# Design Consideration

**Yukun Jiang (yukunj)**

## How does Coordinator and Participants communicate?

To simplify the communication, there are only two kinds of message to be communcated: `CoordinatorMsg` from Coordinator to Participant, which is either a Phase I proposal or Phase II decision about a transaction. `ParticipantMsg` from Participant to Coordinator, which is either a Phase I vote or Phase II ACK.

This simplifies the communication process, since we are sure the type of object from the `byte[]` stream from the messaging interface and could safely cast it.

## How to do Write-Ahead Logging?

To safely recover from failure and stick to previous commitment/resources allocation, both the Coordinator and Participant use write-ahead logging. Essentially each one will have a big mapping containing all the transaction records so far.

On the Coordinator side, the record memorize the unique id for this transaction, what resources and participants are involved and current status of this transaction. Whenever the transaction state is to change (a new Proposal or from PREPARE to ABORT/COMMIT), the log must be persistently flushed out to disk.

On the Participant side, the record will memorize the previous votes to any transaction and ongoing locked resources to a halfway transactions, so that upon failure recovery the Participant will not mess up previous commitment or locked resources.

## How to deal with Failure-Recovery?

Firstly, I make the Participant class simple and idempotent. Upon recovery, it just need to load the persistent logging from disk and answer any questioning messages from the Coordinator. The knowledge in the logging is enough.

Secondly, the recovery process on the Coordinator is not that easy. Upon recovery, it will reload the persistent logging and check the status of each transaction on book. If a transaction is recorded as in Phase I PREPARE, the Coordinator will simple abort this transaction and broadcast this decision to all involved Participants. If a transaction is already Abort/Commit, this decision must be respected and Coordinator will re-broadcast this decision to all involved Participants and wait until they are all acknowledged.

## How to deal with lost messages?

As I mentioned above, my Participant is simple and idempotent. Therefore, the querying timeout process is done on the Coordinator.

There is a messaging queue on the Coordiator that records every outbound messages and when it's sent out. As specified in the writeup, 6 seconds is the timeout threshold. Every second, the Coordinator will check if there is any timeout message in the queue.

If a timeout message is Phase I PREPARE message, the Coordinator will implicitly think this is an DENIAL reply and immediately abort this transaction and broadcast such decision.

If a timeout message is Phase II ABORT/COMMIT message, this result must be respected. Hence the Coordinator will resend this message to its original destination, until it's fully acknowledged.

Unpon a transaction's state changes, for example if it moves from Phase I PREPARE to Phase II ABORT/COMMIT, or move to END state, we will prune this outbound message queue to remove the recording of message associated with this transaction id, since we already have the needed information about this transaction, and thus we don't need to demand any response from involved Participants.

## Potential Weird Scenarios

There are some potentially weird scenarios, if we are handling node failure/message loss.

1. Node never receive a PREPARE message, but directly receive a DECISION message about a transaction. This mostly happens when the Node itself has gone down before, and thus Coordinator has deemed the implicit response as DENIAL and abort this transaction. In this case, the Participant will just remeber and write down the result in LOG.
2. Node repeatly receives PREPARE or DECISION messages from Coordinator, even if it has already heard of this message. This mostly happens when Coordinator has down and reboot back to continue previous transactions. In this cases, the Participant should just echo back the vote/ACK as written down in the LOG.
3. Coordinator received Phase I reply for a transaction, even if it has moved to Phase II stage of that transaction. It could happen if the Participant's message is delayed and Coordinator has implicitly aborted this transaction. In this case, the Coordinator should resend the decision of this transaction back to the Participant.