# # Gossip Protocol

By: Shreyans Jain and Marcus Elosegui

# ## Running the Programs

The gossip\_protocol.fsx file can be found in the Gossip Protocol folder. In order to run it, please use the following command:

dotnet fsi gossip\_protocol.fsx <NumberOfActors> <Topology> <"gossip"/"pushsum">

The entire program will run for the provided number of actors, topology, and algorithm.

After the program has been executed and all the actors have received respective messages and have stopped transmission. The program will converge, and the runtime will be returned to the command prompt.

# ## Report

There are two different types of algorithms in the file

| Gossip

| PushSum

The file also contains 4 different types of topologies

| Full

This is like a mesh where each and every actor is connected to every other actor and has been implemented using an array where each actor has its own neighbor array

| 2D

This is where the entire actor has only actors which are horizontally and vertically adjacent to it and the entire topology is implemented using a 2D array with each actor having its own neighbor array

| Line

This is where each actor is connected to two or less actors depending on the position of the actors in the line topology

| Imp3D

This is where the actors form an improper 3D grid where each actor is connected to 6 neighboring actors

We divided the actors into two main parts,

The first actor is the process controller which keeps track of all of the nodes in the process and current topology

This actor is responsible of not only keeping track of all the nodes in the topology but also for providing nodes with neighbors in case the initial neighbors of that node dies. This is the actor that is responsible for stopping the process when all actors have been essentially stopped transmission.

The second type of actors are the nodes that are part of the topology

These actors are the actors that form the entire topology, these actors are the ones in between whom the entire communication is going on. These actors send both gossip and pushsum messages between each other. The messages are sent from one actor to another and increase a counter in case of gossip algorithm. As soon as an actor reaches a predefined counter which has been set into the program (10 in this case), the actor sends a message to the process controller which is the GossipTerminate message from this actor which means that it has achieved its task and the rest of the actors shall continue transmitting.

In case of the pushsum algorithm, instead of sending Rumor messages, the actors send a pushsum message that has the value of (s,w) these messages when received by an actor are added to the current values of s and w of that given actor. This actor then further sends half of this value to a random actor that is contained in the neighbor array of the given actor. Sending half leaves the actor with only half of the value. These messages are constantly and if the difference of the current value and the value presently contained by the actor is not equal to another given value (10 \*\* -10) in our case for a given number of rounds (3 for the given scenario)

In that particular case, the actor sends a PushSumTerminate message which once again tells the process controller that this actor has reached the given limit and that all others should continue.

The entire process terminates when all the actors have sent either PushSumTerminate or GossipTerminate message to the process controller. The entire program then terminates with the time being printed out to the terminal.

Following are the graphs that we found during the execution of the entire program: