Design:

The router first becomes alive and stay in standstill mode and waits for any server connection. Then when any of the server becomes alive it automatically connects to the router and also spawn a slave server. When any of the master/slave dies, it automatically creates another process that has another slave/master server(which automatically connects to the router). client always can request to the router for an available server host-port and the router will provide it. All new processes are created in the background.

In a VM, whenever a server dies, slave becomes master and server gets a new slave. In terms of client side, whenever available_master_1 dies, available_master_2 becomes 1 and gets a new available_master_2.

To see how to compile/run/test the program…see **Readme.md** file.

There is also a continuous bidirectional rpc running between connected clients and a server. When a server(master1) dies, client makes its master2 as master1 and connects to it, and requests router for an additional master to populate position for master2. When a client dies, server deactivates the client for that time. Later again when client restarts computing, it gets all the data from the client.

We used sqlLite server to store users, follow, unfollow, list data.. posts are explicitly stored in user.txt files and stored in two vms at a time. All the txt files across the vms are consistent and will provide same posts when pulled.

Below are mentioned each of the rpcs that been used and their respective purposes:

Onetime_c_to_r: this rpc is used to communicate between client and router. Client requests router to give it a pair of master1 and master2 using this rpc. This rpc is onetime.

Bidirectional_r_and_s: this rpc is used between router and servers. Whenever server becomes alive, it connects to the router to let router know that its available. Router will store this data in a global container. Whenever server dies, router process deactivates that server.

BuddyConnection: Using this rpc, process1 and process2 in the same VM communicates each other. This is a bidirectional rpc which constantly listens to each other. When master or slave dies, other process resurrects it in the background.

Bidirectional_s_and_c: This rpc works berween server and connected clients. when server dies, client uses another rpc to call router to give it a new master.And, when client dies, server deactivates that clients.

BroadcastPosts: This rpc is used between VMs to send the user_posts,txt files over the network.

Onetime_s_to_buddy: this rpc is used to communicate between process1 to its process2 to let the other process know about a new update.