**TEAM MEMBERS**:

|  |  |
| --- | --- |
| **NAME** | **UFID** |
| NEHA SINGH | 41148924 |
| RUCHI DESAI | 75103513 |

**WHAT IS WORKING:**

**Gossip Algorithm:**

* Each actor has its own round during which it will send the rumor to a random neighbor. We achieved this by including a dispatcher within each actor which will send a message called ‘StartGossip’ periodically to that actor. The actor will only send the rumor to its neighbor after it receives the StartGossip message. The time interval for sending the StartGossip message is calculated as follows:

interval = (actor\_number % 5) \* 10

Hence all actors will not wake up at the same time and thus rounds are not synchronized. We have also ensured that actor will transmit the rumor, after receiving StartGossip message, only if it has received the rumor at least once.

* If an actor receives the rumor, then it will simply increment the counter which counts the number of times the actor receives the rumor.
* After an actor has heard the rumor for 10 times, it stops transmitting the rumor (we call this ‘going offline’). Thereafter, it will send a ‘Stop’ message to all its neighbors.
* If an actor realizes that all its neighbors have gone offline, then it will stop transmitting the rumor.
* Every actor keeps a track of the number of offline nodes at a particular point in time. We achieved this by having a global counter shared by all actors. When an actor finds out that all the nodes have gone offline, it will flood its neighbors with ‘Stop’ message. (We have assumed that when an actor is in offline state, the only message which it cannot send is the rumor message. It can send ‘Stop’ message.)
* Once an actor receives the ‘Stop’ message and it sees that all the nodes are offline, it will send a ‘Converge’ message to the master.
* When master (called as Converger) receives the converge message, it will print the convergence time.

**Push-Sum Algorithm:**

* Each actor has its own round during which it will send its sum to a random neighbor. We achieved this by including a dispatcher within each actor which will send a message called ‘StartGossip’ periodically to that actor. The actor will only send the sum to its neighbor after it receives the StartGossip message. The time interval for sending the StartGossip message is calculated as follows:

interval = (actor\_number % 5) \* 10

Hence all actors will not wake up at the same time and thus rounds are not synchronized. We have also ensured that actor will transmit the sum, after receiving StartGossip message, only if it has received the sum at least once.

* When an actor receives a sum from its neighbor, it will add it to the value that it already has.
* When an actor receives the StartGossip message, it will reduce its current value of sum by half. Then it will send this result to a random neighbor.
* After the ratio of s/w does not change by more than 10-10 for 3 consecutive times that it receives the message, the actor stops transmitting the sum (we call this ‘going offline’). Thereafter, it will send a ‘Stop’ message to all its neighbors.
* If an actor realizes that all its neighbors have gone offline, then it will stop transmitting the sum.
* Every actor keeps a track of the number of offline nodes at a particular point in time. We achieved this by having a global counter shared by all actors. When an actor finds out that all the nodes have gone offline, it will flood its neighbors with ‘Stop’ message. (We have assumed that when an actor is in offline state, the only message which it cannot send is the rumor message. It can send ‘Stop’ message.)
* Once an actor receives the Stop message and it sees that all the nodes are offline, it will send a Converge message to the master.
* When master (called as Converger) receives the Converge message, it will print the convergence time.

**THE LARGEST NETWORK WE MANAGED TO DEAL WITH FOR EACH TYPE OF TOPOLOGY AND ALGORITHM:**

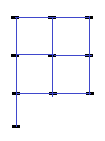
|  |  |
| --- | --- |
| **Gossip Algorithm** | |
| **Topologies** | **Number Of Nodes** |
| Line | 7000 |
| Full | 1040 |
| 2D | 10000 |
| Imp2D | 8000 |

|  |  |
| --- | --- |
| **Push-sum Algorithm** | |
| **Topologies** | **Number Of Nodes** |
| Line | 10000 |
| Full | 1900 |
| 2D | 20000 |
| Imp2D | 1000 |

**ASSUMPTIONS:**

* In 2D grid and imperfect 2D grid topologies, the nodes at the edges and corner have less than four and five neighbors respectively.
* If the number of nodes is not a perfect square then we don’t obtain a perfect grid. For example:

If number of nodes = 10, then we get the following grid:



**STEPS TO EXECUTE:**

Executing from the command prompt:

* Go to the path where the program is located

e.g.: cd C:\Users\neha\Desktop\Semester realted\semester2\DOS\project\project2

* Then type scala project2.scala 500 "full" "push-sum" in the command prompt.

Scala version: 2.11.2.

Akka version: 2.3.6