COP5612 – Fall 2013
Project 2 – Gossip Simulator

Alin Dobra

September 18, 2013

- **Due Date:** October 5, Midnight
- One submission per group
- Submit using eLearning
- **What to include:**
  - `README` file including group members, other requirements specified below
  - `project2.scala` the code for the project
  - `project2-bonus.scala` the code for the bonus part, if any

1 Problem definition

As described in class Gossip type algorithms can be used both for group communication and for aggregate computation. The goal of this project is to determine the convergence of such algorithms through a simulator based on actors written in Scala. Since actors in Scala are fully asynchronous, the particular type of Gossip implemented is the so called *Asynchronous Gossip.*

Gossip Algorithm for information propagation The Gossip algorithm involves the following:

- **Starting:** A participant(actor) it told/sent a roumor(fact) by the main process
- **Step:** Each actor selects a random *neighboor* and tells it the roumor
- **Termination:** Each actor keeps track of rumors and how many times it has heard the rumor. It stops transmitting once it has heard the roumor 10 times (10 is arbitrary, you can play with other numbers).

Push-Sum algorithm for sum computation

- **State:** Each actor A_i maintains two quantities: s and w. Initially, $s = x_i = i$ (that is actor number i has value i, play with other distribution if you so desire) and $w = 1$
- **Starting:** Ask one of the actors to start from the main process.
- **Receive:** Messages sent and received are pairs of the form (s, w). Upon receive, an actor should add received pair to its own corresponding values. Upon receive, each actor selects a random neighboor and sends it a message.
- **Send:** When sending a message to another actor, half of s and w is kept by the sending actor and half is placed in the message.
- **Sum estimate:** At any given moment of time, the sum estimate is $\frac{s}{w}$ where s and w are the current values of an actor.
- **Termination:** If an actors ratio $\frac{s}{w}$ did not change more than 10^{-10} in 3 consecutive rounds the actor terminates. **WARNING: the values s and w independently never converge, only the ratio does.**

Topologies The actual network topology plays a critical role in the dissemination speed of Gossip protocols. As part of this project you have to experiment with various topologies. The topology determines who is considered a neighboor in the above algorithms.

- **Full Network** Every actor is a neighboor of all other actors. That is, every actor can talk directly to any other actor.
- **2D Grid:** Actors form a 2D grid. The actors can only talk to the grid neigboors.
- **Line:** Actors are arranged in a line. Each actor has only 2 neighboors (one left and one right, unless you are the first or last actor).
- **Imperfect 2D Grid:** Grid arrangement but one random other neighboor is selected from the list of all actors (4+1 neighboors).

2 Requirements

Input: The input provided (as command line to your `project2.scala`) will be of the form:

```
project2.scala numNodes topology algorithm
```

Where `numNodes` is the number of actors involved (for 2D based topologies you can round up until you get a square), `topology` is one of `full`, `2D`, `line`, `imp2D`, `algorithm` is one of `gossip`, `push-sum`.

Output: Print the amount of time it took to achieve convergence of the algorithm. Please measure the time using

```
... build topology
val b = System.currentTimeMillis;
..... start protocol
println(b-System.currentTimeMillis)
```

Actor modeling: In this project you have to use exclusively the actor facility in Scala (**projects that do not use multiple actors or use any other form of parallelism will receive no credit**).

README file In the README file you have to include the following material:

- Team members
- What is working
- What is the largest network you managed to deal with for each type of topology and algorithm

Report.pdf For each type of topology and algorithm, draw the dependency of convergence time as a function of the size of the network. You can overlap different topologies on the same graph, i.e. you can draw 4 curves, one for each topology and produce only 2 graphs for the two algorithms. Write about any interesting finding of your experiments in the report as well and mention the team members.

You can produce Report.pdf in any way you like, for example using spreadsheet software. You might have to use logarithmic scales to have a meaningful plot.

3 BONUS

In the above assignment, there is no failure at all. For a 30% bonus, implement node and failure models (a node dies, a connection dies temporarily or permanently). Write a Report-bonus.pdf to explain your findings (how you tested, what experiments you performed, what you observed) and submit `project2-bonus.scala` with your code. To get the bonus you must implement at least one failure model controlled by a parameter and draw plots that involve the parameter. At least one interesting observation has to be made based on these plots.