**Design Document**

In this document I have described the algorithm of my program and how I have implemented the voting protocols to perform data replications across the servers.

**Read Process**

Step 1

The client will randomly select a file to read the contents.

Step 2

Once the file is selected, then client connects to the servers in the quorum which is calculated by the hashing protocol.

Step 3

Now the client sends Read request message to any one of the 3 servers.

Step 4

If the message reaches the server successfully, it is queued up in the priority queue at the server based on the timestamp on the process.

Step 5

At the server, when the job is processed from the top of the queue a lock is put on the file so that no two concurrent process can access the same shared resource. This is done to maintain consistency across all the servers.

Step 6

Once the file is locked for that particular read process, it checks if the file is available and reads the contents of the file.

Step 7

After reading the contents, the lock is removed and also the job request is removed from the top of the priority queue.

Step 8

After successful read, the server returns successful acknowledgement to the client.

Step 9

In case of any failure in the channel (disruption), then message does not reach the server. To handle this the connection will timeout after a while resulting in not reading the file successfully.

**Write Process**

Step 1

The client will randomly select a file to perform write operation.

Step 2

Once the file is selected, then client connects to the 3 servers in the quorum which is calculated by the hashing protocol.

Step 3

Now the client sends Write request message to all of the 3 servers.

Step 4

If the message reaches the server successfully, it is queued up in the priority queue at the server based on the timestamp on the process.

Step 5

At the server, when the job is processed from the top of the queue a lock is put on the file so that no two concurrent process can access the same shared resource. This is done to maintain consistency across all the servers.

Step 6

Now to perform the write process, by putting the lock on the file and making the other process wait the server sends vote request to the other 2 servers in the quorum and waits for acknowledgement. This is part of the voting protocol implemented.

Step 7

Now the other servers receiving the request will queue up the process and when it’s processed, it checks if it had received the same request from the same client for writing into the same file. If true then the server sends back a “Successful acknowledgement” message back to the sending site thus by casting a vote towards the write process.

Step 8

If the receiving server has not received any write request from the client for a that file, then it sends a “No acknowledgement” message back to the sending site and does not contribute a vote towards writing happening at the other server.

Step 9

A server can perform the write process only if it has majority votes from the servers. That is the server needs vote from one more server to be part of majority servers (2 out of 3).

Step 10

If both the servers fails to acknowledge the vote request then no write happens across any of the sites as there is no majority servers in favor of the write so it does not happen across any of the sites. Thus consistency across the files is maintained.

Step 11

After receiving acknowledgments from the servers, based on the votes the write is performed in the server. After the job is complete the lock is released for the next job in the queue to be performed.

Step 12

Depending on the write process, the server sends back acknowledgment back to the client.

Step 13

The same process is carried out by all the clients which sends write requests concurrently.

Note:

In the program, all client to server messages is represented by msg\_id = 100 and all server to server messages is represented by msg\_id =200. Some client connections are randomly stopped at intervals to simulate channel disruption. Some client server channels will run smoothly to simulate how concurrent requests at the server is handles without deadlocks.