/60
/50
/50
/60
/20
/60
/300

CPS 512/590 final exam, 12/8/2015

Your name please: ____

Part 1. 2P*

These questions pertain to transactions with two-phase locking (2PL) and two-phase commit (2PC). [60 points]

a) Suppose that transaction T is to execute the following sequence of operations: [R(x), W(y), W(z)], where x, y, and z are data items, and each item is protected by a separate lock. Show a sequence of acquire/release actions and read/write operations that is well-formed for T under two-phase locking (2PL).

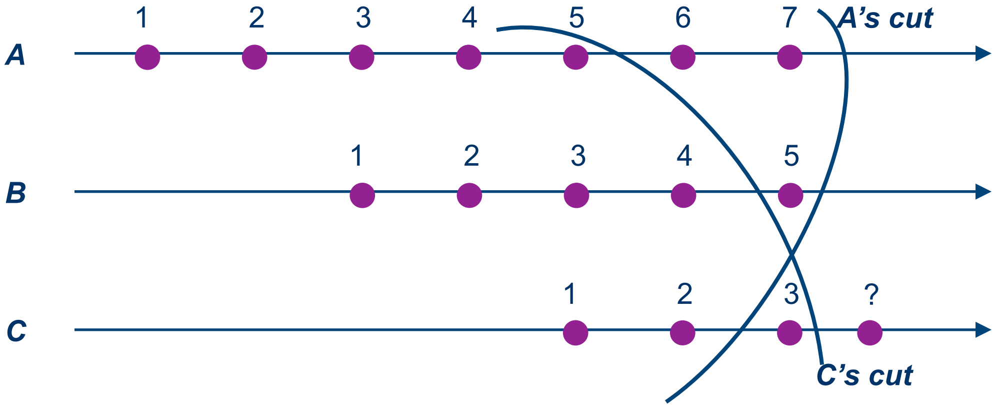
b) In Lab #3 (transactions with 2PL), was deadlock a problem for your solution? Briefly describe how your solution deals with deadlock. (Please try to keep it to one or two sentences.)

c) Spanner does not use locking for transactions that are declared **readonly**. Why? What benefits result from this choice? (Note: I am not asking you to explain how Spanner ensures serializable transactions without these locks.)

d) Spanner uses two-phase commit (2PC) to commit transactions (in most cases). How does Spanner avoid the blocking conditions that could result from coordinator failure in 2PC? Can a commit in Spanner ever be forced to block as a result of a failure? (If so, please describe under what conditions this can occur.)

Part 2. Bayou, vector clocks, and causality

These questions pertain to write propagation with anti-entropy exchanges in Bayou. Consider the following timeline example for three Bayou nodes/replicas (A, B, C), showing numbered updates/writes originating at each replica. The two curved lines illustrate the set of updates known to A and the set of updates known to C at a given point in time: each of these sets is a *consistent cut*. Answer the questions below. [50 points]



a) What is a *consistent cut*? Write down a quick definition. **Extra credit**: state the consistency property as a logic rule.

b) Write down the vector clocks corresponding to A's cut and C's cut in this example.

c) Now suppose node A performs a Bayou anti-entropy transfer to node C. List the accept stamps of the updates that are transmitted from A to C during this anti-entropy exchange.

d) Modify the picture to show any new causal (*happened-before*) relationships resulting from the transfer in (c).

e) What is impact of the transfer in (c) on node C's logical clock? Modify the picture to number the next update to be generated at C after the transfer.

Part 3. Digital signatures and certificates

These questions pertain to signed certificates in various protocols we have studied. [50 points]

a) What is a digital signature? Draw a small picture showing what cryptographic operations produce a digital signature, and indicate the sizes of the inputs and outputs.

b) In what sense do digital signatures offer stronger safety properties than “real world” signatures made with pen and ink? [Presuming that the signatures in question are always authentic and unforged.]

c) What is in an SSL certificate (also called an X.509 identity certificate)? List some important items/elements/parts. How is a SAFE certificate different from an SSL certificate with respect to this list of parts?

d) What is in a DNSSEC name lookup response? List some important items/elements/parts.

e) Liskov’s practical Byzantine consensus protocol (PBFT) uses digital signatures in a key step: PREPARE messages are signed by their senders, and nodes pass recently received PREPARE messages to the new primary on a view change. Why (under what conditions) is it important that these PREPARE messages are signed?

Part 4. Authentication and secure naming

These questions pertain to authentication protocols and the BAN “Logic of Authentication”. [60 points]

a) The Kerberos protocol (like the Needham-Schroeder protocol on which it is based) avoids the use of digital signatures. What alternative does Kerberos use to protect the integrity of messages sent in the protocol? What are the pros and cons of this choice?

b) Kerberos requires synchronized clocks because timestamp values occur in multiple places in the protocol. Briefly summarize how and why timestamp values are used in Kerberos: specifically, what purpose(s) do they serve?

c) Kerberos, TLS/SSL (e.g., HTTPS), and single-sign on (SSO) protocols like Shibboleth are all designed to “convince their participants of the truth of some formula X” (where X is a logical statement, as stated in the BAN paper). Summarize the statement or property X believed by the participants at the conclusion of these protocols, i.e., what is the outcome of these protocols?

d) Is DNSSEC redundant with the PKI/SSL certificates issued by Certifying Authorities? If we have one of them, does the other provide additional protection? Discuss. Be sure to note what each provides.

Part 5. Bitcoin

These questions pertain to the Bitcoin crypto-currency and the cryptographic techniques it uses. Please keep it short. [20 points]

a) How does a user get an account in Bitcoin?

b) A Bitcoin user spends money from an account by issuing a transaction record and broadcasting it on the Bitcoin network. How do the validators who receive the transaction record verify that it is issued by the account owner?

Extra space for earlier parts, e.g., 4d.

Part 6. Consensus

These questions pertain to consensus algorithms under various models, including classical **consensus** (VR-Paxos variants or RAFT), Byzantine consensus (BFT), and the “Nakamoto consensus” used in Bitcoin. Please keep your answers tight. [60 points]

a) Suppose that during a view change in classical **consensus**, the new leader/primary P learns that one replica R has voted for a value *v* for slot *s* in a previous view, and no other replica reports a previous vote for *s*. (A slot is also called in op-number or index.)

- Is it possible that *v* has committed for *s*?
- Is it possible that *v* has not committed for *s*?
- Is it possible that some other value *w* has committed for *s*?
- What should P propose for *s* in the new view?
- Suppose now that another replica R_2 also reports voting for a value *w* for *s*. What should P propose for *s* in the new view?

b) Each of these consensus algorithms (**consensus**, BFT, Nakamoto) assumes that at most *f* failures occur, i.e., the number of faulty participants is bounded as a function of the number of participants *N*. For each of them, summarize what might happen if more than *f* participants are faulty.

- **Consensus** (e.g., VR or Paxos):
- Byzantine:
- Nakamoto:

c) The safety of BFT is based on fragile assumptions about how Byzantine failures occur, so there is some doubt about what protection it offers in practice, e.g., for a replicated server program. However, the public has bet billions of dollars on the safety of Nakamoto consensus in Bitcoin. Discuss. Do you think these consensus protocols are safe? Why or why not?