JBotSim

Agenda

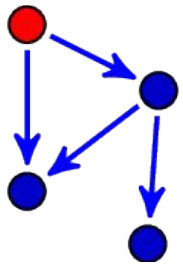
- Create an algorithm
- Flooding algorithm
- Create your topology
- Other examples

Reference

- JBotSim Documentation
- JBotSim Youtube channel

JBotSim

- A simulation library for distributed algorithms in dynamic networks
- It fosters an event-driven programming style: nodes react to various events
- Node movements can be controlled by programmatically or by the user during the simulation
- Home page



https://jbotsim.io/

Using JBotSim with Maven (dependency)

```
<dependency>
 <groupId>io.jbotsim</groupId>
<artifactId>jbotsim-all</artifactId>
 <version>1.1.1</version>
</dependency>
```

To add in the pom.xml

A Empty Simulation

```
import io.jbotsim.core.Topology;
import io.jbotsim.ui.JViewer;

public class HelloWorld{
   public static void main(String[] args){
      Topology tp = new Topology();
      new JViewer(tp);
      tp.start();
   }
}
```

Comments

- Topology manages the nodes and links, and organizes the inner life of the system (timing, messaging, etc.)

- JViewer displays the simulation elements and allow the user to interact with it

Demo

Implementing a distributed algorithm

- To create a distributed algorithm we need to define the behaviour of each node
- Three steps are required
  a. Create a class that extends Node
  b. Write the algorithm through overriding methods
  c. Register your type of node in the topology

A skeleton of algorithm

```
...
public class EmptyNode extends Node{
   @Override
   public void onStart() {
      // JBotSim executes this method on each node upon initialization
   }
...
```

A skeleton of algorithm

```
...
public class EmptyNode extends Node{
...
   @Override
   public void onSelection() {
      // JBotSim executes this method on a selected node
   }
...
```

A skeleton of algorithm

```
...
public class EmptyNode extends Node{
   ...
   @Override
   public void onClock() {
      // JBotSim executes this method on each node in each round
   }
...
```

A skeleton of algorithm

```
...
public class EmptyNode extends Node{
...
    @Override
    public void onMessage(Message message) {
        // JBotSim executes this method on a node every time it receives a message
    }
}
```

Using the class node

```
public class Main{

   public static void main(String[] args){

      Topology tp = new Topology();

      tp.setDefaultNodeModel(EmptyNode.class);

      new JViewer(tp);

      tp.start();

   }

}
```

Example: flooding algorithm 1

In the example

- The inform variable stores the state of the node true when the node had received the message
- The sendAll method sends a message to all the out neighbors
  - It is also possible to send a message to a single neighbor using send()
- The initiator is the node that is selected by the user

Example: flooding algorithm 2

In the example

- The simulation is performed in rounds, discrete unit of time
- At each round the onClock() method is called
- The arrival of a message causes onMessage() to be called on the receiving node
  - A node may possibly receive several messages in the same round, causing as many invocations of onMessage()
- The content of a message is any object: it is possible to assign a flag to a message when different kinds of messages are used

Other useful methods of the Node class

- List<Node> getInNeighbors() returns a list containing every node serving as source for an adjacent directed links
- List<Node> getOutNeighbors() returns a list containing every node serving as destination for an adjacent directed link
- void setID(int ID) sets the identifier of this node. Nodes have an identifier by default, which is the smallest available integer.
- int getID() returns the identifier of this node.
- void setLabel(Object label) sets the label of this node. Default GUI shows it as tooltip when the mouse cursor is held some time over the node
- Object getLabel() returns the label of this node

Many other methods are available, see the documentation

Building a specific topology

The class TopologyGenerators provides a set of static methods to create specific graphs

- generateGrid(Topology topology, int nbNodesRow, int nbNodesColumn) generates a grid with the specified number of rows and columns
- generateLine(Topology topology, int nbNodes) generates a horizontal line of nodes
- generateRing(Topology topology, int nbNodes) generates a ring

Building a specific topology

You can code the topology you want by using the methods of the class Topology

- addNode(double x, double y, Node n) adds a node n to the specific location
- addLink(Link l) adds a link between two nodes
- clear() removes all nodes and link from the topology

Example: building a random graph

Other Examples

- Random Walk
- Moving nodes
- Dying nodes
- Changing Icons
- Mobile Broadcast
- Links
- Cowboy

Conclusions

- Create an algorithm
- Flooding algorithm
- Create your topology
- Examples

Reference

- JBotSim Documentation
- JBotSim Youtube channel