COP5615: Distributed Operating Systems (Fall 2018)

Project 2 - Gossip Simulator

Submitted by:
Kunwardeep Singh (UFID: 2421-3955)
Gayatri Behera (UFID: 3258-9909)

Algorithms Implemented

All nodes after being initiated, are being registered in a Map, with key as the Node ID and value as the Process ID.

a. Gossip:

   Gossip protocol as the name suggests, concerns itself with the dissemination of information between nodes by emulating a gossip environment i.e. a stream of gossip is initiated by some member of a group. It is then passed on to the other members of the group, by random selection of the next member.

   It is adopted with the aim of building fault-tolerant systems that are resilient to node failure at any point. The onus of such systems is that they tend to be decentralized (distributed) and ensures scalability, computation speed and good fault-tolerance.

   In our case, we terminated the actor/node involved in transmitting the gossip message once the actor had heard the rumor at least 10 times.

b. Push-Sum:

   Push-Sum protocol ensures a means of calculating the value of sum or average value at any node in a network. When the push-sum algorithm is initialized, it begins with a sum and weight value at a particular node, half of which is then disseminated to another node in a network. At the end of each iteration, the sum of all weights = n and sum of all sums = total sum. The assumption we place at that start is that the chosen node starts with weight = 1 and all others have weight = 0.

   In our case, we are keeping a note of the sum estimate i.e. s/w. If for any actor the ratio doesn’t decrease below 10^(-10) the actor is terminated.

Topologies implemented:

For implementing the 3D Grid, Random 2D grid and Toroid topologies - we decided to keep x, y and z coordinates for each node, as a means of identifying and visualising these nodes in vector space. In case of Random 2D grid and Toroid topologies, the nodes were visualized to be lying in a 2D vector space, so only the x and y coordinates of the nodes were populated and used for later identification.

i. Full Network -

In a full network there are no restrictions placed on the transmitting node while trying to select the next node. It is allowed to pick any node available from those present in the network.

ii. 3D Grid -
For visualising the nodes in a 3D grid, we got an estimate of the 3 dimensional space in which the nodes could possibly lie. In this space the nodes were sequentially placed. If a node chose to disseminate information it could only choose to do so amongst one of it's 6 neighbours.

iii. Random 2D grid -
For visualising the nodes in a 2D grid, we got an estimate of the planar space in which the nodes could possibly lie. In this space, the nodes were randomly distributed. A node was chosen to disseminate information to when it fit the metric of lying in 0.1 distance space.

iv. Toroid -
For visualising the nodes in a toroid, we got an estimate of the planar space in which the nodes could possibly lie. In this space the nodes were sequentially placed. The planar space was assumed to be curved along all it's edges to form a toroid. If a node chose to disseminate information it could only choose to do so amongst one of it's 4 neighbours.

v. Line -
In a line topology, the nodes are visualised to be equally-spaced along a line. While disseminating, the node chooses to randomly pick another node from either one of its neighbours.

vi. Imperfect Line -
Similar to line topology, the nodes are visualised to be equally-spaced along a line. While disseminating, the node chooses to randomly pick another node from either one of its neighbours or any one other node dispersed along the line.

Running the program:

To start the program in Unix environment, use:

```
time mix run proj2.exs [num_of_Nodes] [topology] [algorithm]
```

Algorithm parameter types:

- Gossip: gossip
- Push Sum: push_sum

Topology parameter types:

- Full Network: full_network
- Line: line
- Random 2D Grid: random_2d
- 3D Grid: 3d
- Imperfect Line: imperfect_line
- Toroid/Sphere: toroid

Sample output:

```
real    0m3.510s
user    0m1.124s
sys     0m0.212s
```

The real time shows the time taken to converge, whereas user+sys is the CPU time.

Convergence times:

a. Gossip algorithm:

Full Network:

Number of Nodes	Time Taken(s)
10	2.658
100	3.078
500	3.772
1000	8.564

Line:

Number of Nodes	Time Taken(s)
10	2.158
100	8.248
500	33.379
1000	74.615

Random 2D Grid:

Number of Nodes	Time Taken(s)
10	Not enough nodes
100	2.680
500	3.061
1000	5.682

3D Grid:

Number of Nodes	Time Taken(s)
10	2.027
100	2.735
500	3.178
1000	3.538

Toroid:

Number of Nodes	Time Taken(s)
10	1.991
100	2.585
500	3.738
1000	4.222

Imperfect Line:

Number of Nodes	Time Taken(s)
10	2.137
100	2.643
500	3.339
1000	4.476

b. Push-Sum algorithm:

Full Network:

Number of Nodes	Time Taken(s)
10	1.047
100	1.368
500	3.064
1000	5.930

Line:

Number of Nodes	Time Taken(s)
10	1.066
100	2.052
200	2.408
400	7.090

Random 2D Grid:

Number of Nodes	Time Taken(s)
10	Not enough nodes
100	1.577
500	4.183
1000	7.681

3D Grid:

Number of Nodes	Time Taken(s)
10	1.136
100	1.891
500	3.484
1000	5.895

Toroid:

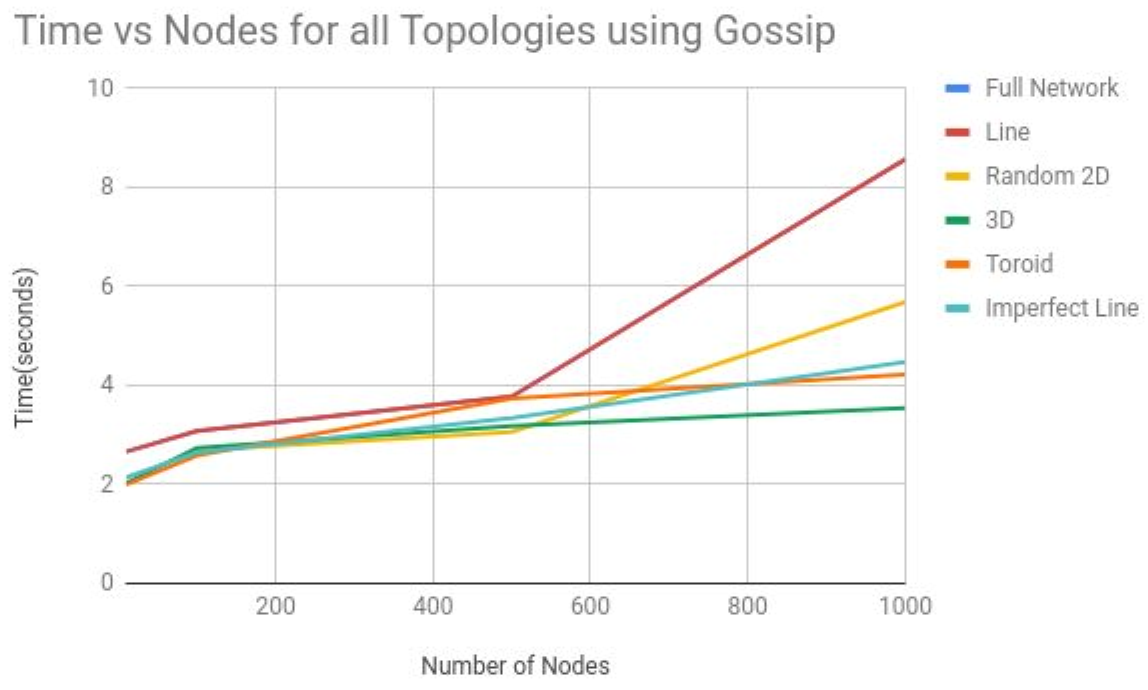
Number of Nodes	Time Taken(s)
10	1.133
100	1.842
500	3.294
1000	5.551

Imperfect Line:

Number of Nodes	Time Taken(s)
10	1.075
100	1.814
500	3.850
1000	6.525

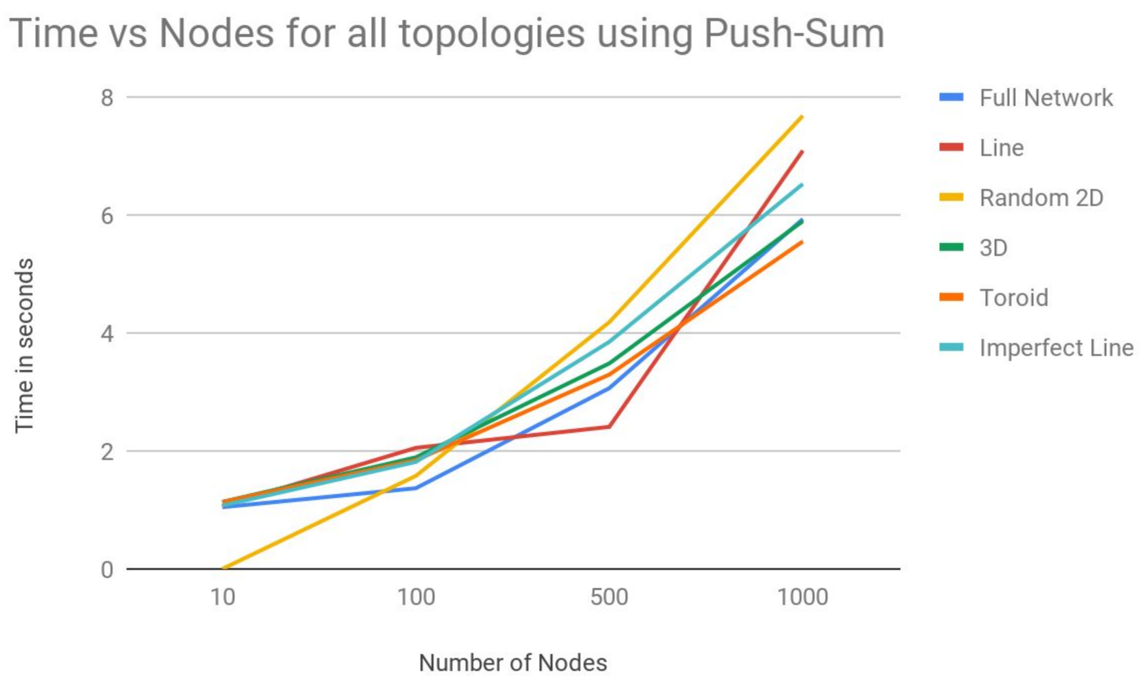
Plots and observations:

Analysis (Gossip Algorithm):



- For Gossip algorithm we observed that, as the number of actors/nodes increases the time taken for all the algorithms increases. This can be attributed to the fact that as number of nodes increases the time required to converge goes up as the pool of available neighbors to which the information can be transmitted to next goes up. The transmitting actor seeks out other actors randomly to transmit the information to.
- Best performing topologies are 3D grid and Toroid as they could handle upto 70000 nodes. Line is among the worst performers as it has fewer number of neighbours and the probability of getting isolated is high.
- Overall trend is as the number of nodes increase, time to converge also increases.
- Random 2D topology fails to converge when there are very few number of nodes in the network
- Fully connected network, although has most number of neighbours, but doesn't have a really good performance as chances of deadlock increase.

Analysis (Push-Sum Algorithm):



- For Push-Sum algorithm we observed that, as the number of nodes increases the time taken for all the algorithms increases. This can be attributed to the fact that as number of nodes increases the time required to converge goes up as the pool of available neighbors to which the information can be transmitted to next goes up.
- Best performing topologies are Toroid and imperfect line as they could handle upto 70000 nodes. These have the perfect balance between required number of neighbours and the parallel computations our systems can handle.
- Overall trend is as the number of nodes increase, time to converge also increases.
- Line is among the worst performers as it has fewer number of neighbours and the probability of getting isolated is high.
- Push sum appears to be getting deadlocked more number of times as compared to Gossip.
- Random 2D topology fails to converge when there are very few number of nodes in the network

References:

1. Gossip Based Computation of Aggregate Information, David Kempe, Alin Dobra, Johannes Gehrke
2. http://www.inf.fu-berlin.de/lehre/WS11/Wireless/Lectures/lecture13.pdf