Gossip Simulator

Distributed Operating Systems

Project 2 - October 1, 2018

Madhukar Allu UFID : 79621171

Bhuvan Malik UFID : 70104341

Gossip Algorithm Introduction

Gossip type algorithms can be used both for group communication 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 Elixir. Since actors in Elixir are fully asynchronous, the particular type of Gossip implemented is the so called Asynchronous Gossip. The Gossip algorithm involves the following:

- **Starting:** A participant(actor) is told/sent a rumor(fact) by the main process
- **Step:** Each actor selects a random neighbor and tells it the rumor
- **Termination:** Each actor keeps track of rumors and how many times it has heard the rumor. It stops transmitting once it has heard the rumor 10 times.

Push-Sum algorithm for sum computation

- **State:** Each actor A_i maintains two quantities: s and w. Initially, $s = x_i = i$ (that is actor number i has value i, play with other distribution if you so desire) and $w = 1$.
- **Starting:** Ask one of the actors to start from the main process.
- **Receive:** Messages sent and received are pairs of the form (s, w). Upon receive, an actor should add received pair to its own corresponding values. Upon receive, each actor selects a random neighbor and sends it a message.
- **Send:** 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.
- **Sum estimate:** At any given moment of time, the sum estimate is s/w where s and w are the current values of an actor.
- **Termination:** If an actors ratio did not change more than 10^{-10} in 3 consecutive rounds the actor terminates. The values s and w independently never converge, only the ratio does.

Topologies implemented :

-> Full Network

-> Random 2D Grid

-> Line

-> 3D Grid

-> Torus

-> Imperfect Line

Implementation Instructions:

Build the file using "**mix escript.build**"

Navigate to the "**project2**" folder in the command line and enter,

"**escript Project2 numNodes topology algorithm**"

Make sure to enter the arguments in the mentioned order.

Output:

The output gives us the convergence time which is the time taken by the given set of arguments to form a network, run the algorithm and converge, which we have decided as (**Current system time** (end time) **- Start time**(time at which the first node receives the gossip)).

In some cases we have to wait for some time to make sure that there is no more convergence happening. In such cases the output time includes a wait time of 5000 milliseconds. Due to this, in certain topologies where randomization plays an important role, we see some significant variations in the plots. We will look into that further when we plot the graphs below and make observations.

Gossip Protocol :

Full Topology	
Size	Time
100	2231
200	2537
350	2693
500	2593
700	2782
850	2584
1000	2984

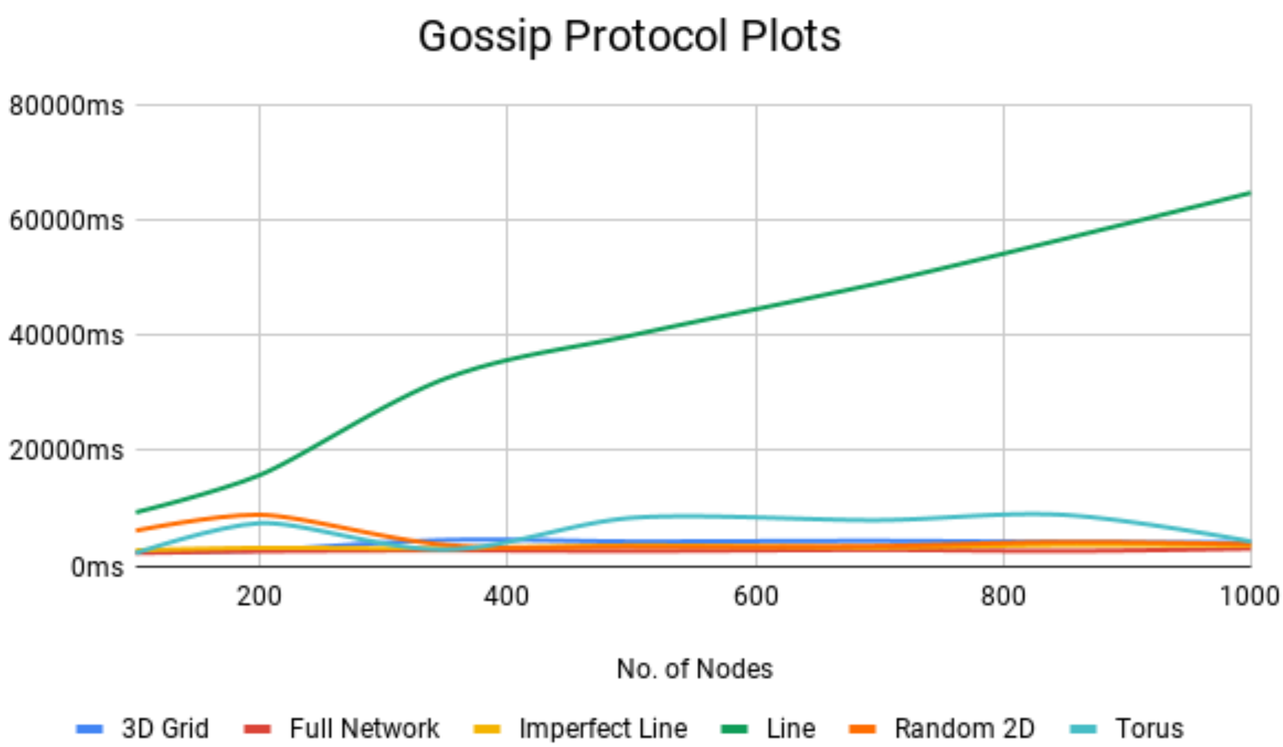
3D Topology	
Size	Time
100	2427
200	2730
350	4490
500	4244
700	4350
850	4147
1000	4049

Random 2D Topology	
Size	Time
100	6123
200	8872
350	3638
500	3340
700	3443
850	4150
1000	3752

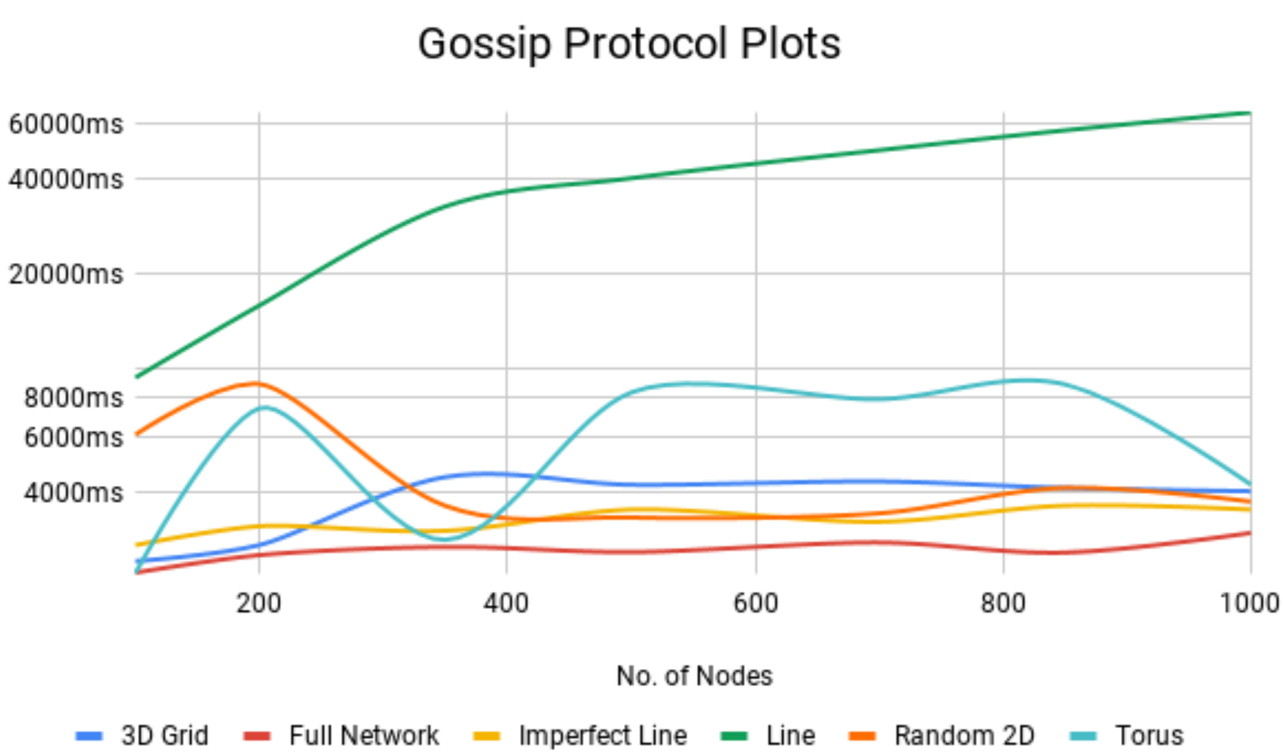
Torus Topology	
Size	Time
100	2225
200	7427
350	2836
500	8340
700	7937
850	8848
1000	4248

Line Topology	
Size	Time
100	9298
200	15762
350	32532
500	40005
700	49099
850	56792
1000	64755

Imperfect Line Topology	
Size	Time
100	2729
200	3135
350	3031
500	3540
700	3236
850	3643
1000	3540



The above graph is represented as a logarithmic plot in order to obtain more clarity for the lower plots, i.e. 3D, Full, Imp Line, Torus & Random 2D.



Observations:

The above graph plots the convergence time of different topologies vs Number of Nodes(the size of the network) for **Gossip** protocol. Following observations were made:

Line topology has the largest convergence time out of all. This is due to the limited number of neighbors (1 or 2 each) for each actor.

Full topology has small convergence time for smaller networks of 100 nodes but for larger networks its convergence time increases due to large memory occupied by the adjacent node lists for every node.

Imperfect Line topology gives a more or less constant performance because the size of neighbors list remains small and also since we are randomizing a neighbor.

3D grid topology is very well connected and hence has low convergence times.

Torus topology has a dip in the plot for certain sizes as some nodes do not converge because of the neighbors terminating before the message reaches them. Hence, the wait time of 5000 seconds is added to the convergence time, resulting in a higher result.

Random 2D topology gives us the most interesting observations due to the random placement of actors, and a fixed distance(0.1), within which the actors can communicate. As sometimes the actors are well connected and the convergence time is low and sometimes a few actors are not connected at all. Apart from this observation, the number of neighbors in a Random 2D is proportional to the size of the network, hence the performance gets better as the number of nodes increases, as the actors are closely packed when the size is big.

PushSum Protocol :

Full Topology	
Size	Time
100	929
200	1273
350	1443
500	2069
700	8088
850	9827
1000	5158

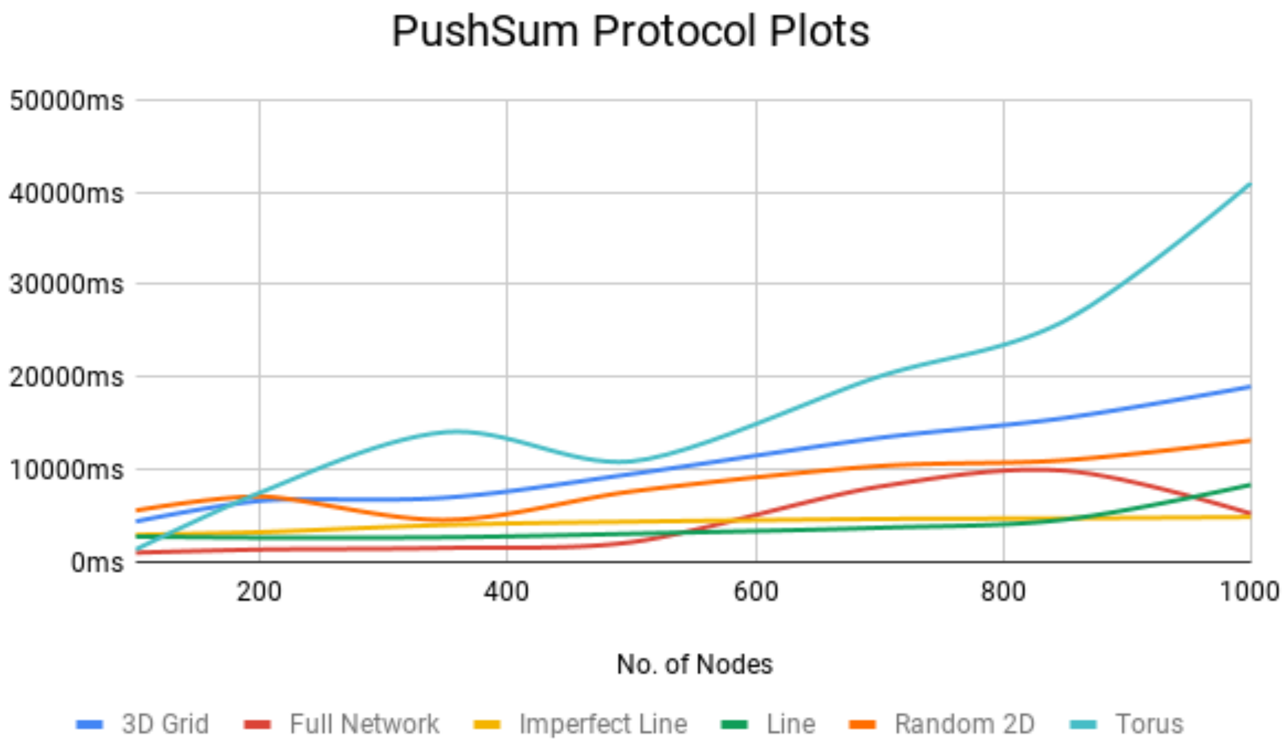
3D Topology	
Size	Time
100	4322
200	6551
350	6892
500	9457
700	13376
850	15534
1000	18928

Random 2D Topology	
Size	Time
100	5506
200	7031
350	4483
500	7565
700	10321
850	10956
1000	13091

Torus Topology	
Size	Time
100	1284
200	7431
350	14003
500	10825
700	20032
850	26083
1000	40997

Line Topology	
Size	Time
100	2686
200	2559
350	2600
500	2988
700	3630
850	4522
1000	8293

Imperfect Line Topology	
Size	Time
100	2878
200	3152
350	3940
500	4292
700	4589
850	4658
1000	4791



Observations:

The above graph plots the convergence time of different topologies vs Number of Nodes(the size of the network) for **PushSum** protocol. Following observations were made:

Torus topology's convergence time increases exponentially with size .

Full topology has least convergence time for smaller networks of 100 and significantly higher convergence time for bigger sizes. An interesting observation was that when the size increased from 850 to 1000, the convergence time dropped. We verified it by running the program several times and got the same results every time.

3D topology has gradually increasing convergence times as the size increases.

Line and **Imperfect Line** have the lowest convergence times for larger sizes. This is interesting since **Line** had the highest convergence time in Gossip protocol.

Random 2D topology gives us the most interesting observations due to the random placement of actors, and a fixed distance(0.1), within which the actors can communicate. As sometimes the actors are well connected and the convergence time is low and sometimes a few actors are not connected at all. Apart from this observation, the number of neighbors in a Random 2D is proportional to the size of the network, hence the performance gets better as the number of nodes increases, as the actors are closely packed when the size is big.