CPS 512/590 second midterm exam, 11/3/2015

Your name please: ____

/40
/40
/40
/30
/30
/20
/200

Part 1. Spanner

These questions pertain to multi-version transactions as used in the Spanner system. Answer each question with a phrase or two or maybe a sentence or two. [40 points]

a) How does Spanner enforce serializability of update (read/write) transactions?

b) Can two Spanner transactions commit with the same timestamp?

c) Spanner is a key/value store. Why is it necessary for each Spanner replica group to use a Replicated State Machine (RSM) model, which imposes an order on write requests even if they affect different keys?

d) A Spanner group leader may force a transaction to abort. Why / under what circumstances?

Part 2. Déjà vu: Chubby revisited

These questions are similar to questions on the first midterm. Answer each question with a few phrases or maybe a sentence or two. [40 points]

a) In Chubby (and in Lab #2), applications acquire and release locks with explicit **acquire()** and **release()** primitives. Leases create the possibility that an application thread holding a lock (the lock holder) loses the lock before releasing it. Under what conditions can this occur?

b) In this circumstance (a), how does a Chubby lock client determine that the lease has been lost? How does the application determine that the lock has been lost?

c) Chubby maintains a single lease for each client's session with Chubby, rather than a lease for each individual lock that a client might hold. What advantages/benefits does this choice offer for Chubby and its clients?

d) Given that Chubby can complete a write after agreement from a subset of replicas, how do the other replicas learn of the write under normal failure-free operation?

Part 3. Causality

These questions pertain to the concepts employed in the Bayou system. Please keep your answers tight. [40 points]

a) Give an example of two vector clocks v_1 and v_2 in which $v_1 < v_2$ (v_1 **happened-before** v_2).

b) Give an example of two vector clocks v_1 and v_2 in which v_1 is **concurrent** with v_2.

c) Give hypothetical real-world example in which a client learns of updates in an order that does not respect potential causality, i.e., that violates causal ordering. Why is this bad?

d) Bayou clients may learn of updates in different orders, yet these orders always respect causality. How does Bayou assure this property? I can think of several acceptable one-sentence answers. Please try to answer in one carefully worded sentence!

Part 4. Consensus

These questions pertain to **consensus** algorithms (Viewstamped Replication (VR), Raft, and Paxos). You may answer with reference to any variant of **consensus**. Please keep your answers tight. [30 points]

a) Where is **consensus** on the CAP triangle? How does **consensus** behave under a network partition?

b) A fundamental property of **consensus** is that each view (also called term or ballot) has at most one primary (also called a leader). Is it possible for a view to have no primary? How could this occur?

c) Unnecessary view changes (or elections/scouts) can prevent **consensus** from making progress. Why? Propose a mechanism to prevent unnecessary view changes in practice.

Part 5. A little more thought [30 points]

Consider an execution of **consensus** with five replicas, in which a leader/primary proposes (prepares) values for a sequence of log entries (also called op-numbers, slots, or indexes). Consider now what happens over a sequence of views, each with a different primary (we may number them P_1, P_2, P_3,…). Give an example in which a primary proposes a value *v* that overwrites a different value *w* that an earlier primary proposed for the same slot, even though the later primary was aware of *w*, i.e., *w* was accepted by at least one member of its voting majority.

Part 6. Mashing it all together [20 points]

We might think of Bayou's approach as "eventual consensus": like eventual consistency the system always makes progress as long as any replicas are reachable and/or can communicate, and like **consensus**, all replicas eventually agree on the same sequence of updates (presuming that all failures eventually resolve). To achieve these properties, Bayou must make stronger assumptions about the application than **consensus** does. Compare/contrast their assumptions about the application.