
Report: Project 2 – Gossip Protocol in Scala (No Failure)

Students: Chiranjib Sur (UFID : 2531 4396)

Pramit Dutta (UFID : 7513 2433)

Course: COP 5615: Distributed Operating Systems Principles

The Gossip protocol have been widely used in distributed system as it provides the perfect way to distribute the information to all the participating nodes and also helps in solving many problems which are related to various kind of computations.

As a case study we have tried two such kinds. One of them is just passing a message and another is computing the average of the numbers possessed by each node.

Before discussing the results, the description of the various topology for the Gossip network has been revisited and a one line description has been provided.

Working of Algorithm:

Full Topology - Any actor can talk to any other actor as all are connected to each other.

Line Topology - The actors are arranged in a line like structure and each intermediate actor has two neighbors except the last and the first ones, which contains only one neighbor.

3DGrid Topology - A 3DGrid is formed with the number of actors in the form of a 3D model where each node is connected with 6 adjacent nodes except the ones at the surfaces. For the implementation we have considered that if the number of considered actors is not a perfect cube then the value is increased to make it a perfect cube and then the nodes are placed in the system.

Imperfect 3DGrid – This is the same thing as the 3DGrid system. The only difference or rather the addition is that an actor has a different random neighbor along with its usual adjacent nodes as in a 3DGrid. So in total we have $6 + 1$ neighboring nodes with one node as random to form the deformation.

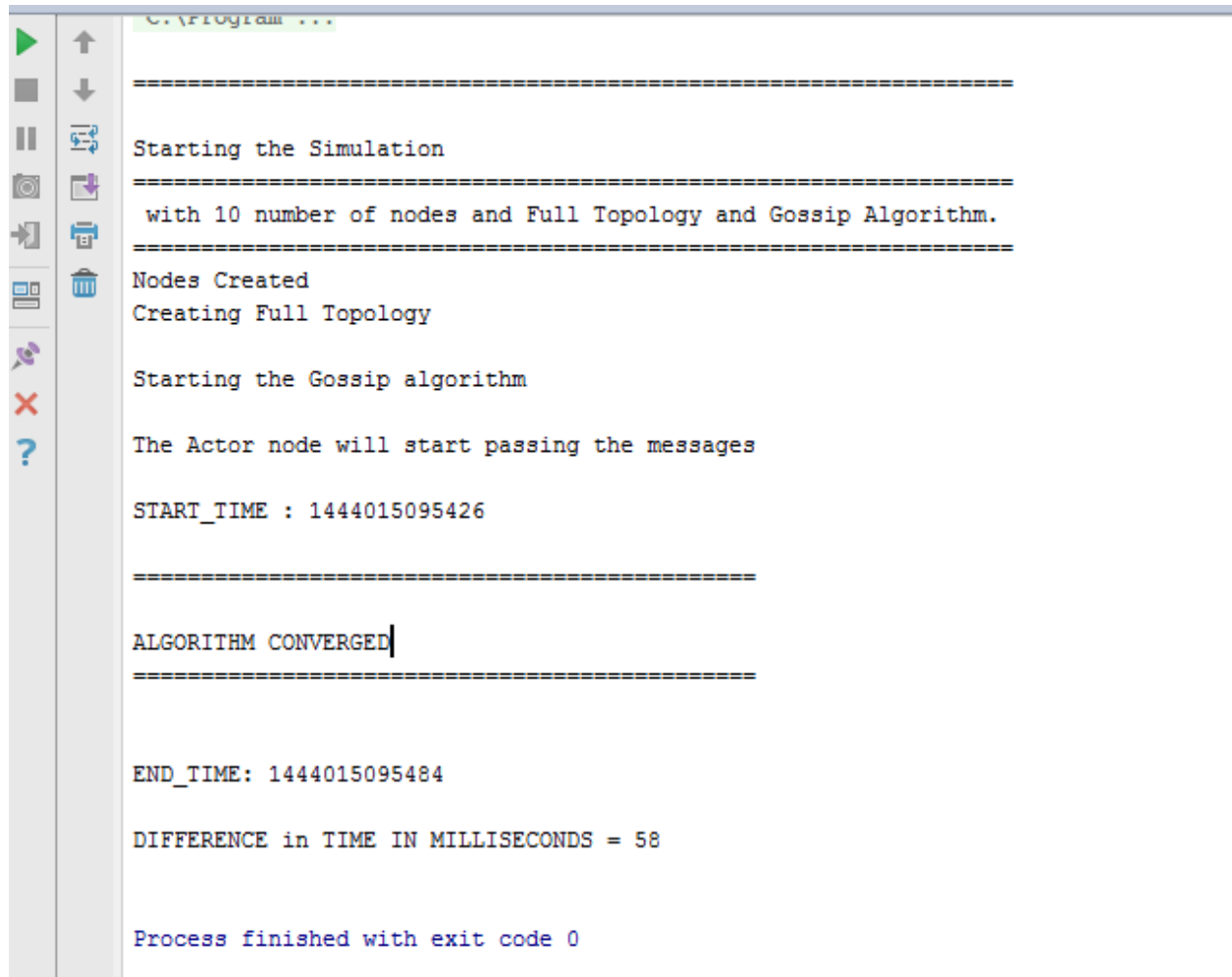
The actor framework of Scala has been used in this simulation as it helps in producing thread based agents which is supposed to run on different cores if present or provided. The actor model works on the basis of message passing based communication protocol and helps in running things parallel and independently without interfering with each other. But there are some overhead of termination

and coordination. However a little bit of care can provide such kind of provable coordination and convergence for the solution. However if the system is of low configuration, there can be the overhead of too many messages and loss of messages can arise exceptions.

The first part mainly works on simple and same message passing scheme and thus there is very low overhead and easy to implement. However the PushSum require some kind of taking care so that spreading of the information actually help in creating the aggregation for the system. To converge we have considered that if the value of computation for a specific actor doesn't change by more than 10^{-10} in 3 consecutive rounds, it must give up spreading the message. This is perhaps provided in the specification. Normal message passing scheme can work on the termination condition that it must stop once it has heard it for more than 10 times.

Now since the whole system works on probability, there is a chance that there are many nodes where the message may not reach or might be in such a portion where the gossiping is scanty. In that case the program may not converge and hence we have used that when more than 99% of the total numbers of nodes have heard a message at least once, the system shuts down. However in case of the line model, we have to use 90% to make it happen. Unlike the failure node based simulation, 90-95% spread of message in line protocol is pretty descent. However if the protocol runs for 80000, the spread need to degraded to 50% to get the result in time, which can withstand our patience. The following are the snap shot of the programs are being provided.

Snap Shots of the Program output



```
C:\Program ...  
  
=====
```

Starting the Simulation

```
=====
```

with 10 number of nodes and Full Topology and Gossip Algorithm.

```
=====
```

Nodes Created
Creating Full Topology

Starting the Gossip algorithm

The Actor node will start passing the messages

START_TIME : 1444015095426

```
=====
```

ALGORITHM CONVERGED

```
=====
```

END_TIME: 1444015095484

DIFFERENCE in TIME IN MILLISECONDS = 58

Process finished with exit code 0

The following are maximum number of nodes our system can handle to process in limited time.

Largest networks dealt with: Maximum number of nodes for which the Gossip Algorithm converged: (We tried with integers ranging from 10000 to 100000, increasing 10000 at a time and recorded these floors of convergences)

Gossip Algorithm:

Line topology- 80000 nodes (for 50% convergence)

Full – 50000 nodes

3DGrid – 30000 nodes

Imperfect 3DGrid- 30000 nodes

Largest networks dealt with: Maximum number of nodes for which the PushSum Algorithm converged: (We tried with integers ranging from 10000 to 100000, increasing 10000 at a time and recorded these floors of convergences)

PushSum Algorithm:

Line topology- 80000 nodes (for 50% convergence)

Full – 50000 nodes

3DGrid – 30000 nodes

Imperfect 3DGrid- 30000 nodes

If the system configuration is better and is made to run for a long time, it can process the convergence for high number of nodes with percentage of spread to much higher rate.

Results and Discussion

In the following figure (Fig 1) we have plotted the time for convergence for the different protocols for the message passing system. It can be easily seen that the full topology performs worst though it has the best connectivity. This is due to communication overhead it has faced and there may be much traffic and loss of data. Whereas the line protocol has communication scarcity and is unable to process the spread throughout the network. Whereas the 3d and the improper 3d has been able to gather much momentum due to more number of connectivity throughout the system network. It can also be seen that for low number of nodes, the difference in time for convergence is much less and as the number of nodes increases, the difference increases and so on. In fig 2, the difference in time spreads out because of change in condition of convergence

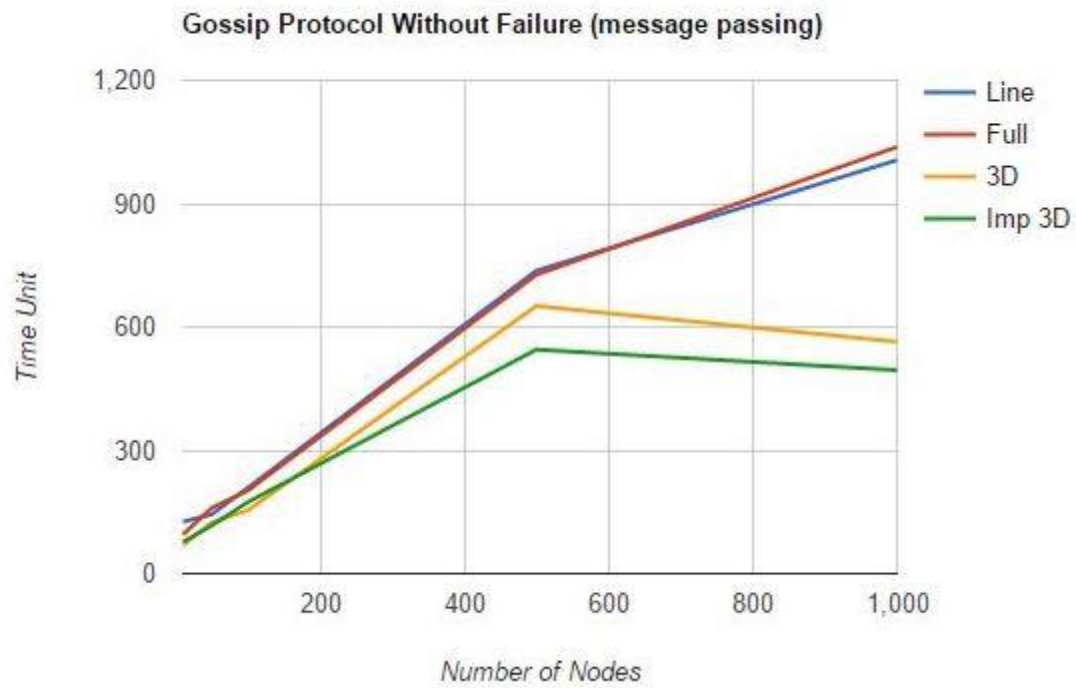


Fig 1: Plot for Different Topologies: Time vs Number of Nodes (only Message)

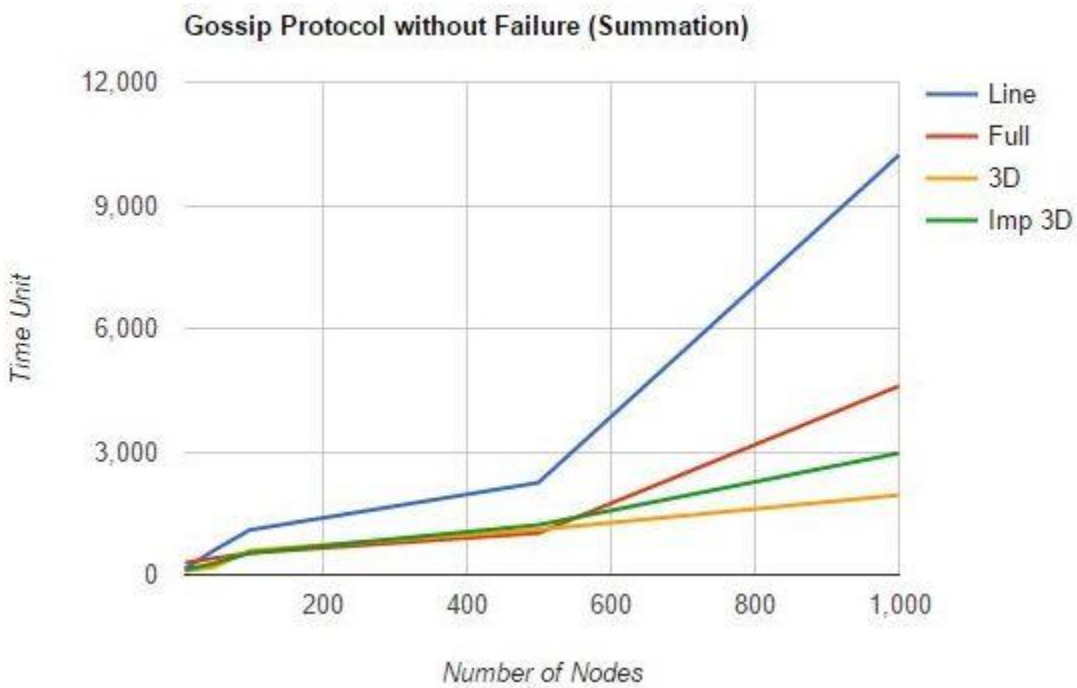


Fig 2: Plot for Different Topologies: Time vs Number of Nodes (PushSum)

Difficulty faced

While propagating the messages we faced an issue when we tried to pass message from one node to just another node. The Algorithms were not converging in that. When we increased the number of nodes a particular node can send messages to, the messages propagated fast and the algorithms converged.

- ➔ Convergence of the program
- ➔ Time calculation
- ➔ Scarcity of propagation, hence increased the propagation number
- ➔ Communication overhead can create problem and can make us feel that the node has fail to receive
- ➔ Sometime random and efficient node can help in better propagation

Findings:

We found out that the line topology has a lower rate of convergence when the number of nodes increase that is because line topology has the slowest rate of message passing among all the topologies we implemented. 3D and the Imperfect 3D topology seems to perform better over here in case of aggregating using the Gossip protocol.

References : http://opendatastructures.org/ods-java/12_2_AdjacencyLists_Graph_a.html
<http://alvinalexander.com/scala/iterating-scala-lists-foreach-for-comprehension>