Architecture:

The network information of would be stored in an input/config file, that would contain the neighbor information. Upon the instantiation of each Process class this file would be read to know which processes exists. Each Process would contain the information about its neighbors in an Hashtable.

Each entry in the Hashtable would have key as the process\_id and value as an object of class Neighbor which encapsulates the IP\_address and port\_no.

In each process, there would be two threads that would be sending the broadcast message and a receiving thread that would be reading the message.

Each Process has a queue which would arrange each message in order of the timestamp of the message.

When a message arrives it would be written in a log file which each Process will have access to. In addition to this when a process sends a message it will also write that message in to another output log file.

Each process has a logical\_clock variable that would implement lamports\_logical clock.

The sending thread will do the following:

* It will send an undeliverable message to the destination process.
* Upon receiving the maximum proposing timestamp from all its recipients. The process will update its timestamp to the maximum timestamp.
* It will then send the final message to its destination.

The receiving thread will do the following:

* Upon getting all the Initiator messages the receiver would send the maximum proposing timestamp to its sender.
* Upon receiving the final message the thread will put it in the queue final processing.

Once all the final messages are put in the queue it will be dequeue to be read by the process.

Message formats:

Sent from Initiator which is marked as undeliverable : @INITIATOR <src\_process\_id> <dest\_process\_id> <message\_id> <message> <timestamp>

Proposed Timestamp : @PROPOSAL <src\_process\_id> <dest\_process\_id> <proposed\_timestamp>

Final Deliverable message : @DELIVERABLE <src\_process\_id> <dest\_process\_id> <message\_id> <message> <final\_timestamp>

Testing Strategy :

At the end of the program execution the received message file for each process will be read by the test script. Each deliverable message timestamp would be checked to ensure total ordering. If a message with smaller timestamp(Ts2) occurs after a timestamp(Ts1). That Is if Ts1 > Ts2 then total ordering is not ensured.