MAS: Activity 9 – Agent Mobility

Alexandru Sorici
22.04.2019

The **Java Agent DEvelopment Framework (Environment) – or JADE**[1] 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)**[2] 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 read the result of a vote in its region for the 5 candidates (from the provided JSON file).
- The `RegionRepresentative` asks the `ElectionManager` to *send* the `VoteCollector` agent to its container (region) to collect.
- The `ElectionManager` can respond affirmatively, in which case it sends a message to the `VoteCollector` to go to the specified region container and collect the votes; the `ElectionManager` will transmit to the `VoteCollector` the name of the container (region) and the name of the `RegionRepresentative` in that region. If the `VoteCollector` is gone to another container, the `ElectionManager` will deny 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 send 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[3], and exemplified[4] on Wikipedia or quickly viewable in Figure 1. It then displays the results to 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] http://en.wikipedia.org/wiki/Java_Agent_Development_Framework
[2] https://en.wikipedia.org/wiki/Single_transferable_vote
[3] https://en.wikipedia.org/wiki/Single_transferable_vote#Finding_the_winners
[4] https://en.wikipedia.org/wiki/Single_transferable_vote#Example

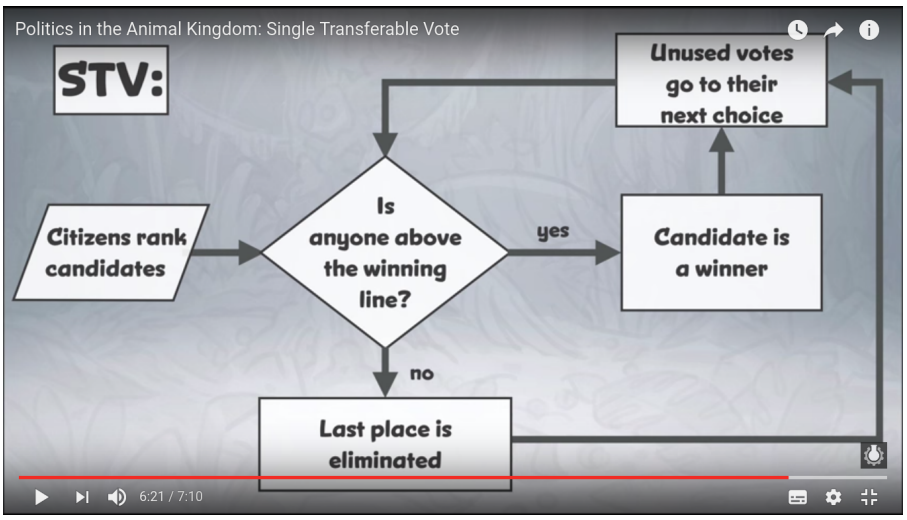

Figure 1: Quick overview of the STV voting process.

Roadmap:

- Start *central election* container as a *main* container and all the *region* containers as secondary ones.
- At initialization, each `RegionRepresentative` subscribes for the central election manager to get its AID. It also reads in the result of votes in its region, which it has to send over to the `ElectionManager`.

- 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 and an **Inform** notification when the collector resturns.

- 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*.

- Getting information about current container in which an agent is running: use the `getContainerController()` method in the agent class

- 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 `setContentObject` and `getContentObject` methods 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.

- **Work in teams** of 2 or 3: divide amongst the responsibility for implementing the `VoteCollector`, the `RegionRepresentatives`, and the `ElectionManager`.