# Section 1. DMA-RRT

Each agent runs the individual and interaction components of Decentralized Multi-Agent Rapidly-Exploring Random Trees (DMA-RRT) in parallel. The individual component is responsible for the path planning and bidding to be the next token holder, while the interaction component is responsible for listening for messages from other agents and updating its state accordingly.

## DMA-RRT Individual Component

```
While not at the goal:
    Find a new best path using RRT
    If you have the token:
        Change your path to the new path, adjusting for conflicts with other agents
        Assign new token holder to be the agent with greatest bid
        Send your new waypoints and new token winner to all other agents
    Otherwise:
        Assign your PPI bid to how much better new path is than your current plan
        Send bid to all other agents
```

In a single iteration of planning, the agent must first interface with the provided RRT implementation to grow a tree from its current position, taking into account the other agents as dynamic obstacles.

Once the optimal path is returned by the RRT algorithm, if the agent holds the merit-based token, it must determine which agent has the most **potential path improvement (PPI)** and thus wins the right to replan its path in the next iteration. It must broadcast the winner to the rest of the team, along with the new plan it determined for itself in this iteration.

If it does not hold the merit-based token, it must instead broadcast its PPI bid for a chance to replan in the next iteration. The PPI is calculated by comparing the cost of the new optimal path the agent would take if it got to replan to the cost of the path it is currently taking.

We will give you the overall skeleton of the individual component in the notebook, but we will be asking you to fill in the helper method to compute which agent has the winning bid and should be the next token holder, as well as the helper method to calculate the bid based on an agent’s current path and best path returned by RRT.

### DMA-RRT Interaction Component

```
While agent is executing:
    Listen for messages
    If received a waypoints message:
        Update that agent’s path in your constraints
        If you are the winner:
            Set yourself to be the token holder
    If received a bid message:
        Update that agent’s bid in your list of other agent’s bids
```

While continuously running the individual process, each agent is also listening for messages from other agents in parallel, which we call the “interaction component” of DMA-RRT. We have implemented this functionality for you in a somewhat different way than the while loop above as part of the Agent and Antenna classes, but the response to each kind of message is still the same.

Whenever an agent receives a “waypoints” message (containing the updated plan of the sender as well as the ID of the PPI bid winner) it must update its current state to reflect whether or not it holds the token, as well as the other agent’s new plan. Whenever an agent receives a “bid” message (containing a PPI bid from the sender), it must update its internal state to reflect that bid. In the problem set, we have provided the broadcasting and listening functionality for you as part of the Agent API, but there are two helper functions for you to fill in, which we have set up to be called by all agents whenever they receive a waypoints message or a bid message.

## Section 2. Cooperative DMA-RRT Extension

The idea behind the Cooperative DMA-RRT extension is that the token holder should not only be given the right to replan its path in a given planning iteration, but also to edit the path of a peer agent in the interest of a lower global cost. This also helps avoid certain common deadlock scenarios, such as the narrow hallway situation presented in lecture. In the interest of time, we will not be asking you to implement the cooperative extension to the DMA-RRT algorithm. However, the pseudocode is provided here for anyone who is curious. More details beyond what is provided here and in the lecture slides can be found in the assigned reading.

## Cooperative DMA-RRT Individual Component

```
While not at the goal:
    Find a new path using RRT ignoring peer paths
    If you have the token:
        If the new path conflicts with only one other agent:
            check_emergency_stops and assign the conflicting agent the token
            Prune the path so it doesn’t conflict with any other agents
        Identify the new path’s emergency stops
        Change your path to the new path
        Assign new token holder to be the agent with greatest bid
        Send your new path and new token winner to all other agents
    Otherwise:
        Assign bid to how much better new plan is than your current plan
        Send bid to all other agents

def check_emergency_stops:
    If checkEstops:
        For all of the other agent’s possible emergency stops:
            Find where your path would have to stop
            Calculate the cost of the new paths
        Choose the stop nodes that minimize the cost
        If the other agent’s stop node changed:
            Send the other agent an estop message for the new stop node
        If your stop node changed:
            Prune your path to end at the stop node
    Else:
        Prune your path to avoid conflict
```

The individual component of the Cooperative DMA-RRT extension follows the same basic outline as with vanilla DMA-RRT, but before computing the winner of the PPI bidding contest, the token holder will check if a lower global cost can be achieved by editing the path of a peer with whom the token holder’s current best path option conflicts. One key observation is that for this to work, the token holder must first plan using RRT but ignoring all of the other agents’ paths. It must then check if any of the other agents’ paths conflict with the best path it found, and if so consider whether it can edit the pair of conflicting paths to achieve a global optimum. Ultimately, it must still move forward with a plan that takes the other agents into account as dynamic obstacles, to avoid collisions.

## Cooperative DMA-RRT Interaction Component

```
While agent is running:
    Listen for messages
    If received a waypoints message:
        Update that agent’s path in your constraints
        If you are the winner:
            Set yourself to be the token holder
    If received a bid message:
        Update that agent’s bid in your list of other agent’s bids
    If received an estop message:
        Stop at the node specified in the message
        CheckEstops = False
```

The interaction component of the cooperative extension works much the same as for vanilla DMA-RRT, except for the added “estop” message stop. In response to this type of message, if the sender is requesting the agent to make an emergency stop, it must truncate its path to stop at the specified node. This means that the sender has found a more globally optimal combination of paths that necessitates the recipient to cut its current path execution short.