DOSP Project 2 – Gossip Protocol

Nirvisha Soni Neel Malwatkar

Problem Statement:

The objective of this project is to evaluate the convergence of Gossip and Push-Sum algorithms using a simulator based on actors implemented in Pony. The project aims to construct and experiment with various topologies, including full network, line network, 3D grid, and Imperfect 3D grid. These topologies define neighbor relationships which is crucial for algorithm performance.

Algorithm:

Gossip Algorithm for Information Propagation: The gossip algorithm is used to spread information across a network of nodes.

- 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 Ai maintains two quantities: s and w. Initially, s = xi = i (that is actor number i has value i) 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 receiving, an actor should add the received pair to its own corresponding values. Upon receiving, 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.
- Termination: If an actor's ratio was did not change more than 10^-10 in 3 consecutive rounds the actor terminates.
  Note: the values s and w independently never converge, only the ratio does.

Topologies:

Topology in the context of distributed systems refers to the arrangement and connection pattern of nodes in a network. Topologies play a crucial role in the performance and behavior of the gossip and push-sum algorithms. In this project, following four topologies are implemented:

1. **Full Topology**: Each node is connected to every other node, which allows information to spread quickly, as every node can communicate directly with all other nodes.

2. **Line Topology**: Nodes are arranged in a straight line where each node communicates with its immediate neighbors.

3. **3D Topology**: Nodes are arranged in a 3-dimensional grid, with each node connected to its surrounding neighbors.
4. **Imperfect 3D Topology**: Similar to the 3D topology but with additional random connections.

Implementation:

User Input: The program starts by prompting the user to input the desired algorithm (Gossip or Push-Sum), the number of nodes, and the network topology (full, line, 3D, imperfect 3D).

Network Initialization: Based on the user input, the program initializes the network with the specified number of nodes and establishes connections according to the chosen topology.

Node Creation: Each node is created as an actor and is assigned neighbors based on the selected topology.

Algorithm Execution:

- Depending on the user's choice of algorithm:
  - **Gossip Algorithm**:
    - The selected starting node begins the rumor-spreading process.
    - Nodes communicate with neighbors randomly, sharing rumors until all nodes converge. A node is said to be converged once it has received the rumor 10 times. For each node, a “Converged” flag is maintained which changes to True from False once it has converged.
  - **Push-Sum Algorithm**:
    - The selected starting node initializes its sum = node_id+1 (since node_ids start from 0) and weight = 1.
    - Whenever a node recieves message, it will add the Sum and weight in the message to itself. Then it takes away 50% of both sum and weight, passes onto a neighbour(chosen randomly).
    - Nodes update their sums and weights, iterating until convergence is detected. For push sum, convergence is achieved when the ratio of s and w has not changed more than 10^-10 in 3 consecutive rounds.

Convergence Check:

- For both algorithms, nodes track their convergence status.
- When all nodes have converged, the program outputs the results and concludes.

Potential Issue:

In this implementation, a potential issue arises when a node converges and stops transmitting messages. This creates scenarios where some nodes may not have converged yet, but there are no active messages being sent to them. As a result, these nodes become isolated in their state, unable to receive the necessary information to converge. This lack of communication can hinder the overall effectiveness of the algorithm, leading to incomplete (not 100%) convergence across the network.

Addressing the Issue:

In Gossip Algorithm: To tackle this problem, each node maintains a record of its neighbors and their convergence status. When a node observes that all its neighbors have converged but it itself has not, it initiates a "pull" for messages from any of its neighbors. This action reactivates the neighbor, allowing it to supply the necessary information for the non-converged node to eventually converge. This mechanism ensures that a 100 percent convergence is achieved across the network.

In Push-Sum Algorithm: However, the same approach is not feasible for the Push-Sum algorithm. In this context, when a non-converged node reactivates its neighbors and pulls messages, it may lead to an increasing ratio of the sum (s) to the weight (w) for that node. This can complicate the convergence process, as the ratios may become unstable and deviate further from the true average.

To address this challenge in the Push-Sum algorithm, we employ a different strategy. If a threshold percentage of nodes—let's say 90%—has converged, the entire network is then considered converged. This approach allows the network to reach a consensus on convergence without requiring every single node to individually finalize its state, effectively improving efficiency and stability in the convergence process.

Largest Network:

The largest network that is working for each topology of Gossip Algorithm is – 10k Nodes.

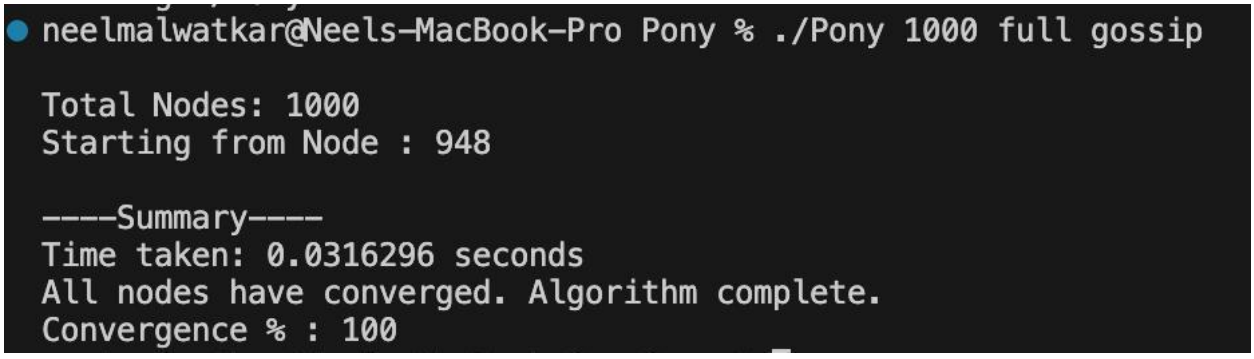
The largest network that is working for each topology of Push-Sum Algorithm is – 2k Nodes.

Apart from this, some topolgies are able to run for larger networks individually.

Topologies	Largest Network (Gossip)	Largest Network (Push Sum)
Full	15,000	12000
Line	10,000	2000
3D Grid	30,000	10,000
Imperfect 3D Grid	30,000	10,000

Sample Input and Output:

For Gossip Algorithm:



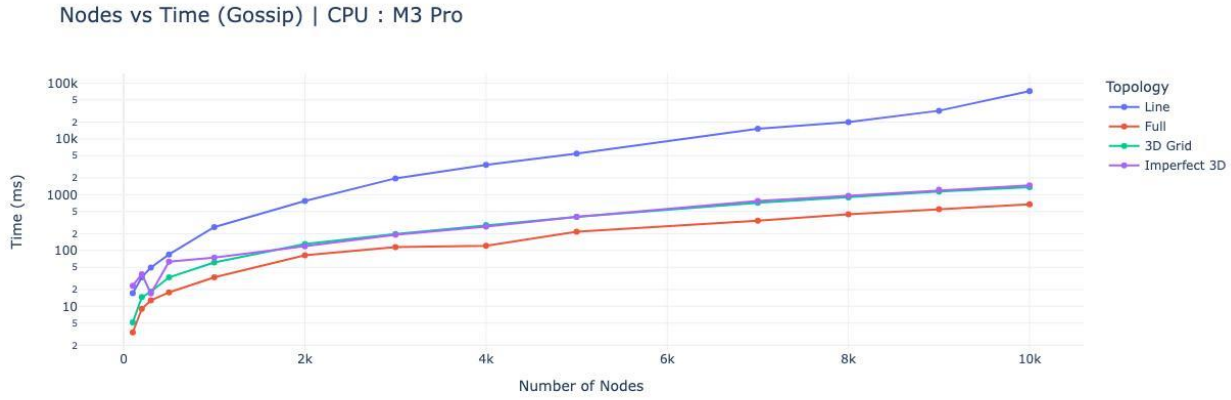
For Push Sum Algorithm:

```
neelmalwatkar@Neels-MacBook-Pro Pony % ./Pony 1000 full push-sum

Total Nodes: 1000
Starting from Node : 413

----Summary----
Time taken: 0.2412 seconds
All nodes have converged. Algorithm complete.
Convergence % : 100
```

Time-Series Plots for Convergence Time vs Network Size:



Topology Impact:

- **Line Topology**: The line topology exhibits the longest convergence time among all configurations, indicating that the linear structure may hinder efficient information propagation due to its limited connectivity.

- **Full Topology**: This topology shows a lower convergence time, which is expected as each node can directly communicate with every other node, facilitating rapid information exchange.
- **3D Grid Topology**: The 3D grid exhibits a moderate convergence time, suggesting a balance between connectivity and the hierarchical structure.
- **Imperfect 3D Topology**: The imperfect 3D grid shows a performance similar to that of the 3D grid, which implies that slight imperfections do not significantly degrade the convergence time in this setup.

Interesting Observations:

- During the implementation the importance of fault tolerance in decentralized systems is highlighted. The Gossip algorithm's ability to pull messages can serve as a safeguard against node failures, while Push-Sum's threshold approach can maintain overall network functionality even in the face of non-responsive nodes.
- The implementation reveals how node density and connectivity influence the algorithms' performance. For example, in sparse topologies (like line), message transmission becomes bottlenecked, while denser networks (like full) allow for more effective information spread. This can inform decisions in real-world applications, particularly in resource-constrained environments.
- The graph illustrates that as the number of nodes increases, the convergence time does not increase linearly; instead, it appears to approach an asymptote. This implies that after a certain point, the addition of nodes may not significantly slow down convergence, indicating an efficiency threshold in large networks.