Homework 4

CS425/ECE428 Spring 2023

Due: Monday, April 10 at 11:59 p.m.

1. **Transaction Processing**

   A bank uses a transaction processing system that complies with ACID properties. Within each transaction, a user can issue one or more of the following operations: (i) `DEPOSIT <account> <amount>` which deposits the specified amount into the specified account, (ii) `WITHDRAW <account> <amount>` which withdraws the specified amount from the specified account, and (iii) `BALANCE <account>` which immediately displays the current balance in the specified account (also including the effects of operations previously executed within the same transaction). As a consistency check, if at the *end of a transaction*, any account has a negative balance, the system aborts that transaction.

   Consider the five transactions shown below that are executed serially (one after another) in order T1, T2, T3, T4, T5. Answer the following questions, assuming all accounts referred in the transactions have a balance of zero before T1 is executed.

   ```
   T1:  DEPOSIT A 20; DEPOSIT B 50; DEPOSIT C 40
   T2:  DEPOSIT A 30; WITHDRAW B 60; WITHDRAW C 10; BALANCE C
   T3:  DEPOSIT A 10; WITHDRAW B 40; WITHDRAW C 10; BALANCE B
   T4:  WITHDRAW A 50; DEPOSIT B 10; WITHDRAW C 20
   T5:  BALANCE A; BALANCE B; BALANCE C
   ```

   (a) (5 points) For each transaction, state whether it gets committed or aborted and why.

   (b) (5 points) What will be the result displayed by each of the `BALANCE` operations invoked in the transactions?

2. **Concurrency: Two-phase locking, Deadlocks, and Timestamped Ordering**

   Consider the following two transactions, each with five operations:

	$T1$	$T2$
1	write E	write A
2	read D	write B
3	read B	write C
4	read A	read B
5	write C	read D

   (a) (2 points) Write down all the conflicting pairs of operations across the two transactions. (You can refer to each operation as $Tn.m$; e.g., $T2.1$ is "write A", $T2.2$ is "write B", and so on).

   (b) (3 points) Is the following interleaving of operations across $T1$ and $T2$ serially equivalent? Explain why or why not.

$T1$	$T2$
	write A
write E	
read D	
read B	
	write B
	write C
	read B
read A	
	read D
write C	

(c) (4 points) Is there a *non-serial* interleaving of operations across $T1$ and $T2$, that could result from using strict two-phase locking (with read-write locks), and is equivalent in effect to a serial execution of $T1$ followed by $T2$? If yes, provide an example. If not, explain why.

(d) (4 points) Is there a *non-serial* interleaving of operations across $T1$ and $T2$, that could result from using strict two-phase locking (with read-write locks), and is equivalent in effect to a serial execution of $T2$ followed by $T1$? If yes, provide an example. If not, explain why.

(e) (4 points) Write down a partial interleaving of the operations across $T1$ and $T2$ that is compliant with strict two-phase locking (with read-write locks) and leads to a deadlock. List which lock (and in which mode – read or write) will be waited upon by each transaction in your deadlock.

(f) (4 points) Write down an interleaving of the operations across $T1$ and $T2$ that is serially equivalent, but impossible with strict two-phase locking. Explain what makes the interleaving impossible with strict two-phase locking.

(g) (4 points) Is there a *non-serial* interleaving of operations across $T1$ and $T2$, that could result from using *timestamped ordering*, and is equivalent in effect to a serial execution of $T1$ followed by $T2$ (without any of these transaction getting aborted)? If not, explain why. If yes, provide an example interleaving and indicate the relevant state maintained by the objects for timestamped ordering. You can use the example format like "write X (X.committedTS = 1, X.RTS = [1,2], X.TW=[2])" to indicate the state maintained by the object (i.e. timestamps for reads and tentative writes) after an operation has been executed.

3. **Two-Phase Commit**

Figure 1 shows a system of three servers processing a distributed transaction. Server 1 is the coordinator and interacts with the client. The network delay between the client and the coordinator, and among the three servers is indicated in the figure.

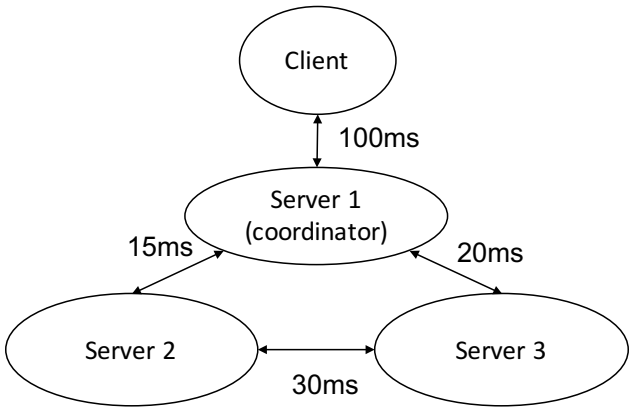

Figure 1: Figure for question 3

Any local processing at a server or self-messages take negligible time. The client issues a COMMIT request for its transaction at time t=0s. Assuming no messages are lost, no server crashes, and no server wishes to abort the transaction. Answer the following questions:

(a) (3 points) When will each of the three servers locally commit the transaction?

(b) (2 points) What is the earliest time at which the coordinator can safely send a message to the client stating that the transaction will be successfully committed?