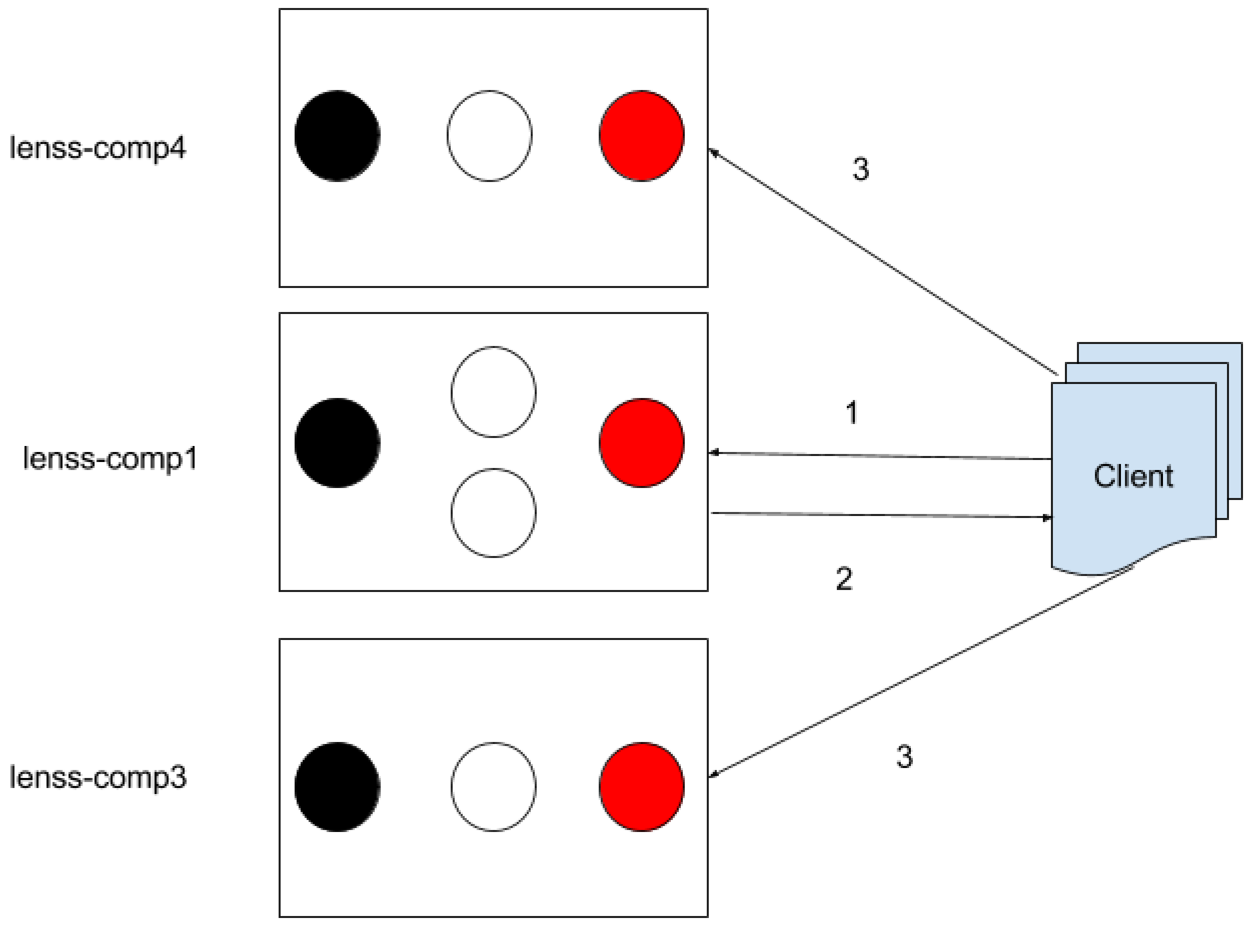
HW3 A Scalable and Highly Available Twitter Service

Liuyi Jin 225009797 Zibo Song 225008766

Design

There are three servers in this distributed system(lenss1, lenss3 and lenss4). In our desgin, we set lenss1 as a server contains a master process(shown as a red one), two master replicas and a connector(worker for connecting other workers running on other servers, shown as a black one). Lenss3 and lenss4 all have a primary worke(red one), a secondary(slave worker) and a connector(balck one).



1. Client to server

We define the addresses of master process(including master replicas) as the build-in knowlege for clients which is the “lenss-comp1.cse.tamu.edu:[portnumber]”.

Step1: When we start a client, it first gets connection with the available master process with calling stub function check() through a loop to choose the master process among three possible candidates

Step 2: .If current connecting process is the master process, it will reply a confirm message and the address of the primary worker process to client. There is a counter in the master process which preserve the number of clients connecting to each primary worker, as a result, master process always return the address of the primary worker which has less clients connecting.

Step 3: Once client receives the address of primary worker from master process, it will create channel towards primary worker and call the stub function login() to log in system. After logging into the system, Client can call stub functions(LIST, JOIN, LEAVE and CHAT) to do CRUD operations.

2.Server to Server

Data Propagate and Storage:

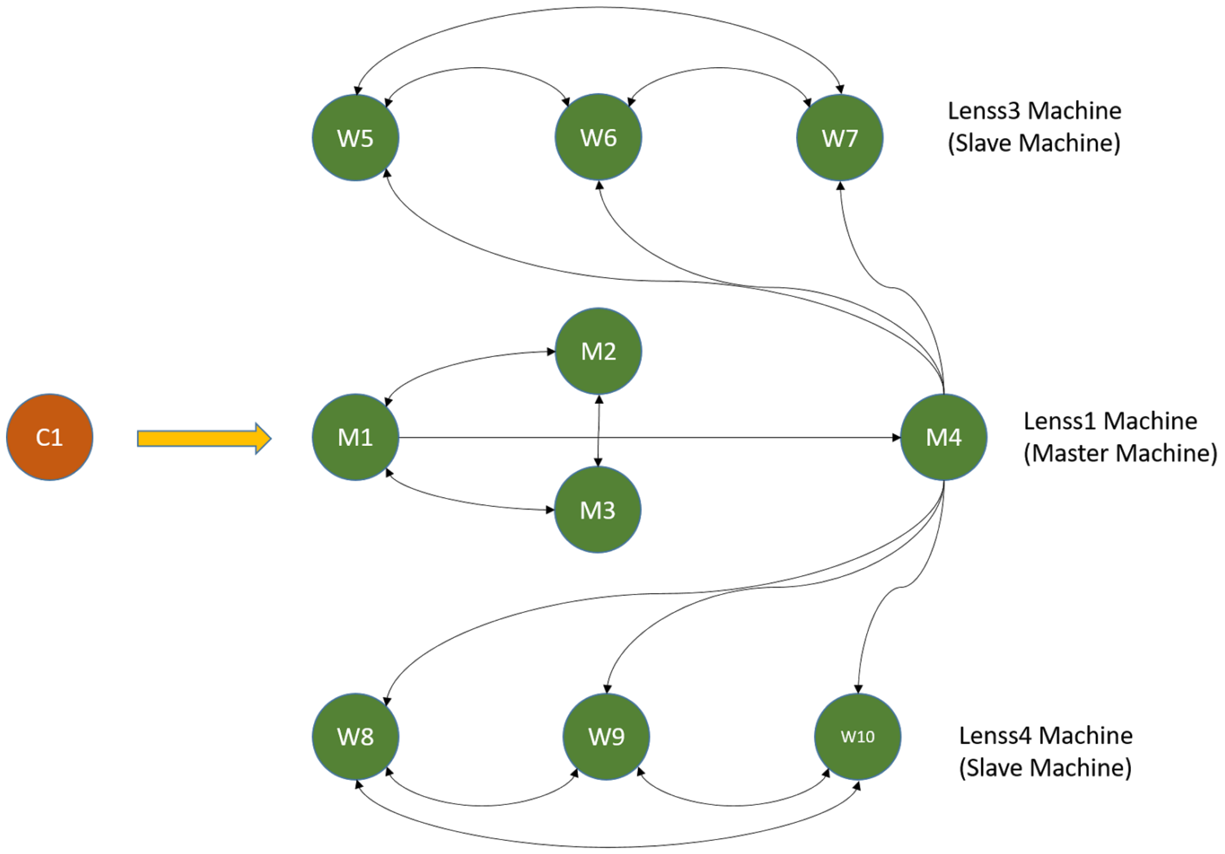
Basically we store our data in all three servers, including the file about the relation between users([servername] database.txt) and files for each user's posts record(also with the posts of users it has joined).

When a Process starts, it will read the relation file first and load it to its own memory. Each time when client send operation requests(which will update memory and database) to the primary worker, primary worker will update memory and local files then it sends request to the connector process on the master server and this connector process will propagate the request to another primary worker to update its memory and local file(including relation database and posts history), thus we can achieve the consensus on relation database and posts history.

Once the Primary Worker dead, the process chosen by election function would upload relation database again to update its memory since our system do not update the memory of slave process during propagate.

3. Heart Beat Detector

In this assignment, to achieve the required scalable and highly available twitter service, we designed a heart beat scheme, in which each process on the three machines are monitored by other processes in real time. In the presence of primary process failure, we let the other processes do election. We describe this complex scheme in the following graph.



The figure above basically shows how we implement the monitoring through establishing periodic heart-beat monitoring. In the 2 slave machines, Lenss3 and Lenss4, we employ 3 processes, respectively. They are monitored by each other upon created. In addition, they are also monitored by the M4, which is created on the Master machine. In contrast, the processes on the master machine are not setup to monitor each other, they are specified to monitor one or two processes within the same machine. This way, M4 was setup only to monitor processes on the slave machines, it serves as a connector (Actually, W7 and W10 also servers as connectors since they need to report to M4 that their machine was turned on).

In this case, the M1, W5, and W8 serves as primary workers, they aim to serve the client properly. A coming client first send a request to the M1, M1 is responsible to allocate a primary worker on the slave machine to further serve this client. Suppose Lenss3 machine is down, then M1 would detect that and only allocate newly coming client to Lenss4. If M1 is down due to some reason, M2 and M3 would detect that and they begin to cooperate to elect a leader. We adopt bully election and the criterion of election is the port number that each process is listening to. Also, if W5 is down, W6 and W7 would detect that and they cooperate to elect a leader to take replace of previous leader W5. Similar for slave machine Lenss4.

In the process of election, we directly let the secondary processes to cooperate to generate a new primary process by sending the port number that the specific process is listening to.