JBotSim: A Tool for Fast Prototyping of Distributed Algorithm in Dynamic Networks

Arnaud Casteigts

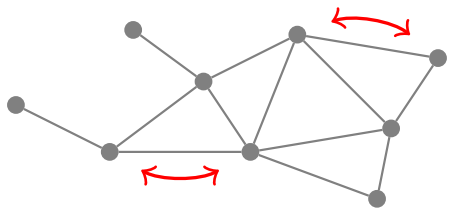
University of Bordeaux

SIMUTools 2015

Distributed Computing (for information)

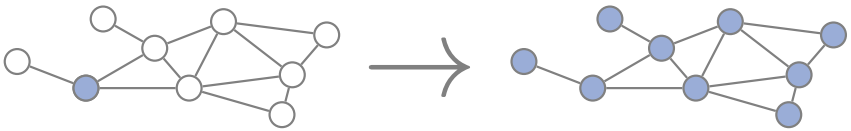
Collaboration of entities to perform a common task, w/o centralization.

(Think globally, act locally)

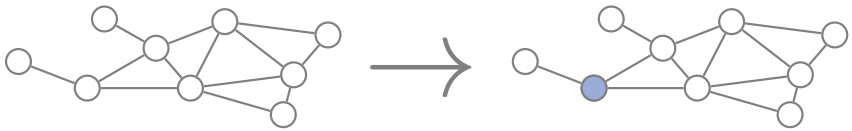


Examples:

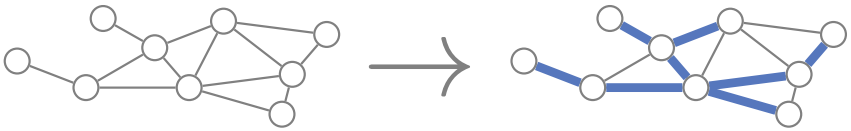
Broadcast



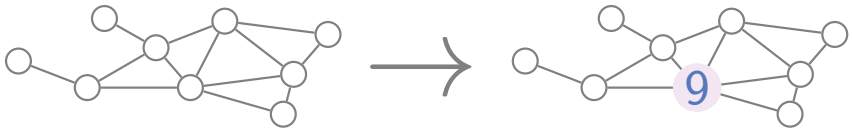
Election



Spanning tree



Counting



+ consensus, naming, routing, exploration, *etc.*

(my focus: highly-dynamic networks).

Distinctive features

Library: JBotSim is not an executable software. It is a library (JAR) to be used in your programs.

Interactivity: Allows you to interact with an algorithm during its execution (*e.g.* add, move, or delete nodes and links).

High-level: Allows you to test algorithmic ideas quickly. High-level API (send message, move, etc.)

Batch mode: Can be run with or without GUI.

Simplicity: Beginner friendly, online tutorials.

Modular: Can be extended or embedded in other software.

Main method

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

The default type of nodes is class `Node`. Objects of this type are created whenever a new node is added to the topology.

By default, the topology is wireless (this can be changed), meaning that links are created automatically depending on the distance between nodes.

The algorithm (extending class Node)

```
public class MBCNode extends Node{
  boolean informed = false;

  public void onStart(){
    setDirection(2*Math.PI * Math.random());
    setColor(null);
  }

  public void onSelection(){
    informed = true;
    setColor(Color.red);
  }

  public void onMessage(Message message){
    informed = true;
    setColor(Color.red);
  }

  public void onClock(){
    move();
    wrapLocation();
    if (informed)
      sendAll(new Message());
  }
}
```

The algorithm (extending class Node)

```
public class MBCNode extends Node{
  boolean informed = false;

  public void onStart(){
    setDirection(2*Math.PI * Math.random());
    setColor(null);
  }

  public void onSelection(){
    informed = true;
    setColor(Color.red);
  }

  public void onMessage(Message message){
    informed = true;
    setColor(Color.red);
  }

  public void onClock(){
    move();
    wrapLocation();
    if (informed)
      sendAll(new Message());
  }
}
```

Main method

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

Event-based programming:

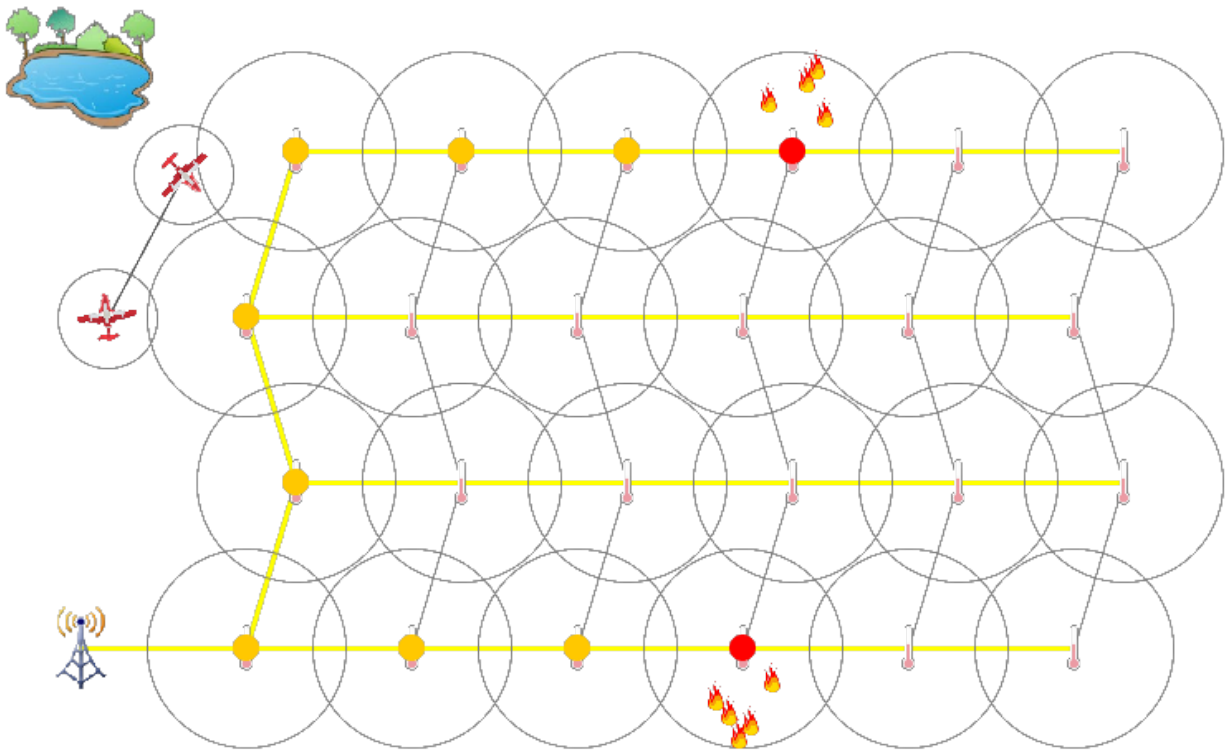
Typical way of coding (distributed algorithms):
→ extend class `Node` and override methods:

`onMessage()`	a message is received
`onLinkAdded()`	a new link appears
`onLinkRemoved()`	a link disappears
`onSensingIn()`	something is sensed
`onSensingOut()`	something gets out of range
`onMove()`	this node has moved (or *was* moved)
`onSelection()`	this node is selected by the user
`onClock()`	once in every round
...	

Note: These events are also available through interfaces (if not in class Node), e.g. using `MessageListener`, `ClockListener`, *etc.*

Example
Complex scenario

A wireless sensor network deployed over an area for fire watching. When a fire is detected, the base station is informed through recursive notification. Canadairs are sent to extinguish it.



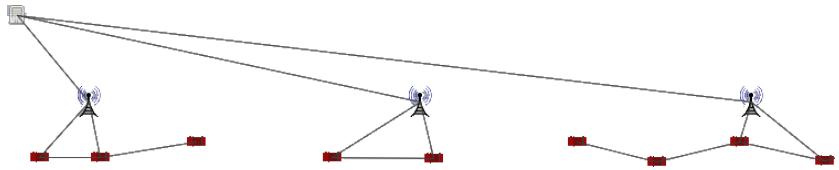
Five types of `Node` are involved:

`Sensor`, `Station`, `Canadair`, `Lake`, and `Fire`.

Versatility

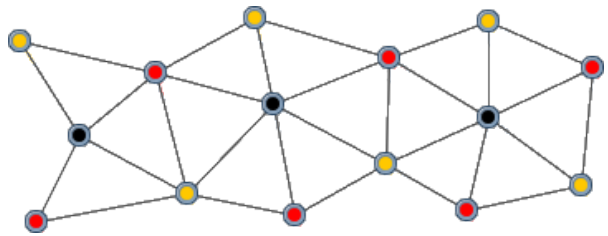
Communication:

- wired or wireless (or mixed)
- directed or undirected (or mixed)



Paradigms:

- distributed or centralized
- message-based or graph-based
- synchronous or asynchronous



Dedicated models:

Compatible with most of the usual models, e.g. $\mathcal{LOCAL}$, $\mathcal{ASYNC}$, *Look compute move*, *Population protocols*, *Mobile agents*, *etc.*

Extensions

Modular, most components can be replaced or extended

Ex: Wireless link resolver, message engine ($\mathcal{LOCAL}$, $\mathcal{ASYNC}$), Viewer (local or remote (Carlos Gomez))

Dynamic graphs

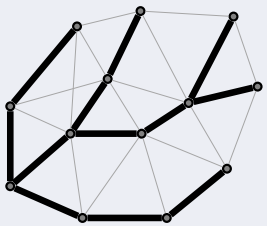
- Edge-markovian generator
- TVG from file
- Trace recorder & player

Mobility models

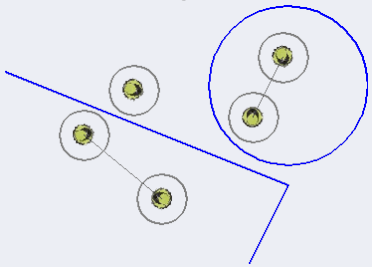
- Random/sequential Way Point
- Aerial models (V. Autefage)
- David Renault's meta walk :)

Miscellaneous

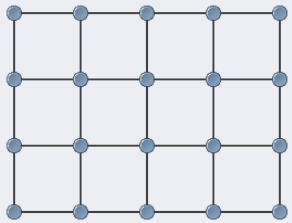
TikZ export



Obstacles (M. Barjon)



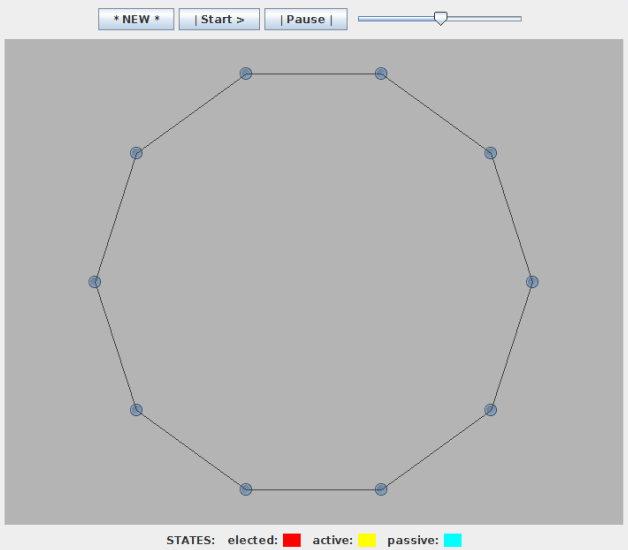
Topology generation



...

JBotSim as a component in other applications

Embed it in your application

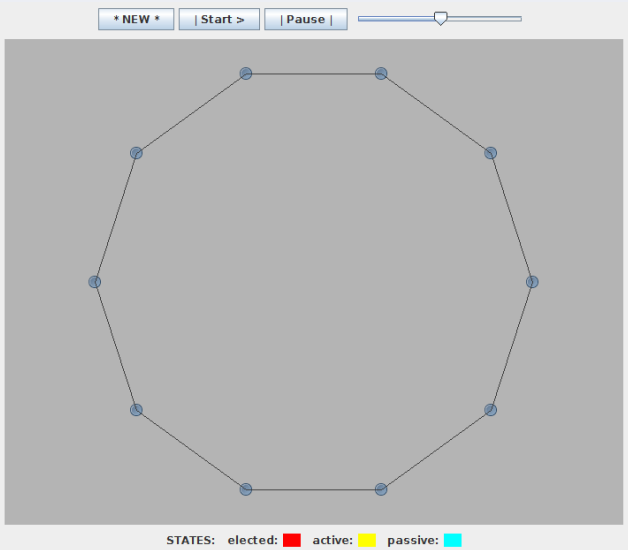


Emmanuel Godard

Société Mugen (via A. Casteler, L3)

JBotSim as a component in other applications

Embed it in your application



Emmanuel Godard

Société Mugen (via A. Casteler, L3)

Learn more about it?

http://jbotsim.sourceforge.net

→ `Download, browse code samples and documentation`

Youtube channel: JBotSim

Paper:

A. Casteigts, "JBotSim: a Tool for Fast Prototyping of Distributed Algorithms in Dynamic Networks," in Proc. of *SIMUTools*, 2015.

→ Full (and up to date) version on arXiv (1001.1435).