Indian Institute of Technology Jodhpur

Computer Science and Engineering Department

Name : Lokesh Chaudhari

Roll : B21CS041

Name : Mohit Kumawat

Roll : B21CS046

Date: 15-02-2024

Gossip Protocol Implementation(README)

Description:

This project implements a Gossip protocol over a peer-to-peer network to broadcast messages and check the liveness of connected peers. The network consists of Seed Nodes and Peer Nodes. Seed Nodes make it easier for new Peer Nodes to register. They also keep track of peer lists and handle alerts for dead nodes. When Peer Nodes link to Seed Nodes, they make connections with other peers and send out gossip messages.

This repository contains the implementation of a Gossip protocol over a peer-to-peer network as per the provided assignment. The project is divided into two main components: seed.py for seed nodes and peer.py for peer nodes.

Directory Structure:

seed.py: Implements the Seed Node functionality.

peer.py: Implements the Peer Node functionality.

config.txt: Contains the IP addresses and port numbers of seed nodes.

Code details:

Seed Node(seed.py):

The seed.py script initializes a Seed Node. It listens for incoming connections from Peer Nodes, manages the connection, and updates the peer list accordingly. The Seed Node outputs connection requests and dead node messages to the console and a log file (seed_network.log).

1. The script initializes a socket, binds it to the specified IP address and port, and listens for

incoming connections.
2. manage_connection function handles the communication with each connected Peer Node.
3. Peer Nodes sending a "Disconnected Node" message trigger the removal of the respective node from the peer list.

Peer Node(peer.py):

1. Implements a peer-to-peer connection and message handling.
2. Periodically checks liveness by sending liveness requests and handles liveness replies.
3. Gossip protocol: generates and broadcasts gossip messages, maintaining a Message List (ML).
4. Outputs relevant information to the console and a log file (peer_network.log).
5. Handles peer connections, dead node reporting, and forwarding gossip messages.

Steps to compile and run:

Configure config.txt:

Open the config.txt file and provide the IP addresses and port numbers of seed nodes.

Seed Node:

1. Open the project folder.
2. Open the file "config.txt" and provide the system IP and port number in the format "IP:PORT". Add entries for each seed node.
3. Run the Seed Node script using the command. python seed.py
4. Run multiple instances of the seed script with different port numbers based on the entries in the config file.

Peer Node:

1. Run peer node script using the command: python peer.py
2. Enter a distinct port number for each peer.
3. Run multiple instances of the peer script to generate more peers.

Note:

Output files and logs will be generated to capture connection requests, gossip messages, and dead node notifications.Enter a distinct port number for each peer.