MAS: Activity 9 – Agent Mobility

Alexandru Sorici

The JADE framework has built-in support for agent mobility (i.e. wrapping agent state and resuming activity on a remote container).

In this activity, you are requested to implement a **privacy enforcing** voting system. Specifically, you will implement the **Single Transferable Vote (STV)**[1] system.

The process will work as follows: there is a central election container which contains two types of agents: an `ElectionManager` agent and a `VoteCollector` agent; then there are 4 containers, each representing a voting college (region). Within each voting college container there is a `RegionRepresentative` agent, which collects and holds the votes cast by the voters of that region.

There are a total of **1000 voters**, **250** of them **in each region**. For each region there are **5 independent candidates** that compete for **3 available slots for the region**.

The election process goes as follows:

- Each `RegionRepresentative` agent simulates voting in its region by generating 250 random orderings of the preferences for its 5 candidates.
- When generation in completed, the `RegionRepresentative` requests the `ElectionManager` to send the `VoteCollector` agent to its container (region) to collect the votes.
- The `ElectionManager` can agree, in which case it requests the `VoteCollector` to go to the specified region container and collect the votes; the request to the `VoteCollector` contains the name of the container (region) and the name of the `RegionRepresentative` agent in that region.
- If the `VoteCollector` is gone to another container, the `ElectionManager` will refuse performing the request, in which case the `RegionRepresentative` must wait for a random period of time and then reinitiate the request.
- When the `VoteCollector` arrives at a region container it will initiate a request to the `RegionRepresentative` in that region to hand over the vote results.
- Immediately after the `VoteCollector` returns to the central election container, it informs the `ElectionManager` that it is back and it transmits the collected voting situation from the region where it has been.
- When the `ElectionManager` receives a voting result, it applies the decision algorithm described[2], and exemplified[3] on Wikipedia or quickly viewable in Figure 1. It then displays the results to the console.
- When the `ElectionManager` receives the results from all 4 regions, the process stops.

To implement the above process use the roadmap on page 2 .

[1] `https://en.wikipedia.org/wiki/Single_transferable_vote`

[2] `https://en.wikipedia.org/wiki/Single_transferable_vote#Finding_the_winners`

[3] `https://en.wikipedia.org/wiki/Single_transferable_vote#Example`

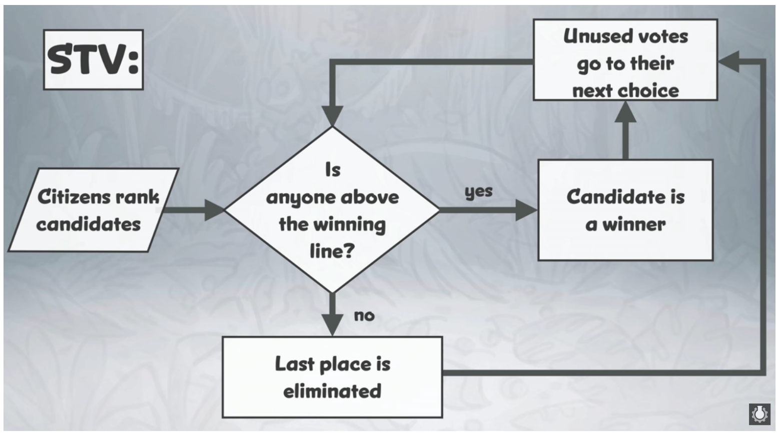

Figure 1: Quick overview of the STV voting process.

Roadmap:

- Start central election container as a main container and all the region containers as secondary ones.
- At initialization, the `VoteCollector` agent receives the AID of the `ElectionManager` and the **name** of its home container (e.g. CentralElection) as arguments.
- At initialization, each `RegionRepresentative` agent receives the following arguments:
  - the list of candidate names
  - the AID of the `ElectionManager` agent.
- Implementing communication between your agents:
  - The `RegionRepresentative` and the `ElectionManager` need to follow a **Request** protocol where the original request needs to contain the name of the region container where the `VoteCollector` is to move.
  - The `RegionRepresentative` and the `VoteCollector` use a **Request** protocol from collector to representative, where they exchange the vote result.
  - The `VoteCollector` and the `ElectionManager` use a **Request** protocol when the collector is asked to visit a region; the `inform` will be sent after the `VoteCollector` returns and the message will contain all the votes.
- Implementing mobility of the `VoteCollector` agent:
  - move it with `Agent.doMove(new ContainerID(targetContainerName, null));`
  - processing that needs to be done immediately beforeor after the movement, override the agent's `beforeMove()` and `afterMove()` methods;
  - see the Jade example in src/examples/mobile/MobileAgent.
- Sending Serializable messages between agents
  - Since the `VoteCollector` and `RegionRepresentative` agents must exchange more complex information (e.g. voting preferences) you may want to use the methods `setContentObject` and `getContentObject` in `ACLMessage` to send Java Serializable messages between agents. Make sure that the whatever you set as a content object implements the `Serializable` interface. Typical Java collections already do this.
  - in order to support mobility, all the fields in the `VoteCollector` agent must be `Serializable`.

[3] https://www.youtube.com/watch?v=l8XOZJkozfI