Computer Networks
Assignment 1

Overview of the Problem:

The objective is to design and implement a Gossip protocol over a peer-to-peer network to broadcast messages and ensure the liveness of connected peers. In this context, "liveness" refers to the ability of a peer to respond to messages sent to it.

Implementation:

We implemented the above-mentioned ‘Gossip Network’ using socket programming. Major problem we encountered was implementing parallel connections between various seed nodes and peer nodes. To solve this problem, we used python Select module and timer functions. We avoided using multithreaded programming to ensure better performance. In our solution, Seed nodes are implemented to act as a Server and Peer nodes are implemented to act as both Server and Client.

Code Explanation:

1) **Seed.py :** This file contains a class for Seed node which contains all the attributes and methods required for Seed functionalities. In the “activate()” function, we continuously listen to connection requests and messages from peers using the Select module. We have created various utility functions like “acceptConnection()”, “removeDeadNode()”, “writeLog()” to separate code for various tasks. Tasks are self explanatory based on the name of the method. In the main function, We are creating an instance of the Seed class and we can activate the Seed node by calling the “activate()” method.

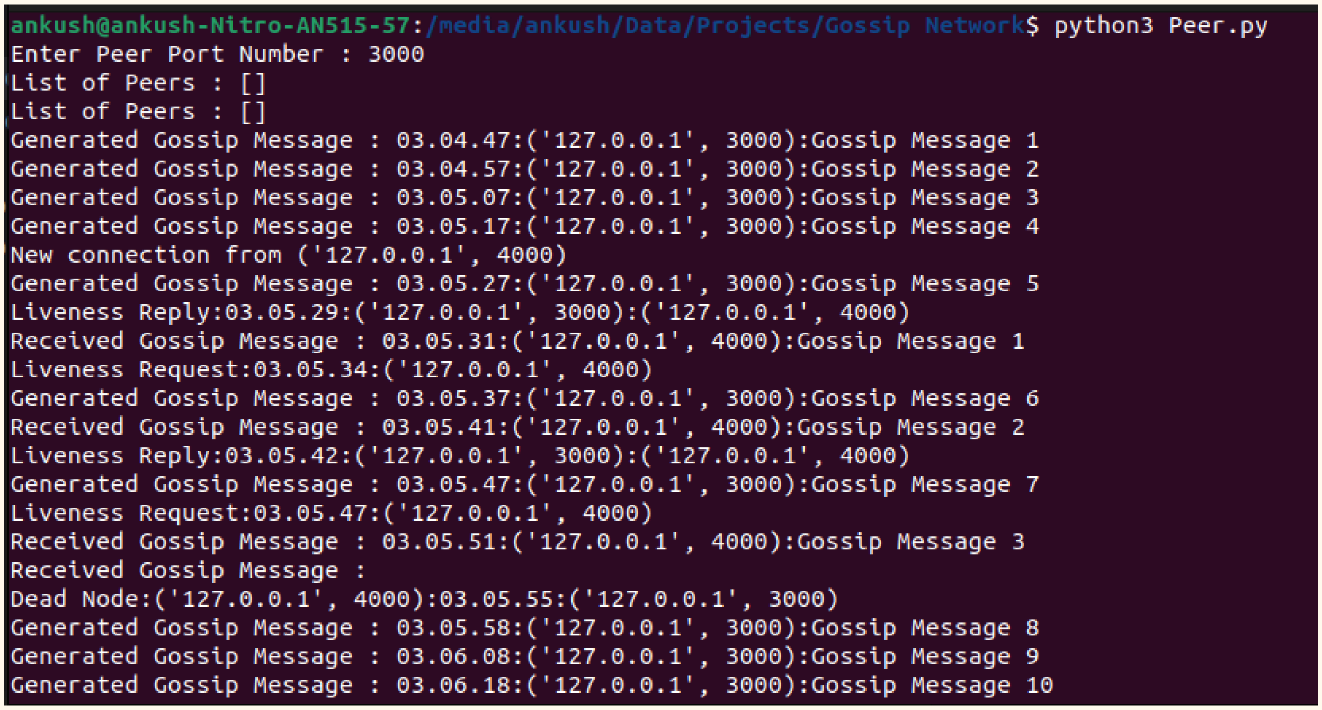
2) **Peer.py :** This file contains a class for Peer node. Peer class contains various attributes which we have explained in code comments. Peer is acting as both server and client. When Peer is created it sends connection requests to seed nodes and peer nodes using client sockets and creates various connection objects for communicating with them. Also Peer node is continuously listening for connection requests from new peers as a Server and creating socket objects to communicate with them. Also as a peer node is acting as both server and client, there are two addresses/ports we can use as a peer address i.e. one port where the peer program is running and one port where peer is listening as a Server. For simplification, We considered peer Server address as peer address. "establishConnections()" method randomly selects (n / 2 + 1) seed nodes and establishes socket connection with them. After receiving the peer list from all of the seed nodes, it connects randomly to a maximum of 4 peer nodes. In the function "activate", we monitor all the sockets for any event using the Select module. "generateGossipMessage()" and "generateLivenessMessage()" are timer functions which are called periodically to generate messages periodically after a fixed time period. Rest of the functions are utility functions created to separate code based on various tasks. We have implemented necessary exception handling and added all the necessary comments in the code. If there is an error sending a message to a particular peer, that peer is considered as a dead node and removed directly. If there is no response from a peer node 3 consecutive times, the node is considered as a dead node and is reported to all the connected seed nodes. Similar to Seed, we can create instances of Peer in the main and execute the Peer program.

Steps to execute the program:

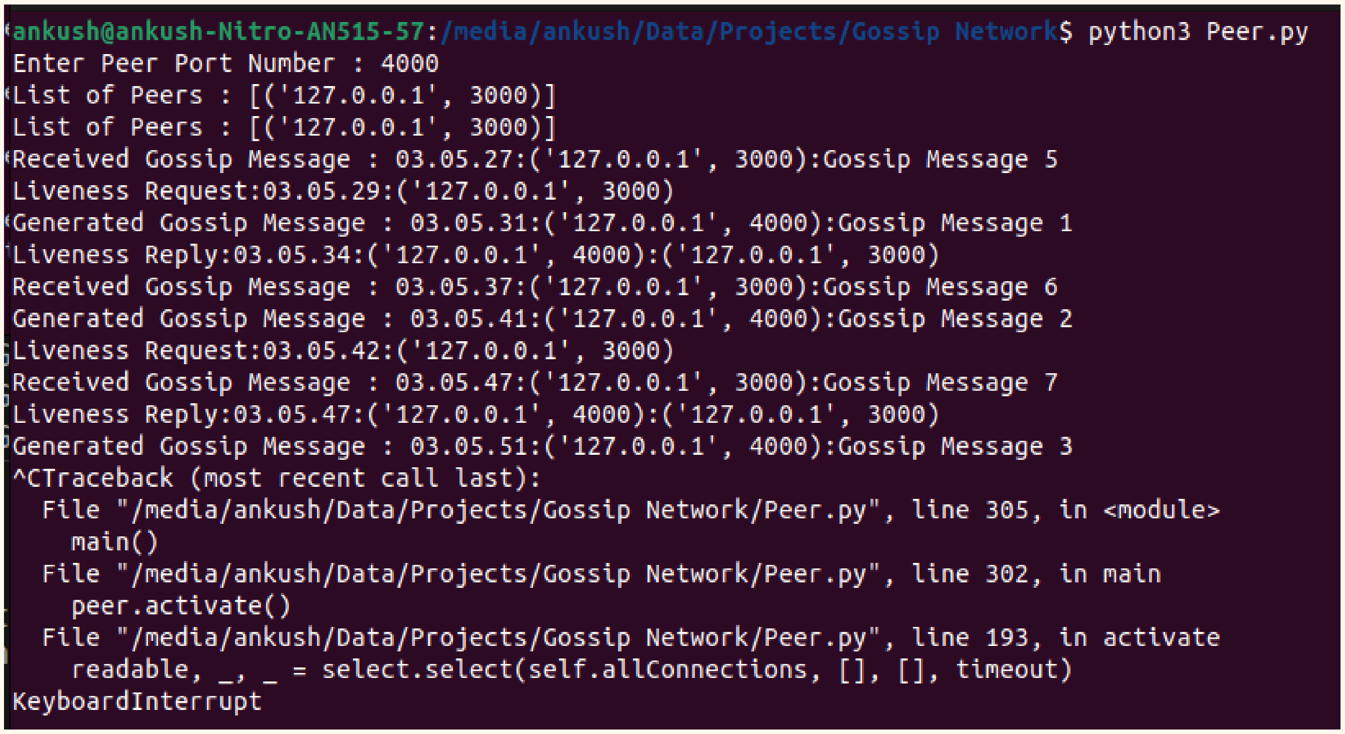
1) Run all the Instances of Seed nodes on different addresses. To run a seed node, execute command "python3 Seed.py"(Linux) or "python Seed.py"(Windows) and then provide address i.e ip address and port number as an input.

2) Add a list of all the addresses where seed nodes are running in the file "config.txt".
3) Execute Peer.py file keeping config.txt file with it. You need to provide an address i.e. ip address and port number. On this address server of Peer will listen to connection requests from newly connected peers and this address will be used as peer address.
4) We can create any number of instances of peer nodes. We can see output messages of Peer nodes and Seed nodes in log files and also on the console. We can close any peer using 'keyboard Interrupt' i.e. ctrl + c. Log files will be created for each node instance named "peer/seed_log_portnumber.txt"
5) Make sure to delete all the log files before running the network again to remove old logs and write(append) new logs to a new file which will be created automatically.

Output Screenshots: Output with two seed nodes



```
ankush@ankush-Nitro-AN515-57:/media/ankush/Data/Projects/Gossip Network$ python3 Peer.py
Enter Peer Port Number : 3000
List of Peers : []
List of Peers : []
Generated Gossip Message : 03.04.47:('127.0.0.1', 3000):Gossip Message 1
Generated Gossip Message : 03.04.57:('127.0.0.1', 3000):Gossip Message 2
Generated Gossip Message : 03.05.07:('127.0.0.1', 3000):Gossip Message 3
Generated Gossip Message : 03.05.17:('127.0.0.1', 3000):Gossip Message 4
New connection from ('127.0.0.1', 4000)
Generated Gossip Message : 03.05.27:('127.0.0.1', 3000):Gossip Message 5
Liveness Reply:03.05.29:('127.0.0.1', 3000):('127.0.0.1', 4000)
Received Gossip Message : 03.05.31:('127.0.0.1', 4000):Gossip Message 1
Liveness Request:03.05.34:('127.0.0.1', 4000)
Generated Gossip Message : 03.05.37:('127.0.0.1', 3000):Gossip Message 6
Received Gossip Message : 03.05.41:('127.0.0.1', 4000):Gossip Message 2
Liveness Reply:03.05.42:('127.0.0.1', 3000):('127.0.0.1', 4000)
Generated Gossip Message : 03.05.47:('127.0.0.1', 3000):Gossip Message 7
Liveness Request:03.05.47:('127.0.0.1', 4000)
Received Gossip Message : 03.05.51:('127.0.0.1', 4000):Gossip Message 3
Received Gossip Message :
Dead Node:('127.0.0.1', 4000):03.05.55:('127.0.0.1', 3000)
Generated Gossip Message : 03.05.58:('127.0.0.1', 3000):Gossip Message 8
Generated Gossip Message : 03.06.08:('127.0.0.1', 3000):Gossip Message 9
Generated Gossip Message : 03.06.18:('127.0.0.1', 3000):Gossip Message 10
```



```
ankush@ankush-Nitro-AN515-57:/media/ankush/Data/Projects/Gossip Network$ python3 Peer.py
Enter Peer Port Number : 4000
List of Peers : [('127.0.0.1', 3000)]
List of Peers : [('127.0.0.1', 3000)]
Received Gossip Message : 03.05.27:('127.0.0.1', 3000):Gossip Message 5
Liveness Request:03.05.29:('127.0.0.1', 3000)
Generated Gossip Message : 03.05.31:('127.0.0.1', 4000):Gossip Message 1
Liveness Reply:03.05.34:('127.0.0.1', 4000):('127.0.0.1', 3000)
Received Gossip Message : 03.05.37:('127.0.0.1', 3000):Gossip Message 6
Generated Gossip Message : 03.05.41:('127.0.0.1', 4000):Gossip Message 2
Liveness Request:03.05.42:('127.0.0.1', 3000)
Received Gossip Message : 03.05.47:('127.0.0.1', 3000):Gossip Message 7
Liveness Reply:03.05.47:('127.0.0.1', 4000):('127.0.0.1', 3000)
Generated Gossip Message : 03.05.51:('127.0.0.1', 4000):Gossip Message 3
^CTraceback (most recent call last):
  File "/media/ankush/Data/Projects/Gossip Network/Peer.py", line 305, in <module>
    main()
  File "/media/ankush/Data/Projects/Gossip Network/Peer.py", line 302, in main
    peer.activate()
  File "/media/ankush/Data/Projects/Gossip Network/Peer.py", line 193, in activate
    readable, _, _ = select.select(self.allConnections, [], [], timeout)
KeyboardInterrupt
```

Made by:-

1)Anurag Kumar Bharti [B21CS012]

2)Ankush Deshmukh [B21CS022]