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. (8 points)

   2-phase commit is blocking, 3-phase commit is non-blocking.
   In 2PC, when the coordinator and a member crashes, the new coordinator cannot decide which state the crashed member was in and thus cannot proceed until the crashed member recovers. Therefore 2PC is blocking.
   In 3PC, the coordinator can only COMMIT after collecting all PRECOMMIT ACKs from members, such that all members are awared of the decisions of other members in the COMMIT stage. Hence 3PC is non-blocking.

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 under one cycle? What are the messages used for? Write your answer for each algorithm.(6 points)

   Under heavy contention,
   Token Ring: One message, assuming every node has requests for executing critical section. The message is used to pass the token (from previous node who just exited critical section to next node).
   Central Server: 3 messages: one request message for entering critical section, one grant message from coordinator to give permission, and one release message to notify the coordinator after exiting critical section.
   Majority Voting: 3mk messages. m is the number of nodes needed for a majority, k is the number of retries.

   (b) List at least one disadvantage of each algorithm.(3 points)

   Token Ring: Token loss causes the system to shut down and it is costly to regenerate token.
   Central Server: Coordinator is performance bottleneck in large systems. Centralized algorithm also suffers single point of failure.
   Majority Voting: Majority voting may suffer from starvation during concurrent voting.

   (c) Which of these systems is the most robust to failure? Why? (3 points)

   A majority voting system is most robust to failure. Token ring suffers from failure when the token is lost, and the prevention measure is costly to implement. The coordinator in central server system is a single point of failure; the system will only recover after a new coordinator is created. In majority voting system, since there are multiple coordinators, crashing of one coordinator affects little for the node to gain majority votes from voters.

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 $\rightarrow$ 111, P2 $\rightarrow$ 212 and P3 $\rightarrow$ 103
   (20 points)

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)			
5	1	251	S to 3	(reply 113)	121	222	113
	2	262	S to 3	(reply 113)	222		121
							222
6	1	261	R from 2	(reply 121)	121	222	113
					222		121
							222
7	3	263	R from 1	(reply 113)	121	222	113
	3	273	R from 2	(reply 113)	222		121
							222
8	3	283	ExCS		121	222	121
					222		222
9					121	222	121
					222		222
10	3	293	S to 1	(reply 121)	121	222	
	3	303	S to 2	(reply 222)	222		
11					121	222	
					222		
12	1	301	R from 3		121	222	
	2	312	R from 3		222		
13	1	311	ExCS		222	222	
14					222	222	
15	1	321	S to 2	(reply 222)		222	
16						222	
17	2	332	R from 1	(reply 222)		222	
18	2	342	ExCS				

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?) (5 marks)

Including REDO information allows ARIES to reply OK to committed transactions before writing their information to the primary data structure.

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: (24 marks)

   - 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:

[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.

```
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)

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

Only the transfer can succeed. The higher-numbered PREPARE, which was received by all servers, will cause them to ignore the lower numbered proposals by S1.

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.
```

Either can succeed. If S2 receives a prepare OK from S3 before hearing from S1 and S2, it will include the value "withdraw" in its later messages. If it hears from S2 and S1 first, it will conclude (correctly) that it has a majority telling it can do what it wants, and it will propose the transfer.

5 GFS/HDFS/Spanner

1. Short answer questions (For true/false : write a small explanation for your answer.) (20 marks)

   (a) How many node failures can GFS or HDFS tolerate (at least)?

2 failures, since there are 2 secondary replicas available for fault tolerance.

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

The number of blocks for a given size increase which means the master node has to store more metadata which can be overwhelming for the master.

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

The client contacts the master to get the chunk mapping, the GFS client fetches the data chunks directly from the respective chunk servers based on the metadata.

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

False. By default, the fixed chunk size is set to 64 MB. Too many small files can overwhelm the master node and also lead to space wastage if the file size is < 64 MB.