COP5615 Project Report 2

**Group Members:**

1. **Sai Pradyumna Reddy Chegireddy (UFID: 3463-1711)**
2. **Sri Greeshma Avadhootha (UFID: 1613-6609)**

**Implementation Details:**

In our implementation of the gossip algorithm, a node terminates after hearing a rumor 60 times i.e. it stops passing the Rumor to a random neighbor. The convergence of the gossip implementation is measured when all the nodes in the network have terminated. For 2D grid and Imperfect 2D grid topologies, we round the number of nodes to the nearest perfect square.

In our implementation of Push-Sum algorithm, every node is initialized with the values suggested in the project handout of s = i and w = 1. Similar to our gossip implementation, the number of nodes is rounded to the nearest perfect square for 2D grid and Imperfect 2D grid topologies. The main process asks a random node to start which then passes a message consisting of a tuple of (s/2, w/2) to a random neighbor while keeping values of s/2 and w/2 as its state. When an actor receives a message tuple, it adds the tuple to its state and keeps half its value while passing on another half to a random node. This process continues until an actor’s s/w ratio does not change more than 1.0e-10 for three consecutive iterations after which the actor terminates i.e. it stops passing a tuple to a random neighbor and the algorithm converges when the sum estimates i.e. s/w converge to the average of the sum.

**Graphs plots: Convergence time (Milliseconds) vs Number of Nodes for each topology**

**Gossip Algorithm:**

The above graph plots the convergence times in milliseconds vs number of nodes for different topologies.

**Push-Sum Algorithm:**

The above graphs plot the convergence times of push-sum algorithm vs number of nodes for different topologies.

**Findings and Analysis:**

1. Line topology is the most inefficient in both Gossip and Push-Sum implementations. Especially in the case of Push-Sum, it takes inordinately long time to converge since it has only two consecutive neighbors and converging the sum estimate to the average of the sum will intuitively take a very long time. Our observations confirm this.
2. Full topology is the most efficient in both Gossip and Push-Sum implementations. Since every node is connected to every other node, the diffusion of the Rumor in the network is quickest among all topologies. This was confirmed in our observations for our Gossip implementation.
3. In a Full topology, since every node is connected to every other node, the sum estimate for each node should converge quickly to the average of the total sum. Our observations for the Push-Sum implementation for a Full topology bore out this prediction.
4. A 2D Grid topology with at most four neighbors for every node should improve on the performance of the line topology as it has more neighbors to spread the Rumor. This was borne out in our observations as nodes in a 2D Grid topology diffused the Rumor quicker than in nodes in a line topology.
5. An Imperfect 2D Grid topology with at most 4 regular grid neighbors and another randomly assigned neighbor should have a better performance than a regular 2D Grid topology in a Gossip implementation due to the extra random assignment. Our observations confirmed this expectation as the Rumor was diffused much more rapidly in an Imperfect 2D Grid topology compared to a regular 2D Grid topology.
6. Since there is an extra randomly assigned neighbor for each node in an Imperfect 2D Grid topology, the sum estimate converges quicker to the average of the total sum than in a 2D Grid topology. Our observations confirm this and show that an Imperfect 2D Grid topology has a superior performance than a 2D Grid topology in a Push-Sum Implementation.
7. The general order of convergence was more or less similar in both Gossip and Push-Sum implementations with Full topology being the quickest and a line topology being the slowest to converge. Imperfect 2D Grid topology showed better performance than a 2D Grid topology while being sandwiched between Full and Line topologies.