# DOSP Project 2

Yu, Mingjun (UFID:61707843)

Sun, Hui (UFID:66542614)

# 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 F#. Since actors in F# 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 select other values).

**Push-Sum algorithm for sum computation**

- State: Each actor Ai maintains two quantities: s and w. Initially, s = xi = 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 actors 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.
- 2D Grid: Actors form a 2D grid. The actors can only talk to the grid neighbors.
- 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).
- Imperfect 2D Grid: Grid arrangement but one random other neighbor is selected from the list of all actors (4+1 neighbors).

# 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 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.

# Results Excepted

From discussion in class, whether it be Gossip or Push-Sum, the time taken to converge from less to high should be full < imperfect 2D < 2D < line.

# Implementation Details.

We have 5 files in total. They are 'Message.fs', 'Boss.fs', 'Worker.fs', 'Topology.fs', 'Program.fs'.
**Message.fs**
In this file, we define two types. The PushSumMsg and MyMessage. This will help us write code in a tidy way.

**Program.fs**
The main entry is in this file. The 'getAlgoGossipCode' help us to deal with the input from command. And the 'getRevisedActor' function help us to return a number's square so that it could be used in the 2D and imperfect 2D situations.

**Boss.fs**
In this file, we define a 'BossActor' which used for helping us starting the program and receive some important messages from workers.
In this file, we define some static mutable variables so that the workers could call those variables. We also define 'changeValue' to help us find the corresponding method. For example, if we receive "full gossip", we could find the corresponding topology and algorithm. 'BossActor' would match the received message with a 'string' and 'int'. If the received message is 'string' type. It would print this string. If the received message is a type of 'int', it means that 'BossActor' received a message from workers.

**Workers.fs**

'Worker' should send message to 'BossActor' and 'Topology' so that the 'BossActor' could compute the converge time and the Topology could run the code properly.

**Topology.fs**

This file is the foundation of our code, because in this file, we define different kinds of function to implement different situations, including four topology, full, imperfect 2D, 2D and line. In all of these methods, we deal with the situations of 'gossip' and 'push-sum'. And the 'getRandomStartNodes' help us find the start nodes which we need. Also, the 'spreadRumor' help us to transfer rumor as required.

# How to Run

We use the VS to compile our code, and after clicking run button of visual studio, we would get a command window from the compiler. Enter parameters in this form:

50 full gossip 5

The first parameter is the number of nodes. The second parameter is the type of topologies. The third parameter is gossip or push-sum. The fourth parameter is the number of nodes which know the rumor at the beginning.

# Results

Below are the screenshots of running our code.

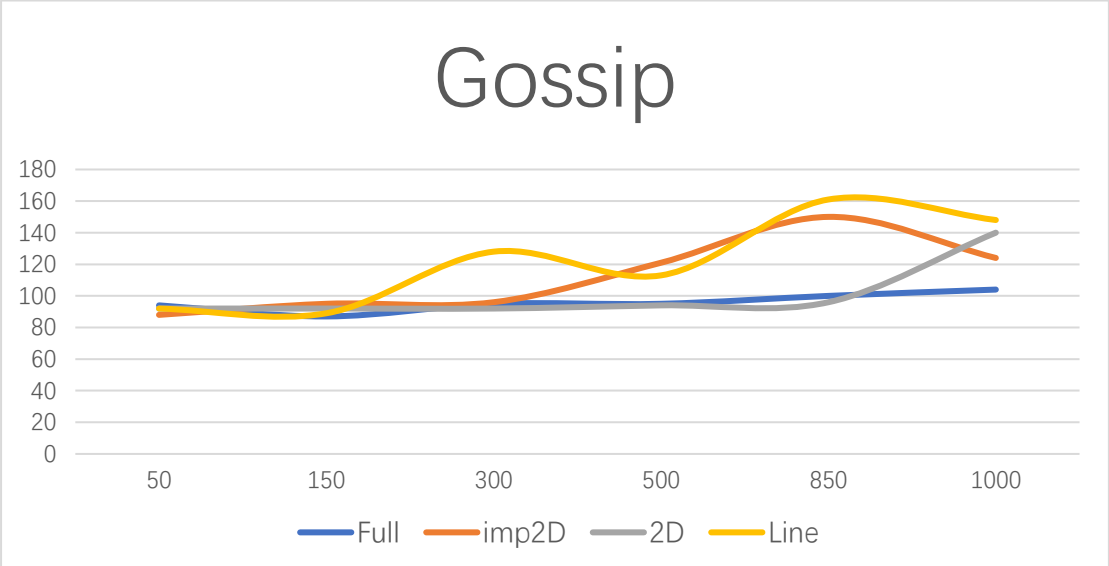
```
50 2D gossip 5
 ========== STARTING THE MASTER NODE ===============
start
 ==========  GOSSIP 2D  ================ 64
========== RANDOM NODES SELECTED ======= [|12; 2; 55; 40; 35|]
 ========== GOSSIP PROPAGATION 10.9375 ==========
 ========== GOSSIP PROPAGATION 21.875 ==========
 ========== GOSSIP PROPAGATION 32.8125 ==========
 ========== GOSSIP PROPAGATION 43.75 ==========
 ========== GOSSIP PROPAGATION 54.6875 ==========
 ========== GOSSIP PROPAGATION 65.625 ==========
 ========== GOSSIP PROPAGATION 76.5625 ==========
 ========== GOSSIP PROPAGATION 87.5 ==========
 ========== GOSSIP PROPAGATION 98.4375 ==========
The convergence time should be: 101.929600
```

```
50 imp2D gossip 5
 ========== STARTING THE MASTER NODE ===============
start
 ========== GOSSIP 2D IMP ================ 64
========== RANDOM NODES SELECTED ======= [|6; 21; 33; 1; 23|]
 ========== GOSSIP PROPAGATION 10.9375 ==========
 ========== GOSSIP PROPAGATION 21.875 ==========
 ========== GOSSIP PROPAGATION 32.8125 ==========
 ========== GOSSIP PROPAGATION 43.75 ==========
 ========== GOSSIP PROPAGATION 54.6875 ==========
 ========== GOSSIP PROPAGATION 65.625 ==========
 ========== GOSSIP PROPAGATION 76.5625 ==========
 ========== GOSSIP PROPAGATION 87.5 ==========
 ========== GOSSIP PROPAGATION 98.4375 ==========
The convergence time should be: 134.423200
```

**We could find that the answers above shows that the converge time of imp2D > 2D which meets the expected answers.**

Below is the answers and graphs of our convergence time.

| Number of nodes | Full | Imp2D | 2D | Line |
|---|---|---|---|---|
| 50 | 94 | 88 | 92 | 92 |
| 150 | 87 | 95 | 92 | 89 |
| 300 | 95 | 96 | 92 | 128 |
| 500 | 95 | 121 | 94 | 113 |
| 850 | 100 | 150 | 96 | 161 |
| 1000 | 104 | 124 | 140 | 148 |




| Number of nodes | Full | Imp2D | 2D | Line |
|---|---|---|---|---|
| 100 | 6715 | 7824 | 7982 | 7521 |
| 200 | 10245 | 13526 | 12236 | 14312 |
| 300 | 15316 | 12531 | 15323 | 14255 |

| | | | | |
|---|---|---|---|---|
| 400 | 23546 | 23291 | 25568 | 34511 |
| 500 | 32332 | 33615 | 34231 | 36144 |