CS 162 – Worksheet 10

True/False

1. For high availability, we want a small MTTR / (MTBF + MTTR). True

2. Just like 99.99% available has 10x less outage time than 99.9% available, 88.88% is ten times more available than 88.8% available. False

Short answer

1. What does two-phase commit allow us to guarantee? Why does it not violate the Generals' Paradox?

   Two-phase commit allows us to guarantee that all participants will agree to either commit or not commit some operation; eventually everyone will do the same thing. This does not violate the Generals' Paradox because we make no guarantee about *when* everyone completes the operation.

Longer answer

Imagine a system with three worker nodes and a master (a bit of a toy system, but let's ignore that). A client connects and performs an action which triggers a two-phase commit.

The one-way latency between the client and the master is 100ms. The latency between the master and each of the slaves is 10ms. Let's imagine it takes each slave 20ms to update its log and/or commit, and all other processing/transmission time is negligible.

For each scenario, please answer the amount of time that passes from when the client first sends its request to the time the slaves receive the final GLOBAL_COMMIT or GLOBAL_ABORT message.

2. Every transmission occurs correctly (and no nodes fail).

   100ms (client send)
   + 10ms (VOTE_REQ)
   + 20ms (workers advance to READY state, write to log, send VOTE_COMMIT)
   + 10ms (master receive VOTE_COMMIT messages)
   + 10ms (master send GLOBAL_COMMIT messages)
   150ms

3. One worker node disconnects and misses the VOTE_REQ, and does not reconnect fast enough to stop the master from timing out. The master times out in 5s. The worker comes back online just as the master times out, for simplicity's sake.

   100ms (client send)
   + 5000ms (master timeout)
   + 10ms (GLOBAL_ABORT)
   5110ms, or 5.11s

   Keep in mind that any delay due to network or logging on the slaves is unimportant; the server's timeout is triggered without needing to know what the RTT or other delays are.

4. The master crashes while in the READY state (i.e. the workers have voted to commit and are also in the READY state). The master fully reboots in 30s. Keep in mind that (as in lecture 19, slide 52) the master will send a GLOBAL_ABORT once it recovers in this case.

   Assume the master crashes immediately after successfully sending the VOTE_REQ.

   100ms (client request)
   + 10ms (VOTE_REQ)
   + 30,000ms (master crash/reboot)
   + 10ms (GLOBAL_ABORT)
   30,120ms, or 30.12s