Sensor network simulation in YAES

February 18, 2008

This is a following to the original quick YAES tutorial, and it shows the functionality for sensor network simulation supported in YAES. It utilizes the EEL6788 package as an example, which contain a (partial) implementation of the DirectedDiffusion protocol.

The code supporting sensor networks in YAES is in the `yaes.framework.world.sensornetwork` package. Most of these classes can be overwritten by you.

The general picture

A sensor network contains "entities", which can be sensors, sinks, mobile sinks, intruders and so on. These objects are represented as having two parts:

- the physical object itself (class `SensorNode` , `SinkNode`, `ActuatorNode`). Most of the time, you will not need to overwrite these objects.
- the agent which controls the node. This needs to implement `IAgent` (which is a very simple interface having an single `action()` function). If you want the agent to participate in communication, you should implement the `ICommunicationAgent` interface.

In general you will have code structure in the form of Figure 1.

If the intruder is very dumb and if only follows a fixed movement pattern, you don't need to give it an agent, you can simply set its path.

As you might guess, the agent is the key to the behavior of the sensor, and you will spend most of your time writing the agent. The agent needs to do:

- sensing
- communication
- movement (if it is mobile)

In order to start your agent with the most that YAES can offer in this direction, you should inherit your agent from `yaes.framework.world.sensornetwork.AbstractSensorAgent`. your node should inherit from `yaes.framework.world.sensornetwork.SensorNode`.

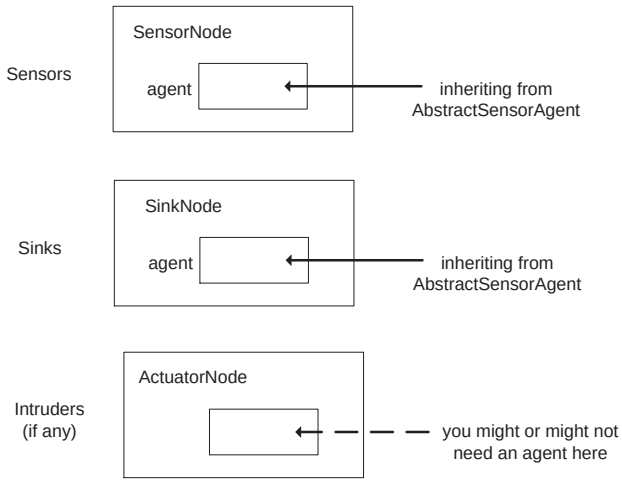

Figure 1: code structure you need to implement.

sensor network world

The `yaes.framework.world.sensornetwork.SensorNetworkWorld` class manages the communication and sensing between the sensor nodes.

This is how it works:

- every time an agent transimits a message through its `transimit()` function call, it is recorded by the `SensorNetworkWorld`.
- every time an intruder moves (Actuator), it should call the `move(ActuatorNode actuatorNode)` function on the `SensorNetworkWorld`. This will record the movement which will be delivered to the sensors as an obsenvation.

All these movement, message, sensing are delivered when you call the `messageFlush()` function in the `SensorNetworkWorld`.

You should do this as the last step of your `update` function in your `simulicationCode`.

What will happen at `SensorNetworkWorld.messageFlush()`

- Messages sent from one sensor node to another will be delivered to the mailbox of the sensor node as a `PerceptionType.ReceivedMessage` if it is in the transimission range of the source node. If the node is outside the transmission range, it does not receive anything.
- Messages sent to a broadcast target ("*") are delivered to the mailbox of nodes in the transmission range as a RECEIVED_MESSAGE.

- Messages sent to a specified destination are delivered to the mailbox of the nodes which are in the transmission range but are not the destination as `PerceptionType.Overhearing`.
- Movement by the actuator are delivered to the mailbox of the sensors which contain the actuator in their sensing range as a `Perception.Observation`

Operation of a sensor node

As we said, the operation of the sensor node takes place in `action()` function of the agent attached to the node. In broad lines, this is what the agent is supposed to do:

- Check its mailbox, which contains received messages, overhearings and observations. Process these according to the rules of the communication protocol etc.

```
SensingHistory sensingHistory = getSensorWorld().getSensingHistory(getNode());
List<Perception> perceptions = sensingHistory.getMailbox();
  // note: this will empty the mailbox
for(Perception p: perceptions){
  // process each perception p
}
```

- Send some messages, if needed.

```
ACLMessage m = new ACLMessage(getNode().getName(), ACLMessage.Performative.INFORM);
m.setDestination("Destination");
m.setValue(...);
transmit(message);
```

Note that the destination will receive the message not right away, but in the next timestep. Also, note that nodes in addition to the destination will also receive the message (as an Overhearing).

If you have a mobile node you also need to perform whatever movement you want here. The simplest is to just get the node attached to the agent, and set its location to the new position:

```
getNode().setLocation(new Location(30, 30));
```