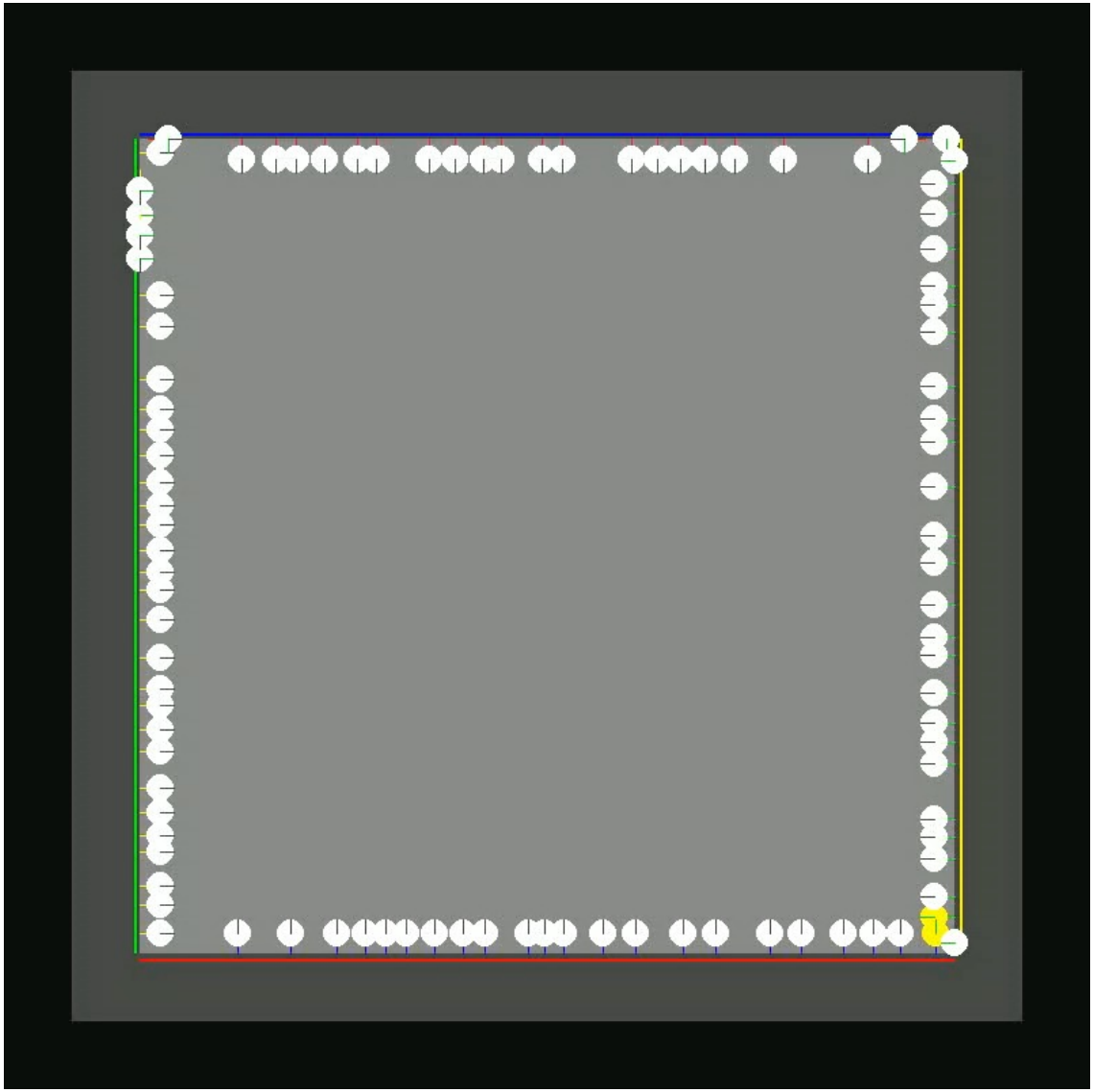
- Generally, in reinforcement learning we aim to maximize the cumulative rewards
- However, shaping the rewards in order to induce the desired behavior of the agents is one of the problems of RL
- The evaluation of the behavior should be independent of the action reward formulation
- For goal-oriented tasks, the following metrics seem adequate:
  - **Accuracy**: The percentage of episodes that complete successfully
  - **Steps**: The average number of time steps to complete a successful task



APPO with few neighbor observations



APEX DDPG



“Small” Multi-Agent Systems: Game Theory

Normal-form games:

- Finite set of agents N = {1,...,n}
- For each agent i:
  - Finite Set of actions A_i
  - Reward function: $u_i: A \rightarrow \mathbb{R}$, where $A = A_1 \times \cdots \times A_n$, (joint action space)

Example Prisoner Dilemma

- Two prisoners are interrogated in separate rooms
- Each prisoner can Coorporate (C) or Defect (D) (stay silent)
- Reward Matrix:

	Prisoner B	
Prisoner A	Cooperate	Defect
Cooperate	(-1,-1)	(-5,0)
Defect	(0,-5)	(-3,-3)

Solving a Multi-Agent Game

How do we solve such a game?

- If the game has common rewards, then solving the game is like solving a MDP
  → Find a policy that maximizes the returns for all states s

If the agent rewards differ, what should a policy optimize?

There are different solutions concepts:

- Minimax solution
- Nash equilibrium
- Pareto-Optimality
- No-Regret
- ....

Nash Equilibrium

"No agent can improve their reward by changing their own strategy"

(Every agents plays the best response to other agents)

	Prisoner B	
Prisoner A	Cooperate	Defect
Cooperate	(-1,-1)	(-5,0)
Defect	(0,-5)	(-3,-3)

The only Nash Equilibrium is (Defect / Defect)

Repeated Game

Learning is to improve performance via experience

A normal-form game is a single interaction → no experience

Experience comes from **repeated** interactions

Repeated game:

- Repeat the same normal-form game for time steps t = 0,1,2, ...
- Learn by modifying the policy based on the history of the actions and rewards

Independent Learning: Centralized training with decentralized execution (CTDE)

- A single agent RL algorithm such as Q-Learning, Policy-Gradient or Actor Critic Method is used.
- A centralized algorithm is trained (and shared) for all agents.
- Each agent executes the algorithm and receives its own observations and rewards.

Independent Learning: Centralized training with decentralized execution (CTDE)

Advantages:

- Experience for all agents go into the training data

Disadvantages:

- Algorithms must generalise to the behaviour for all agents.
- (I.e. for the crossing, there can not be a different policy learned for the agents crossing from left to right and the ones crossing from right to left)

Independent Learning: Decentralized Learning

- In a decentralized setting, each agents learns its own model
- For example using Actor-Critic this is called *Independent Actor Critic* (IAC)

- A combination of both cases uses a decentralizes policy function but a centralized critic, this is called *Independent Actor with Centralized Critic* (IACC)

Joint Action Learning

A single algorithm learns the joint actions for all agents.

Advantages:

- No information exchange necessary

Drawbacks:

- High dimensional action space
- Only a single reward → Credit Assignement Problem
- Can be hard to train when the number of agents increases

Star Craft Example

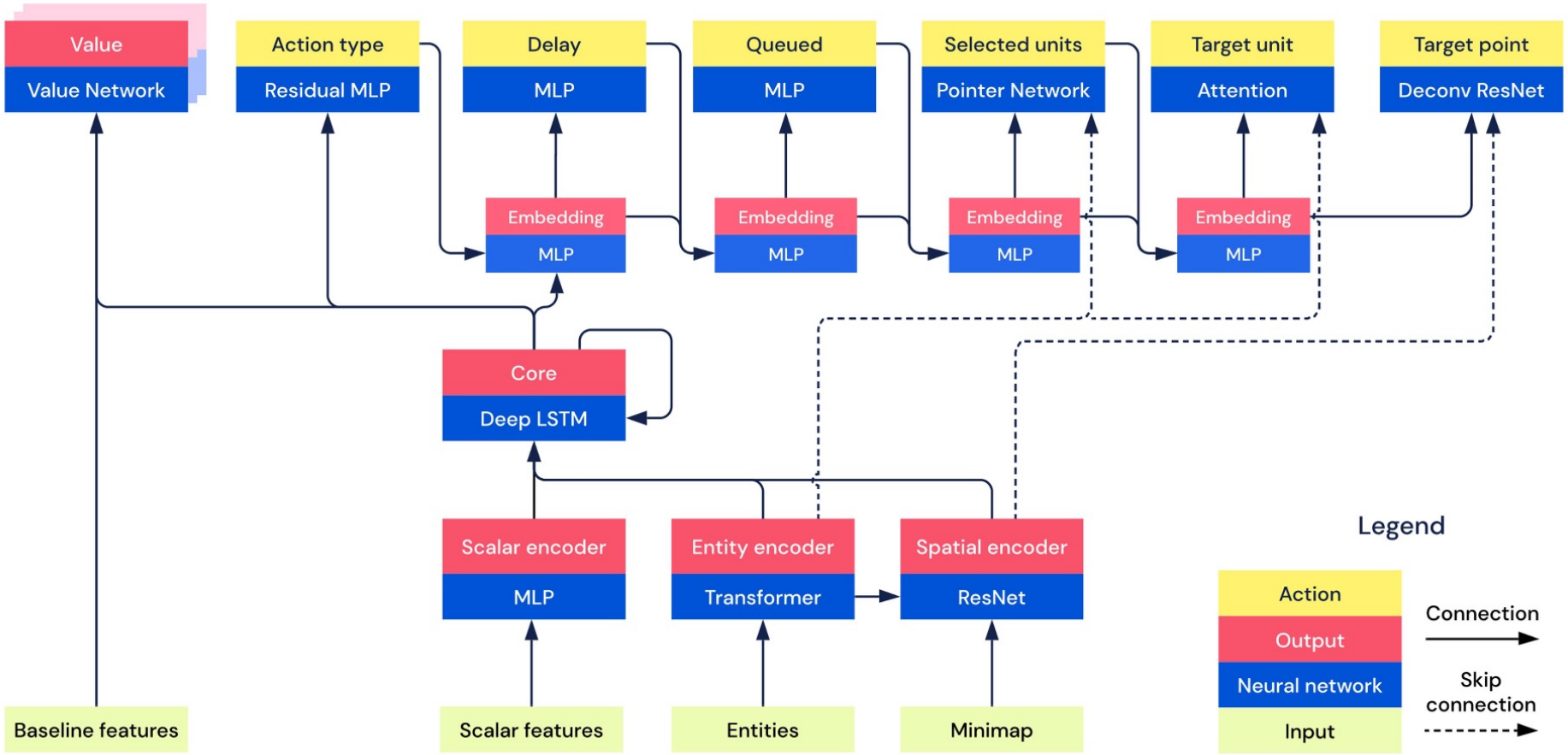
Policy represented by a neural network

$$\pi_\theta(a_t|s_t, z)$$

where s_t is the vector of all observations and actions so far, and z a statistic learned from human data

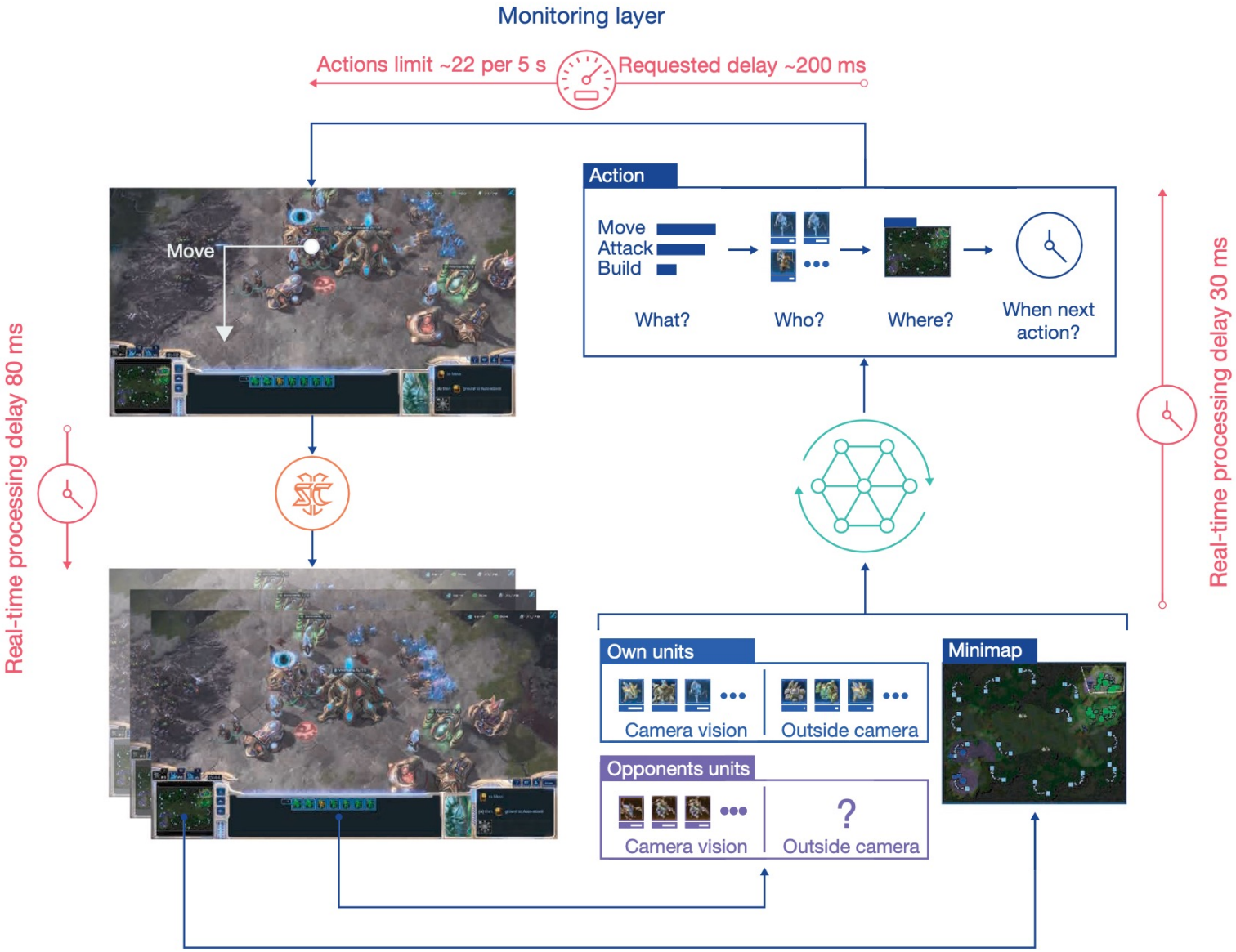
- Initially trained with supervised learning on human data
- Then RL learning with Actor-Critic approach
- Introduces League training:
  - Train against different types of opponents sampled from a collection of policies

Star Craft Architecture



Star Craft

a

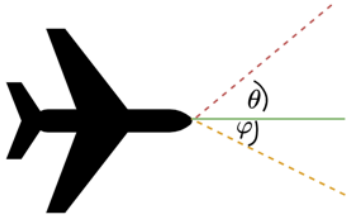

Monitoring layer
Actions limit ~22 per 5 s
Requested delay ~200 ms
Move
Action
Move
Attack
Build
What?
Who?
Where?
When next action?
Real-time processing delay 80 ms
Real-time processing delay 30 ms
Own units
Camera vision
Outside camera
Minimap
Opponents units
Camera vision
?
Outside camera

Examples: MAM: Machine-Managed Air Mobility in Swiss Upper Airspace

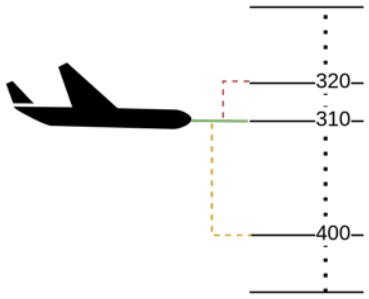
Innosuisse Project between HSLU (ABIZ), SkySoft and SkyGuide

- Aircrafts want to fly as close to their flight plan as possible without getting too close to other aircrafts.
- RL agents provide controlling

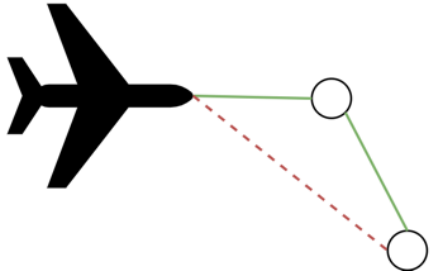
ATM Simulation – Action Spaces



Change
Heading (CH)



Change Flight
Level (CFL)



Skip Route
Waypoint (SW)

Joint = CH + CFL + SW

Example: Joint Action Space

Learning in MARL

What is learned in multi-agent learning:

- Best own action

- But potentially also how other agents behave

Example:

Autonomous cars

Crossing -> If all agents use the same algorithms, the solution is different from when you would have a random agent (or another policy for agents) that distur

Communication

Communication can be viewed for different objectives in MARL:

- Exchange of information about the environment: Each individual agent may not view the exact state, but only its own observation which is a partial view of the global environment
- Agents might also exchange information about the action that they are considering to make, so that other agents can react on them

- Communication can most generally be modelled by **exchanging messages** between agents.

Graph Neural Networks for communication

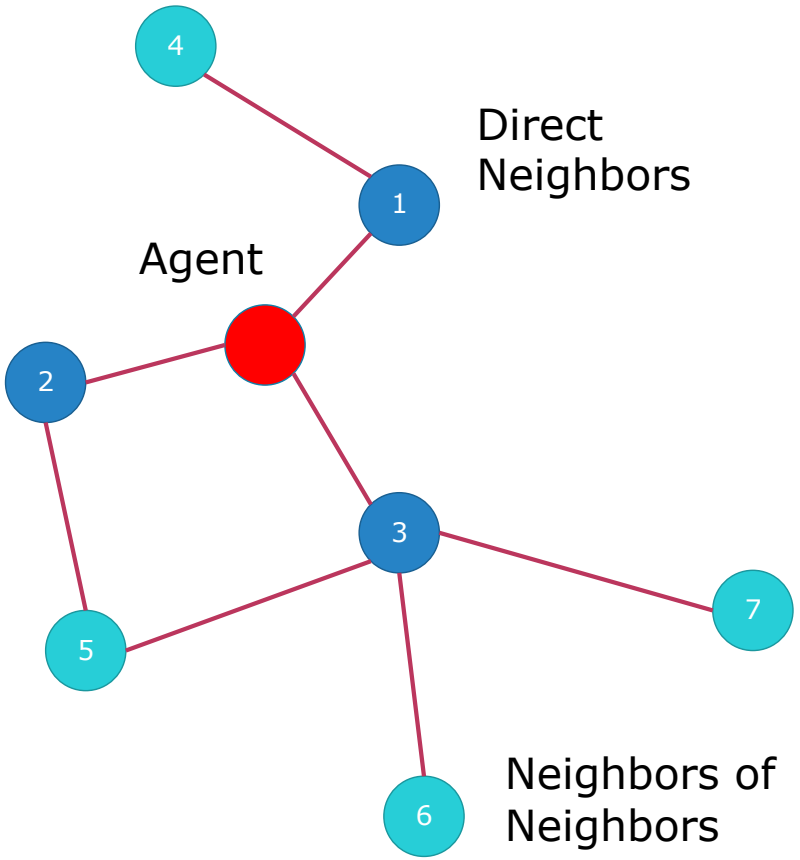
Allowing all agents do exchange messages might lead to a very complex environment.

We could also model the agents as graph, where messages can only be retrieved (or sent) to the direct neighbors.

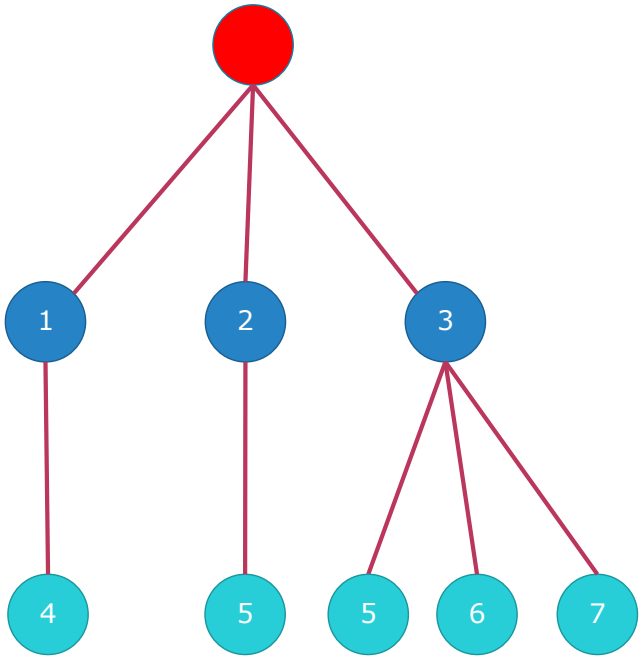
This could be modelled by graph neural networks

Graph Neural Networks

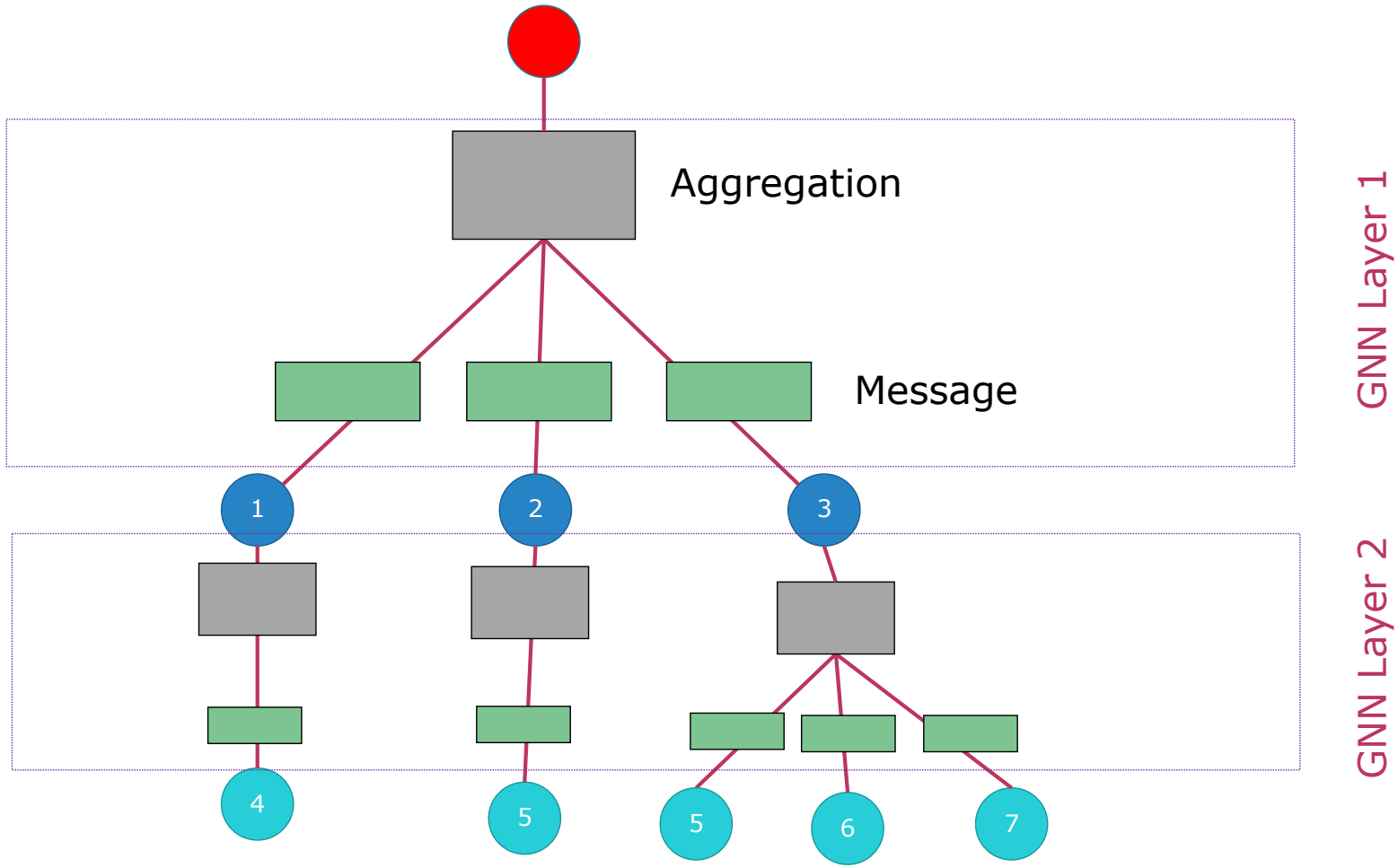
Graph Representation



Computational Graph



GNN Framework



GNN Framework

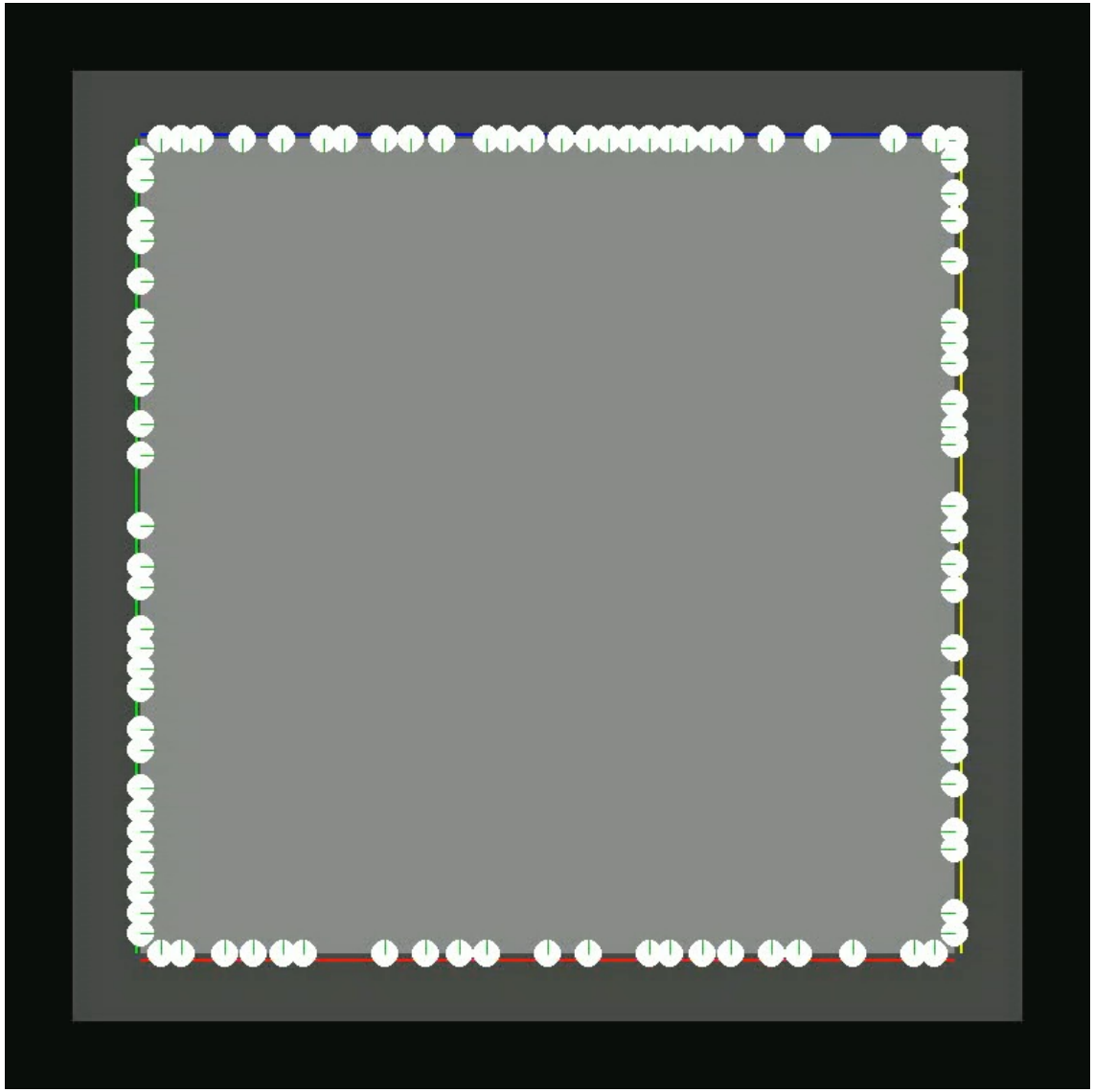
Messages:

- Use Neural Networks to calculate messages from node properties (and possibly edge properties)

Aggregation:

- Classical: Use Mean or Max from Messages
- Use Nodes from Recurrent Networks such as Simple RNNs, LSTMs etc.
- Use Multiheaded Attentation Networks

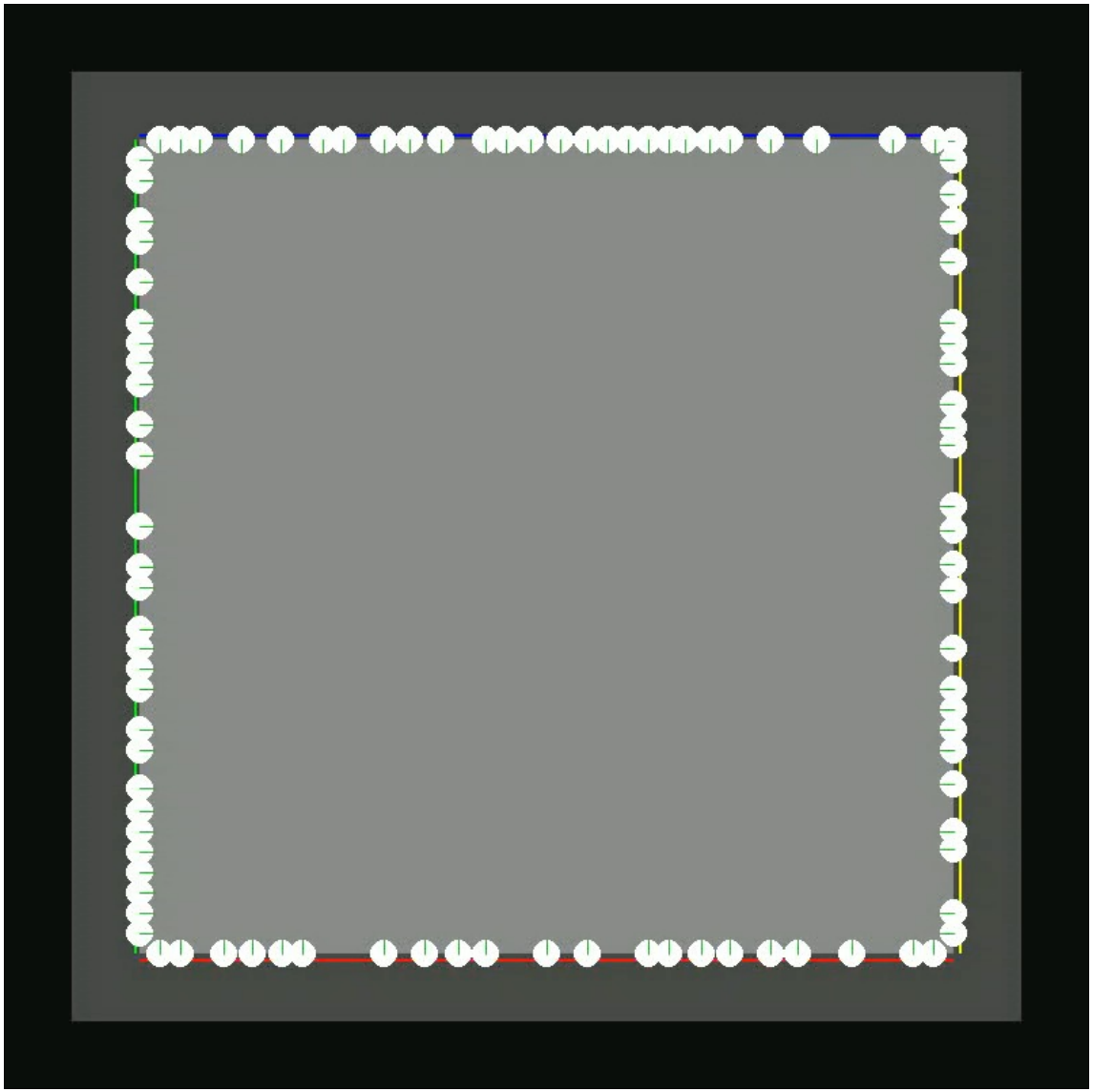
PPO using dense model



0.96 Accuracy

52.2 Steps

PPO using CNN with Attention Model



0.99 Accuracy

46.6 Steps