Project 2: Gossip Protocol

Distributed Operating Systems Principles - Fall 2025

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 Gleam. Since actors in Gleam are fully asynchronous, the particular type of Gossip implemented is the so-called Asynchronous Gossip.

2 Gossip Algorithm for Information Propagation

The Gossip algorithm involves the following:

- **Starting**: A participant (actor) is 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 select other values).

3 Push-Sum Algorithm for Sum Computation

- **State**: Each actor A_i maintains two quantities: s and w. Initially, $s = x_i = i$ (that is, A_i has value i, you can experiment with other distributions if desired) and $w = 1$.
- **Starting**: One of the actors starts upon request from the main process.
- **Receive**: Messages sent and received are pairs of the form (s, w). Upon receiving, an actor adds the received pair to its own corresponding values. After receiving, 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, the sum estimate is $\frac{s}{w}$, where s and w are the current values of the actor.
- **Termination**: If an actor's ratio $\frac{s}{w}$ does not change by more than 10^{-10} over three consecutive rounds, the actor terminates. **Warning:** The values of s and w never converge independently—only the ratio does.

4 Topologies

The network topology plays a critical role in the speed of information dissemination in Gossip protocols. In this project, you are required to experiment with various topologies. The topology determines who is considered a neighbor in the algorithms.

- **Full Network**: Every actor is a neighbor of every other actor. That is, every actor can communicate directly with any other actor.
- **3D Grid**: Actors form a 3D grid. Actors can only communicate with their grid neighbors.
- **Line**: Actors are arranged in a line. Each actor has two neighbors (one on the left and one on the right, except for the first and last actors).
- **Imperfect 3D Grid**: A grid arrangement where each actor also has one additional randomly selected neighbor from the set of all actors ($6 + 1$ $neighbors$).

5 Requirements

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

```
project2 numNodes topology algorithm
```

Where:

- `numNodes` is the number of actors involved (for 2D-based topologies, you can round up to the nearest square).
- `topology` is one of: full, 3D, line, or imp3D.
- `algorithm` is one of: gossip, push-sum.

Output: Print the amount of time it took to achieve convergence for the algorithm. Measure the time using:

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

Actor Modeling: In this project, you must use only the actor model in Gleam. Projects that do not use multiple actors or use other forms of parallelism will not receive credit.

README file: In the README file, you must include the following:

- Team members
- What is working
- The largest network size you managed for each type of topology and algorithm.

Report.pdf: For each type of topology and algorithm, plot the dependency of convergence time as a function of network size. You can overlap different topologies on the same graph (i.e., draw 4 curves—one for each topology—and produce only two graphs for the two algorithms). Write about any interesting findings from your experiments in the report and list your team members.

You can generate **Report.pdf** in any way you like, such as using spreadsheet software. You may need to use logarithmic scales for meaningful plots.

6 Bonus

In the above assignment, there is no failure model. 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, and what you observed) and submit `project2-bonus.tgz/zip` with your code. To get the bonus, you must implement at least one failure model controlled by a parameter and draw plots involving that parameter. Make at least one interesting observation based on these plots.