Project 2 – Gossip Simulator Report

1. Group members

Name: Zixun Wang **UFID:** 3725-9823

Name: Yixin Wei **UFID:** 5114-6181

2. Design and Analysis

2.1 Topologies

a. Full Network: Because every actor is a neighbor of all other actors, we can choose a random actor to be one actor's neighbor.

b. 2D Grid: Actors form a 2D grid. Round up the numNodes (number of actors involved) until you get a square. Maximum number of neighbors are 4 and minimum is 2.

c. Line: Each actor has only 2 neighbors (one left and one right, unless you are the first or last actor), where an actor at position I has neighbors at position i+1 and i-1.

d. Imperfect 2D Grid: Grid arrangement but one random other neighbor is selected from the list of all actors (4+1 neighbors). Maximum number of neighbors are 5 and minimum is 3.

2.2 Algorithms

2.2.1 Gossip Algorithm for information propagation

2.2.1.1 Implementation

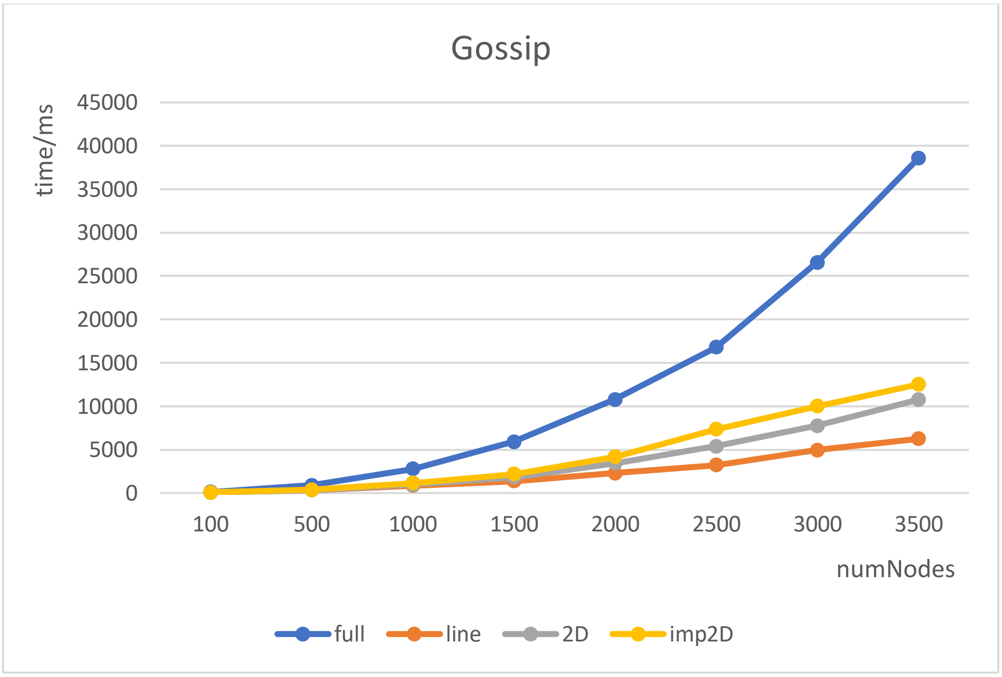
Define a boss actor that keep track of all the worker actors and perform the gossip algorithm.

a. Firstly, the ***boss actor*** gets the number of numNodes, the type of topology and gossip algorithm from console. The boss actor creates ***worker actors*** with the number of numNodes. An array with the size of numNodes is created to store neighbors of each worker according to the selected topology.
b. Boss actor tells a worker actor to start sending a message (rumor). The gossip program starts.
c. Upon receive the message from the boss, the first worker actor sends rumor to one of its neighbors according to the current topology.
d. Each actor receives the rumor starts to select a random neighbor and tells it the rumor.
e. Whenever a worker actor receives a message for the first time, it tells the boss actor. Hence, the boss actor can keep track of the number of remaining nodes who have not received the message and can also update the list of nodes who have received the message.
f. Each worker actor (node) keeps track of the number of times it has received the message. The worker actors which have received rumors continue sending rumor to a random neighbor until it has heard the rumor 10 times or terminate due to, for example, all the neighbors of it has terminated.
g. Once the boss actor finds out that every worker actor (node) has received the message, the gossip model works successfully, and we can print the total time it took and terminates the whole application.

2.2.1.2 Results and Findings

a. Results

The gossip algorithm of 4 curves, one for each topology as shown in the following figure.



As shown in the figure, the full topology has the longest time to reach the convergence, and the line topology has the shortest time to reach it.

Because the full topology has more neighbors than the others, while the line topology has the least neighbors. As the size of numNodes increase, the full topology will take up more memory because each node has to keep track of all its neighbors, which will take up large memory.

b. Interesting Findings

While using line topology, here is an example that produce failure.

For example, if the boss actor chooses the second worker actor to start the algorithm, the second one may choose one of its neighbors to tell the rumor. If it chooses the first one, and then the first one chooses the second one again, and second one chooses the first one as well, so on and so forth. Then the second node will first receive the rumor ten times and stop itself. At this time, only the first actor knows the message (rumor), but it has no neighbor now. Because its only neighbor (the second node) has been stopped. So the algorithm will turn to failure.

Maybe the probability of this circumstance is small, but failure would happen. Hence, the network can have blind spots.

2.2.2 Push-Sum algorithm for sum computation

2.2.2.1 Implementation

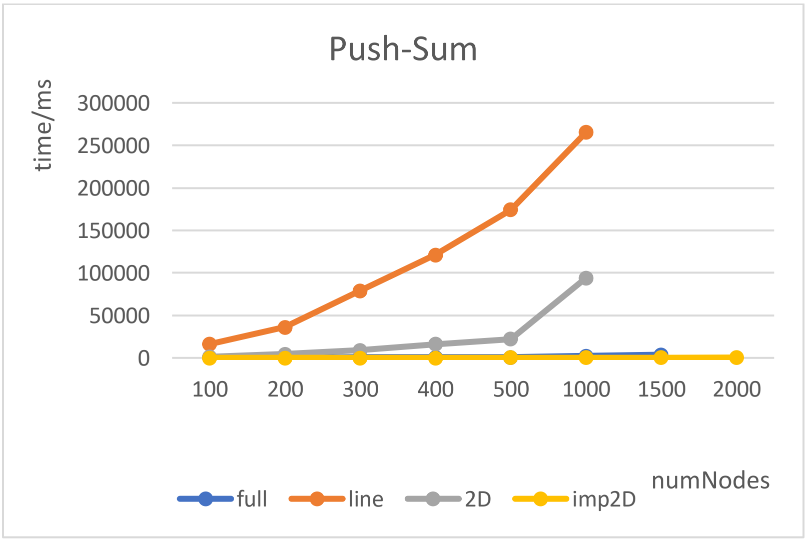
Define a boss actor that keep track of all the worker actors and perform the gossip algorithm.

a. Firstly, the *boss actor* gets the number of numNodes, the type of topology and push-sum algorithm from console. The boss actor creates *worker actors* with the number of numNodes. An array with the size of numNodes is created to store neighbors of each worker according to the selected topology.
b. Each actor Ai maintains two quantities: s and w. Hence, each worker actor node has two values in a pair form (s, w) as its state to send and receive messages.
c. Boss actor tells a worker actor to start sending a message (the pair form (s, w)). The push-sum program starts.
d. Upon receive the message from the boss, the first worker actor sends message to one of its neighbors according to the current topology.
e. 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. And we need to calculate the ratio s/w and record the change of the ratio each time when an actor receives a message.
f. Note that only upon receive, an actor adds received pair to its own corresponding value and selects a random neighbor to send it a message.
g. Once one worker actor's ratio s/w did not change more than 10^{-10} in 3 consecutive rounds, it tells the boss actor that its ratio (sum estimate) has converged. This worker actor (node) would not transmit any more.
h. Since one of the worker actors terminates, the other actors will not receive message and the program will terminate. Once one worker actor's ratio s/w has converged, the boss actor prints the total time it took to converge and terminates.

2.2.2.2 Results and Findings

a. Results

The push-sum algorithm of 4 curves, one for each topology as shown in the following figure.



As shown in the figure, the line topology has the longest time to reach the convergence, and the imperfect2D topology has the shortest time to reach it.

b. Interesting Finding

We find that the push-sum algorithm takes more time to converge than the gossip algorithm. Since we need to record the sum from one end of the network to the other end of that, which is impossible in large networks or line topology.

It is hard to converge the line topology using push-sum algorithm. We tried to analyze why this happened by changing various number of nodes in push-sum. Because the sum of each node has to reach all the other nodes, which is hard for line topology that only have two neighbors.