ANET - Lab 1

M. Potop-Butucaru, F. Petit, S. Tixeuil

Sinalgo Programming

Exercise 1

a. Download Sinalgo «Regular Release» (http://dcg.ethz.ch/projects/sinalgo/) on the machine you will be using. Make sure that Java is also properly installed on your machine.

b. Install and setup the Sinalgo simulator by following http://disco.ethz.ch/projects/sinalgo/tutorial/Installation.html

c. Execute the Sinalgo examples (http://disco.ethz.ch/projects/sinalgo/tutorial/Execution.html) and play with the simulation environment.

d. Read the «Projects, Node Implementation, Model Implementation, and Configuration» sections of the Sinalgo tutorial. By now you should understand what are the components involved in the simulator.

Exercise 2

We want to implement in the simulator a coloring protocol. The protocol works as follows :
1. at the begining each node randomly draws a color in a set of *k* possible collors ;
2. each node then broadcasts its color to its neighbors ;
3. when a node receives a neighbor's color that is equal to its own colors, it randomly selects a new one (among *k*).

The first step is to get a folder with every necessary file, copy the folder «template» in the subfolder «src/projects» into a new folder «coloring» in the same «src/projects» folder.

The second step is to provide implementation for new types of nodes, messages, and timers. The code for the message is to be placed in the file `CMessage.java` in the «src/projects/coloring/nodes/messages» folder. The contents of the `CMessage.java` is as follows :

```
package projects.coloring.nodes.messages;

import sinalgo.nodes.messages.Message;

public class CMessage extends Message {

      public int id;
      public int color;

      public CMessage(int id, int color) {
            this.id=id;
            this.color = color;
```

```
	}

	public Message clone() {
		return new CMessage(id,color);
	}

}
```

The code for the timer is to be placed in the file `CTimer.java` in the «src/projects/coloring/nodes/timers» folder. The contents of the `CTimer.java` is as follows :

```
package projects.coloring.nodes.timers;

import projects.coloring.nodes.nodeImplementations.CNode;
import projects.coloring.nodes.messages.*;
import sinalgo.nodes.timers.Timer;

public class CTimer extends Timer {
	CNode sender;
	int interval;

	public CTimer(CNode sender, int interval) {
		this.sender = sender;
		this.interval = interval;
	}

	// called upon timer expiration
	public void fire() {
		// create message with color
		CMessage msg= new CMessage(sender.ID,sender.getColor());
		sender.broadcast(msg); // send to all neighbors
		this.startRelative(interval, node);
		// recursive restart of the timer
	}
}
```

The code for the node is to be placed in the file `CNode.java` in the «src/projects/coloring/nodes/nodeImplementations» folder. The contents of the `CNode.java` is as follows :

```
package projects.coloring.nodes.nodeImplementations;

import java.awt.Color;
import java.awt.Graphics;
import java.util.*;

import sinalgo.configuration.WrongConfigurationException;
import sinalgo.gui.transformation.PositionTransformation;
import sinalgo.nodes.Node;
import sinalgo.nodes.edges.Edge;
import sinalgo.nodes.messages.Inbox;
import projects.coloring.nodes.timers.*;
import projects.coloring.nodes.messages.*;
import sinalgo.nodes.messages.Message;

class state
{
	int color;

	state(int color){
```

```
            this.color=color;
      }
}

public class CNode extends Node {

      private int color;
      private final int nb = 10;
      private            final            Color            tab[]            =
{Color.BLUE,Color.CYAN,Color.GREEN,Color.LIGHT_GRAY,Color.MAGENTA,Color.ORANGE,C
olor.PINK,Color.RED,Color.WHITE,Color.YELLOW};

      private Hashtable<Integer,state> neighborStates;

      public int getColor(){
            return color;
      }

      public Color RGBCouleur(){
            return tab[getColor()];
      }

      public void setColor(int c) {
            this.color=c;
      }

      public void initColor(int range){
            setColor((int) (Math.random() * range) % range);
      }

      public void compute(){
            boolean same=false;
            Iterator<Edge> it=this.outgoingConnections.iterator();
            boolean SC[]=new boolean[nb];

            for (int i=0;i<SC.length;i++)
                  SC[i]=false;

            while(it.hasNext()){
                  Edge e=it.next();
                  state tmp=neighborStates.get(new Integer(e.endNode.ID));
                  if(tmp!=null){
                        if(tmp.color==this.getColor()){
                              same=true;
                        }
                  SC[tmp.color]=true;
                  }
            }

            if (same){
                  int dispo=0;
                  for (int i=0;i<SC.length;i++)
                        if(SC[i]==false) dispo++;
                  if (dispo == 0) return;

                  int choix= ((int) (Math.random() * 10000)) % dispo + 1;
                  int i=0;


                  while(choix > 0){
```

```
                  if(SC[i]==false)
                        choix--;
                  if(choix>0) i++;
            }
            this.setCouleur(i);
      }
}

public void handleMessages(Inbox inbox) {

      if(inbox.hasNext()==false) return;

      while(inbox.hasNext()){

            Message msg=inbox.next();

            if(msg instanceof CMessage){
            state tmp=new state(((CMessage) msg).color);
      neighborStates.put(new Integer(((CMessage) msg).id),tmp);
      compute();
            }
      }

}

public void preStep() {}

public void init() {
      initColor(nb);
      (new CTimer(this,50)).startRelative(50,this);
      this.neighborStates    =    newHashtable<    Integer,    state    >
                  ( this.outgoingConnections.size() );

}

public void neighborhoodChange() {}

public void postStep() {}

public String toString() {
      String s = "Node(" + this.ID + ") [";
      Iterator<Edge> edgeIter = this.outgoingConnections.iterator();
      while(edgeIter.hasNext()){
            Edge e = edgeIter.next();
            Node n = e.endNode;
            s+=n.ID+" ";
      }
      return s + "]";
}

public void checkRequirements() throws WrongConfigurationException {}

public void draw(Graphics g,PositionTransformation pt,boolean highlight) {
      Color c;
      this.setColor(this.RGBColor());

      String text = ""+this.ID;
      c=Color.BLACK;

      super.drawNodeAsDiskWithText(g, pt, highlight, text, 20, c);
```

```
        }
}
```

The last step is to define the simulation model based on existing models. Open the `Config.xml` file in the «`projects/coloring`» folder and modify the following parameters :

1. forbid mobility and interferences,
2. change the type of nodes from `DummyNode` to `coloring:CNode`,
3. change the edge type to bidirectional (`sinalgo.nodes.edges.BidirectionalEdge`),
4. initialize connexions from the begining of the simulation,
5. change the transmission mode from synchronous to asynchronous,
6. to change network connectivity, change `rMax` parameter in `GeometricNodeCollection` and `UDG`.

Test and practise the simulation of the coloring protocol.

Exercise 3

Implement in Sinalgo a distance-2 coloring protocol. You may want to reuse the distance-1 coloring seen in Exercise 2. Here a node is satisfied if it doesn't have any two neighbors using the same color. If it is not the case, at least one of those two nodes should take a new color.

Exercise 4

Implement in Sinalgo the clustering protocol seen in ANET Class 2. You may reuse the distance-1 coloring protocol in your implementation.

The sketch of the protocol (in French) can be found at:
http://www-sop.inria.fr/mascotte/Algotel2005/Actes/09.pdf

The details of the protocol can be found at:
http://www.researchgate.net/profile/Sebastien_Tixeuil/publication/230846444_Self-stabilization_in_self-organized_multi-hop_wireless_Networks/links/0c96052b37bc3b4bad000000.pdf

Exercise 5

Implement in Sinalgo the TDMA protocol seen in ANET Class 2.

The sketch of the protocol (in French) can be found at:
algotel2004.conf.citi.insa-lyon.fr/articles/tdmaalgotel.pdf

The details of the protocol can be found at:
http://www.researchgate.net/profile/Sebastien_Tixeuil/publication/220483222_A_Distributed_TDMA_Slot_Assignment_Algorithm_for_Wireless_Sensor_Networks/links/09e4150e4236c696a8000000.pdf