15-440/15-640: Homework 2

Due: October 13, 2016 10:30am (**NO LATE DAY**)

Name:
Andrew ID:

1 Concurrency Control

1. Is 2-phase commit blocking or non-blocking? What about 3-phase commit? Elaborate on your answers by explaining the situations where a transaction blocks or why it doesn't block.

2 Distributed Mutual Exclusion

1. In class we discussed enforcing mutual exclusion, among other ways, via a central server, majority voting, and token ring.
   (a) How many messages are required per request under heavy contention? What are the messages used for? Write your answer for each algorithm.
   (b) List at least one disadvantage of each algorithm.
   (c) Which of these systems is most robust to failure? Why?

2. Consider three processes. The system has totally ordered clocks by breaking ties by process ID. It uses the Ricard & Agrawala algorithm. The timestamp for each process of id i is $T(p) = 10 * L(p) + i$, where $L(p)$ is a regular Lamport clock.

   Each message takes 2 "real-time" steps to get delivered. Critical section takes 2 real-time steps. Fill in the table with the messages that are being broadcast, sent, or received between the processes until all nodes have executed their critical sections. Write "execute critical section" as the action for a node when it enters its critical section. The first three rows have been filled in for you, and the fourth row has been started. Assume that if a process receives messages from the other two processes at the same time, the message that comes from the lower process ID will be received first.

   Action Types: Broadcast (B), Receive (R), Send (S), Execute Critical Section (ExCS) Initial timestamps: P1 → 111, P2 → 212 and P3 → 103

Real Time	Process	Lamport Time	Action(to/from)	Contents	Q at P1	Q at P2	Q at P3
1	1	121	B	(request 121)	121		
2	2	222	B	(request 222)	121	222	113
	3	113	B	(request 113)			
3	2	232	R from 1	(request 121)	121	121	113
	3	133	R from 1	(request 121)		222	121
4	1	231	R from 2	(request 222)	113	113	113
	1	241	R from 3	(request 113)	121	222	121
	2	242	S to 1	(reply 121)	222		222
	2	252	R from 3	(request 113)			
	3	233	R from 2	(request 222)			

3 Logging and Crash Recovery

1. The ARIES logging and crash recovery design we talked about includes REDO information in its log. What is the rationale for this? (i.e., what functionality is enabled by including this information?)

4 Distributed Replication/Paxos

1. You have set up a fault-tolerant banking service for the PNC bank (Paxos National Corporation bank). Based upon an examination of other systems, you've decided that the best way to do so is to use the Paxos algorithm to replicate log entries across three servers, and let one of your employees handle the issue of recovering from a failure using the log.

   The state on the replicas consists of a list of all bank account mutation operations that have been made, each with a unique request ID to prevent retransmitted requests, listed in the order they were committed.

   Assume that the replicas execute Paxos for every operation.[1] Each value that the servers agree on looks like "account action 555 transfers $1,000,000 from Yuvraj A. to Srini S.". When a server receives a request, it looks at its state to find the next unused action number, and uses Paxos to propose that value for the number to use.

   The three servers are S1, S2, and S3.

   At the same time:

   - S1 receives a request to withdraw $500 from Yuvraj.
     - S1 picks proposal number 501 (the n in Paxos)
   - S2 receives a request to transfer $500 from Yuvraj to Srini.
     - S2 picks proposal number 502

   Both servers look at their lists of applied account actions and decide that the next action number is 15. So both start executing Paxos as a leader for action 15.

   Each sequence below shows one initial sequence of messages of a possible execution of Paxos when both S1 and S2 are acting as the leader. Each message shown is received successfully. The messages are sent one by one in the indicated order. No other messages are sent until after the last message shown is received.

   Answer three questions about the final outcomes that could result from each of these sequences:

   - Is it possible for the servers to agree on the withdrawal as entry 15?
   - Is it possible for the servers to agree on the transfer as entry 15?
   - For each of these outcomes, explain how it either could occur or how Paxos prevents it.

   Sequence 1:

```
S1 -> S1 PREPARE(501)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(501)
```

[1] In practice, most systems use Paxos to elect a primary and let it have a lease on the operations for a while, but that adds complexity to the homework problem.

```
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(501)
S3 -> S1 RESPONSE(nil, nil)

S2 -> S1 PREPARE(502)
S2 -> S2 PREPARE(502)
S2 -> S3 PREPARE(502)
... the rest of the Paxos messages.
```

Sequence 2:

```
S1 -> S1 PREPARE(501)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(501)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(501)
S3 -> S1 RESPONSE(nil, nil)

S1 -> S3 ACCEPT(501, ‘‘withdraw...’’)

S2 -> S1 PREPARE(502)
S2 -> S2 PREPARE(502)
S2 -> S3 PREPARE(502)
... the rest of the Paxos messages.
```

5 GFS/HDFS/Spanner

1. Short answer questions (For true/false : Write a small explanation for your answer.)

   (a) GFS and HDFS, by default, triplicate each data block on three different nodes. How many node failures can it tolerate (at least)?

   (b) If we decrease the default block size, how could this change affect the master node?

   (c) In GFS, how does the client fetch the data it needs to read? (Explain in 2 statements)

   (d) GFS is designed to better handle files of smaller sizes. (True/False)