CprE 450/550: Distributed Systems and Middleware Spring 2015

Take-home Final Exam

Handout: April 27, 2015
Due: May 6, 2015, 11:59pm

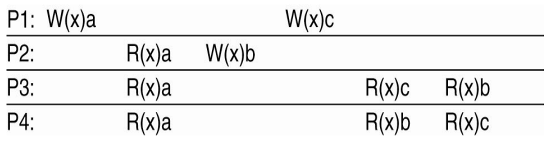
Student Name:

Please type your answers and submit it through Blackboard System before May 6, 2014, Midnight.

1. (5 points) In the two-phase atomic commit protocol, each client has an uncertainty period after it has voted yes for a request (transaction) and is waiting for the final decision (commit or abort) from the coordinator. Can the client time out and unilaterally abort the transaction? Explain.

2. (5 points) Given the read and write operations shown in the following Figure (a) and (b), please answer wether they are (a) sequentially-consistent, (b) causally-consistent, and (c) FIFO consistent. Please explain your answer.

(a)



P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

(b)

3. (10 points) What happens during each of the following operations in the following execution schedule of four transactions T_1, T_2, T_3 and T_4 accessing the three items A, B, and C? The concurrency control mechanism is time-stamp based. (Simply note *ok* if nothing special happens during the execution of a

particular operation.)

	T_1	T_2	T_3	T_4
	250	200	210	275
(1)		READ A		
(2)			READ A	
(3)		WRITE B		
(4)				WRITE A
(5)	READ B			
(6)				READ B
(7)	READ A			
(8)	WRITE C			
(9)				WRITE A

What are the final read and write times for A, B, and C (using the timestamps of the transactions in the Table, like 250,200, etc.)?

4. (5 points) In the release consistency model, acquire and release operations only need to be processor consistent. Why is this a sufficient condition for the shared variables to be FIFO consistent? What are the benefits of using release consistency as compared to sequential consistency?

5. (10 points) Assume you have the following transactions T_1 and T_2:

   T_1: **begin transaction**, 1: Read A, 2: Write A+1 to B, **end transaction**

   T_2: **begin transaction**, 3: Read B, 4: Write B+1 to A, **end transaction**

   Both A and B have an initial value of 1. Read operations require a *shared lock*, while Write operations require an *exclusive lock*. All locks are released before the transaction can end. We say that *legal schedules* do not violate the lock operations.

   Below you find a table with entries for all possible schedules of the four operations in the two transactions. Complete the entries in the table as follows:

   - Mark whether the schedule is legal or not.
   - Mark whether the schedule is serializable or not.
   - Fill in the result of A and B after execution of the given schedule.
   - Mark whether Two-Phase-Locking succeeds or aborts.
   - Assuming that T_1 starts with timestamp smaller than T_2, describe whether the timestamp-based concurrency control algorithm succeeds without rollbacks. If not, describe which transaction needs to be rolled back.

schedule	legal? (Y/N)	serializable? (Y/N)	value for A	value for B	2 PL successful? (Y/N)	result for timestamp ("successful" or describe what happens)
1,2,3,4						
3,4,1,2						
1,3,4,2						
3,1,2,4						
1,3,2,4						
3,1,4,2						

6. (10 points) Transactions T and U execute on a single server. In the table below, time proceeds from top to bottom, and relative position of operations indicates the relative order in which they were performed.

   State whether the execution below is possible in each of the following cases. If you answer no, provide a brief explanation. With read-write locking, the transactions acquire read lock when that is adequate.

	Transaction *T*	Transaction *U*
(1)	a=read(y)	
(2)	b=read(x)	
(3)		write(z,1)
(4)		d=read(x)
(5)		f=read(z)
(6)		write(y,3)
(7)		commit
(8)	c=read(z)	
(9)	write(w,2)	
(10)	commit	

   (a) exclusive locks are used with strict two phase locking, and all locks required by a transaction are acquired at the start of the transaction.

   (b) exclusive locks are used with non-strict two phase locking.

   (c) read-write locks are used with strict two phase locking, and all locks required by a transaction are acquired at the start of the transaction.

   Note: A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data. An exclusive lock on data prevents other transactions from accessing the data.

7. (5 points) In Message 2 of the Needham-Schroeder authentication protocol, the ticket is encrypted with the secret key shared between Alice and the KDC. Is this encryption necessary? Please explain.

8. (5 points) Please do a literature survey on TCB, TPM, and SGX, in particular, what will we be able to do with SGX in secure distributed applications or systems? If possible, please use some examples.

9. (5 points) Please do a literature survey on Software-Defined Networks (SDN) and its implications/changes to the implementation of current popular distributed applications or systems? Advantages or new issues brought by SDN? It doesn't have to be comprehensive. A case study is fine.