COP5615 – Fall 2019
Project 2 – Gossip Simulator
Alin Dobra
September 10, 2019

- **Due Date**: 1st October (till 11:59pm)
- One Submission Per Group
- Submit using eLearning
- **What to Include**:
    - README file including group members
    - Lastnames.zip: the code for the project
    - Lastnames-bonus.zip: the code for the project

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 Elixir. Since actors in Elixir 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 rumor(fact) by the main process
- **Step**: Each actor selects a random neighbor and tells it the rumor
- **Termination**: Each actor keeps track of rumors and how many times it has heard the rumor. It stops transmitting once it has heard the rumor 10 times (10 is arbitrary, you can play with other numbers or other stopping criteria).

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 neighbor 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 s/w where s and w are the current values of an actor.
- **Termination:** If an actor ratio 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 neighbor in the above algorithms.

- **Full Network:** Every actor is a neighbor of all other actors. That is, every actor can talk directly to any other actor.
- **Line:** Actors are arranged in a line. Each actor has only 2 neighbors (one left and one right, unless you are the first or last actor).
- **Random 2D Grid:** Actors are randomly position at x, y coordinates on a [0-1.0] x [0-1.0] square. Two actors are connected if they are within .1 distance to other actors.
- **3D torus Grid:** Actors form a 3D grid. The actors can only talk to the grid neighbors. And, the actors on outer surface are connected to other actors on opposite side, such that degree of each actor is 6.
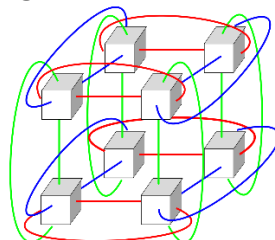
- **Honeycomb:** Actors are arranged in form of hexagons. Two actors are connected if they are connected to each other. Each actor has maximum degree 3.
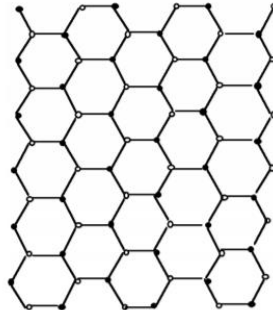
- **Honeycomb with a random neighbor:** Actors are arranged in form of hexagons (Similar to Honeycomb). The only difference is that every node has one extra connection to a random node in the entire network.

2 Requirements

Input: The input provided (as command line to your program will be of the form:

my_program numNodes topology algorithm

Where numNodes is the number of actors involved (for 3D based topologies you can round up until you get a cube, similarly for 2D until you get a square), topology is one of **full, line, rand2D, 3Dtorus, honeycomb and randhoneycomb,** algorithm is one of **gossip, push-sum.**

Output: Print the amount of time it took to achieve convergence of the algorithm. Please described how you measured the time in your report.

Actor modeling: In this project you have to use exclusively the actor facility (GenServer) in Elixir (**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 20% 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.zip 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.