2PCMaster

- Implemented TPCRegistrationHandler and started it in a separate thread
    - If its not in a separate thread, then they cannot handle restarts of slaves after failure (there won't be anything listening for re-registration)
    - Keep track of slaveIDs and their liveness status (register => live; failure in connection form master => dead). Note, this will be required for handling re-registering etc.
- Implements consistent hashing
- SlaveInfo implements Comparable properly (will be required to maintain/sort the DHT/Chord-like ring)
- Sends PUTs and DELs through the same lock (serialized 2PC operations)
- Sends GETs outside the 2PC lock
- Implements an inner-class for the threadpool in handle() method of NetworkHandler in KVClientHandler
- KVMessage extended with the new fields necessary from 2PC
- 2PC Master **MUST** block until all slaves are up.
    - Requests that arrive in the meantime will be queued
    - Assume, there won't be so many requests so that the queue will blow up

2PCMasterHandler

- Implements an inner-class for the threadpool in handle() method of NetworkHandler
- Handles GET
- Handles PUT/DEL
- Writes to log after each state change (more on logging later)
- Sends back ACK to the Master

2PCLog

- Loads the log from disk
- Implements the recovery state machine properly

WHAT TO LOG

- The actual PUT/DEL requests in the first phase (so that they can rebuild in 2PCLog)
- The global decision in the second phase

WHEN TO LOG

- Log before sending back local decision (in the first phase) or ACK (in the 2nd phase)
    - In the second phase, if the slave does not have any outstanding/interrupted 2PC operation, and it receives a global decision again from a master, it should reply with an ACK even though this decision may be redundant for this slave
    - This is VERY important to ensure proper recovery from failure without having to talk to the master (the reason has been explained in a mail between Edward and Mosharaf)
- Logging after sending back the ACK is also fine, but in that case the slave will have to actively contact the master after restarting (This is harder to do and is not provided for in the skeleton. Encourage them to keep it simple and have the master retry until the slave ACKs).

ignoreNext

- If received by the master, forward to ALL slave servers **[Not necessary. Can ignore if implemented.]**
- If received by the slave server, fail the next 2PC operation in the first phase
- ignoreNext is for their debugging/testing only; NO need to worry about pathological cases (e.g., if ignoreNext is forwarded to all slaves, then more than one request will be ignored, not just exactly the next)

FAILURES

- Failures first phase should result in an abort; Failures in the second phase **should** be handled.
- Failures in 2nd phase, must not result in data loss or inconsistency;
    - They must perform logging properly (see WHAT and WHEN above) to ensure this