CS 6847 : Cloud Computing
Assignment 3 : Exactly Once Semantics in Java Socket Program

--Mahesh Gupta (CS12M018)

---

1. **Assumptions**
   - There is one to one communication between client and server i.e. Client does not sends another message if the response of previous message is not received.
   - There are no power failures.

2. **Failure Scenario**

   For Client Server Communication, 2 types of failures have been handled
   - When client sends request and goes down before receiving the response
   - When client sends the request and server processes it and server goes down before sending the response.

3. **Failure Handling**
   - When client request is received at the server side, server stores it on its side in the map as unprocessed request. Both the receiving and writing to map is an atomic operation.
   - Later, request is processed and response is stored in the map. Atomically request is removing from unhandled response map.
   - Response is sent back to the client and if the response is received successfully then it is removed form the response map.
   - If the response does not reaches client in first turn then it is sent 3 times.

4. **Implementation**
   - Each client is tied to a particular port. The port number can be configured using external file for each client.
   - The **ip:port** value of a client is used as a key on the server side to differentiate two clients from each other.
   - When the connection request is received it is checked if any unprocessed request from client is present or not. If it is, server sends a message about unreceived response to the client then sends the message and removes message from the map.
   - Later server receives the request from client and as mentioned above handles the request.
   - For atomically executing multiple statements, **synchronized** keyword has been used.
   - Each client is handled by separate thread so as to allow multiple clients at the same time. Hence the central map used is **ConcurrentHashMap** data structure.
   - For persistent storage, each time the map is modified, it is stored back to the file again. Writing to file is an atomic operation to avoid clashes.

5. **Issues**
   - Each time when the map is changed it is written back to the file and hence the solution is not scalable.
   - Use of locks in the code avoids the performance to slow down if multiple clients are present.

6. **Conclusion**
   - Implementation of globally exactly once semantics is pratically not a feasible solution.
   - Hence taken some suitable assumptions the implementaiton can be done with reasonable scalability.