COP 5615 Distributed Operating System Principles
Project 2_Bonus
Date - 10/07/2014

TEAM MEMBERS –

Mugdha Mugdha (54219168)
Palak Shah (55510961)

CODING ENVIRONMENT –

Scala Version – 2.11.2
Akka Version – 2.3.6
SBT version -0.13.6

INSTRUCTIONS TO COMPILE AND RUN –

1) To run Project 2_Bonus , go to the project folder and enter the sbt console

```
$ cd Project2_54219168_55510961
$ cd Project2_Bonus
$ sbt
```

- publishLocal
- runMain gossip.project2_bunus <number of nodes> <topology> <algorithm>

(If Project2_Bonus gives binding error, it is because of a hidden copy of application.conf might be there in its resources folder. This error can be removed by replacing the application.conf of Project2_Bonus with the application.conf of Project2)

<number of nodes> is the total number of nodes in the system
<topology> can be any one of the following – “line”, “full”, “2D” or “imp2D”
<algorithm> can be “gossip” or “push-sum”

INTRODUCTION –

In this part, we have handled the case where many nodes in a topology fail randomly. We have used a scheduler in the main class to fail one random node every 5 seconds.

In order to handle these node failures, we have used a centralized bossActor to poll the nodes regularly. This actor keeps a track of how many nodes are alive at a particular time and helps the algorithm to converge.

This method works correctly for “full” topology, but does not work so well for the “line” topology. The results we observed for each topology were very interesting.

FAILURE HANDLING –

We have created a bossActor that handles node failures and helps in graceful termination of the system.

The bossActor polls all the nodes at regular intervals to check whether they are alive. If a node is alive, it will send a return message to bossActor. On receipt of this message, bossActor increments a counter to determine how many nodes are alive. bossActor will use this value to help the gossip algorithm to converge. When all the living nodes have received a message, bossActor gracefully terminates the system.

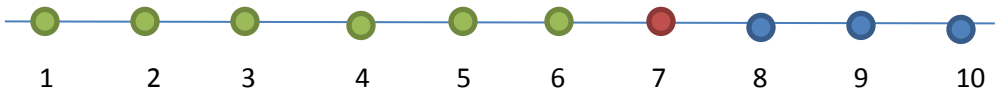
The push-sum algorithm does not need to know the number of nodes to converge. It will work as usual. The effect of node failure on push-sum is also interesting

OBSERVATIONS AND ANALYSIS –

GOSSIP –

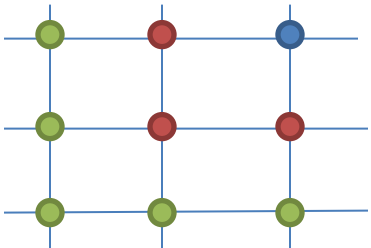
Here we present our results for each of the topologies for the Gossip algorithm –

1) Full Topology –
    - Full topology converges at almost the same rate as if there were no failures at all. This observation matches our expectations because in a full topology, all nodes can talk to all other nodes so there are no single points of failure.
    - In our tests, we observed that for small number of failures, the topology converges in almost the same time as the normal case.
    - We could not observe the performance for large number of failures, because our topology converges before we can fail more than a few nodes. But, we are guessing that for large number of failures, the topology will converge at the same rate.
2) Line Topology –
    - This method fails for a line topology because each failed node acts as a single point of failure. For example,



If node 7 fails before 8,9 and 10 have received a message, then nodes 8, 9 and 10 will never receive a message and the algorithm will never converge.

3) 2D Grid Topology –
    - In our experiments, we observed that node failures in a 2D grid topology are correctly handled by our approach.
    - But theoriticallly, we can prove that there are some rare cases of node failure that cannot be handled by this method. For eg if in the following figure, all the red nodes fail, then the blue node in top right corner may never receive a message



4) Imperfect 2D Grid Topology –
    - In our experimens, we observed that node failures in an imperfect 2D grid topology are also correctly handled by our approach
    - The shortcomings of 2D grid are overcome by including a random neighbour in Imperfect 2D Grid. This topology will converge at the same rate even if some of the nodes fail.

PUSH-SUM –

For push-sum algorithm, the aggregate value depends on the number of living nodes. The push-sum algorithm converges at nearly the same rate for small number of failures.

Although we could not test for large number of failures, we can guess that push-sum will converge at nearly the same rate.

Push-sum will take longer time to converge if the rate of failures is same as the length of a round in the algorithm.

CONCLUSION

In this project, we implemented a method based on polling to handle node failures. It worked correctly for Full and Imperfect 2D grid topologies. It failed miserably for Line topology. Although we did not observe failure in 2D Grid, we proved theoretically that it might fail for this topology also. From our observations, we conclude that it is easier to handle failure of nodes for topologies that have random neighbors, rather than rigid topologies