README

Group Members
Balaji Balasubramani - 98763981
Divyareddy Surapa Reddy - 13727408

Problem Statement:

Project 2:

Gossip type algorithms can be used both for group commu- nication 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.

Maximum Number of Nodes Considered for project 2 : 10000

Project 2 Bonus:

Implement node and failure models (a node dies, a connection dies temporar- ily or permanently).

Maximum Number of Nodes Considered for project2 Bonus: 1000

** This file includes the commands to run for both project 2 and project2 Bonus

Input:

1. numNodes - Number of nodes(actors) in participation, 1000 always in this case
2. topology - One of the networks: full, 2D, imp2D and line
3. algorithm - Either gossip or push-sum

Additional argument for project2 Bonus

4. removeNodes - Remove the nodes from the participation List.

Commands to run:

For project2:

dotnet fsi --langversion:preview project2.fsx <numNodes> <topology> <algorithm>

For project 2 Bonus:

dotnet fsi --langversion:preview project2Bonus.fsx 1000 <topology> <algorithm> <remNodes>

Implementation:

Project 2

Initial Steps

1. Built the topology by constructing neighbors list for each actor during its initiation. The neighbors list consists of only the remaining nodes present in the network.
2. Now, will start the given algorithm by randomly selecting a single actor/participant.

Algorithm

3. The information of the actor's neighbors are stored as a list (neighborsList) which is constructed during its initiation

4. Upon external rumour message arrival (from one of its neighbors actors), the actor sends the rumour message to one of its neighbours and also maintains its own state (external message count) of tracking the number of times rumour message is received by incrementing it by 1.
5. Also at a periodic interval it sends the rumour message to itself (self message) from the time the first message to it is received until it terminates.

Termination

6. The program terminates when all of its actors receive the rumour message exhaustion number of times.

Output

7. Time algorithm takes to converge for the given topology with n nodes in milliseconds

Project 2 - Bonus

For the Bonus, we developed an algorithm based on the number of nodes that the main process invokes (removing failure nodes, only remaining nodes is present) and handling convergence issues.

Initial Steps

1. Built the topology by constructing neighbors list for each actor during its initiation. The neighbors list consists of only the remaining nodes present in the network.
2. Now, will start the given algorithm by randomly selecting a single actor/participant.

Algorithm

3. The information of the actor's neighbors are stored as a list (neighborsList) which is constructed during its initiation
4. Upon external rumour message arrival (from one of its neighbors actors), the actor sends the rumour message to one of its neighbours and also maintains its own state (external message count) of tracking the number of times rumour message is received by incrementing it by 1.
5. Also at a periodic interval it sends the rumour message to itself (self message) from the time the first message to it is received until it terminates.

Termination

6. The program terminates when all of its actors receive the rumour message exhaustion number of times.

Output

7. Time algorithm takes to converge for the given topology with n nodes in milliseconds