Name _____Solutions_____ netid ____solutions_____

CS 425/ECE 428 Distributed Systems

Mid-term Exam II

Total points: 100

1. (10 points) Consider a network of 3^k nodes where K is an integer. Each node in this network has a unique identifier consisting of K digits, where each digit may take the value 0, 1 or 2. In this network, two nodes are connected by a link if and only if their addresses differ in one digit position. For example, for K=3, node 120 has a link to node 121 and 122, but does not have a link to node 111.

Assume K=4 in parts (a), (b), (c) and (d) below.

(a) List all the node with links to 2100.

0100, 1100, 2000, 2200, 2110, 2120, 2101, 2102.

(b) What is the number of links on a shortest route between nodes 2100 and 1202?

3

(c) Specify any one shortest route between nodes 2100 and 1202 (i.e., list all the nodes on the route).

2100, 1100, 1200, 1202

(d) How many node-disjoint *shortest* routes exist between nodes 2100 and 1202?

3

(e) Determine the following in terms of K, for all K>1:

Degree of each node (i.e., number of its neighbors): 2* K

Longest length among all possible shortest paths (across all pairs of nodes): K
Note that the last quantity defined above is called **diameter** of the network.

2. (10 points) Consider a Chord system that hashes key and node identifiers to 8-bit values. There are twenty nodes in the given system, whose hashed identifiers have the following values:

9, 10, 11, 12, 13
128, 129, 130, 131, 132
150, 151, 152, 153, 154
250, 251, 252, 253, 254.

a) List the 8 nodes that node 13's finger table will point to, in order, starting from the immediate successor.

0	1	2	3	4	5	6	7
128	128	128	128	128	128	128	150

b) Say keys with hashed identifiers 5, 50 and 80 have been inserted into the above system. Determine the nodes that store these keys.

9 stores 5, 128 stores 50 and 80

c) When node 13 queries for key 11, what path will it take? List each node along the path.

Node 13th finger table

0	1	2	3	4	5	6	7
128	128	128	128	128	128	128	150

Go to node 150
Node 150's finger table

0	1	2	3	4	5	6	7
151	152	154	250	250	250	250	128

278-256 = 22
Go to node 250
Node 250's finger table

0	1	2	3	4	5	6	7
251	252	254	9	10	128	128	128

Go to node 10
Go to node 11

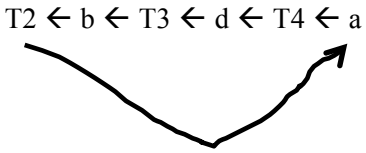
Path: 13, 150, 250, 10, 11

3. (10 points) Operations performed by four transactions are shown in each part below. Assume there are no other operations. No timing information between the transactions are given for the execution of the operations. Assume that *strict two-phase locking* mechanism with *read-write locks* is employed. In each part below, state whether deadlock may possibly occur when these transactions are executed. If you answer no, explain why. If you answer yes, show the Wait-For graph at the time when the deadlock occurs.

(a) YES

T1	T2	T3	T4
read(a)	write(b) read(a)	write(d) read(b)	write(a) read(d)

Example: T2, T3 and T4 perform their first operation (each is a write operation) first. Their read operations block waiting for a lock. The Wait-for graph is shown below.



(b) NO.

Except for variable c, there are no write operations to other variables. Variable c is only accessed by T3.

T1	T2	T3	T4
read(d)	read(d) read(a)	read(c) write(c)	read(a) read(d)

4. (10 points) The table below shows the interleaving of the operations performed by transactions T1, T2 and T3.

T1	**T2**	**T3**
	openTransaction	
openTransaction	write(a)	
read(b)		
write(c)		
closeTransaction	read(b)	
	read(a)	
		openTransaction
		write(a)
	write(b)	read(c)
	closeTrasaction	
		closeTransaction

(a) Suppose that optimistic concurrency control with **forward** validation is used. When forward validation fails, assume that the transaction performing the validation is immediately aborted. Which of the transactions (if any) must be aborted? Briefly explain why.

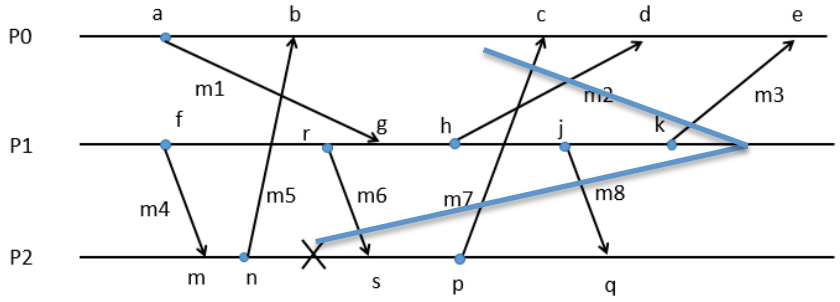
NONE.

Note that active periods of T1 and T3 do not overlap.

(b) Now suppose that **backward** validation is used. Which of the transactions (if any) must be aborted? Briefly explain why.

NONE

5. (10 points) The execution in the figure below shows application messages sent by three processes. Suppose that process P2 initiates the Chandy-Lamport snapshot algorithm at time marked by X on the execution of P2. The marker messages are not shown in the figure. Recall that the snapshot algorithm saves *channel* states and *process* states both.



a) What is the minimum number of messages that could be recorded by the snapshot algorithm? Write the channel states that are recorded as non-empty in this case.

P1 records m1

b) What is the maximum number of messages that could be recorded by the snapshot algorithm? Write the channel states that are recorded as non-empty in this case.

P0 records m2, m3,
P2 records m6, m8
The consistent cut corresponding to the above channel record is shown in the figure above.

c) The snapshot algorithm assumes that the channels are FIFO. Now imagine the system in the figure below is non-FIFO, and the fact that the messages shown are FIFO ordered is just a lucky accident: the additional messages for the snapshot algorithm might not be FIFO. Draw on the figure below an example of snapshot messages that would result in an inconsistent cut. Only draw the snapshot messages that are involved in your inconsistency. Briefly describe how the nodes' snapshots in your example will be inconsistent. (Note: this non-FIFO hypothetical is only for part c).

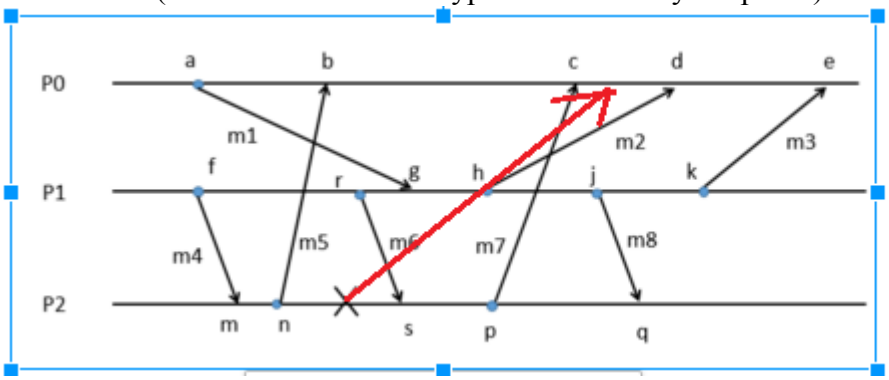
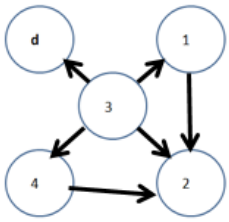


Figure for part c

6. (10 points) Link reversal algorithm: Perform the full link reversal algorithm on the following network. The destination node is labeled as **d** in the figure.

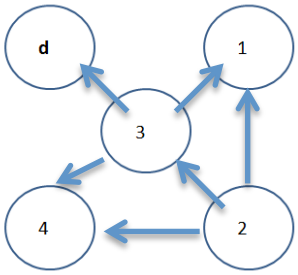
Use the figures below to present the directions of the links after each step of the algorithm. (You may not need to use all the figures.)

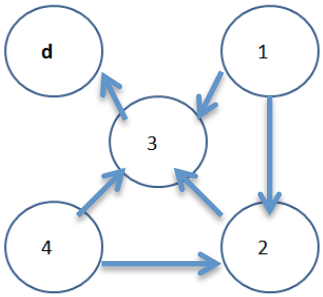


In each step, mark the node(s) performing reversals by a *

First use the figures in the left column, and then the figures in the right column.

2 reverses links in step 1, and nodes 1 and 4 reverse links in step 2 below.





7. (10 points) Consider the following properties for mutual exclusion algorithms:
   - No deadlock: If one or more process is waiting to enter critical section, eventually some process enters critical section.
   - No lockout: Each process that wants to enter critical section enters critical section after waiting for a finite amount of time.

For each algorithm below, state whether the algorithm satisfies the properties specified below (write Yes or No for each property). Assume that a process only stays in the critical section for a fixed finite duration of time.

(a) 2-Processor algorithm used with only 2 processes

- No deadlock _Yes
- No lockout ___No____

(b) Modified 2-Processor algorithm (provided in the handout for this exam) used with only 2 processes

- No deadlock __Yes
- No lockout ___Yes____

(c) The following modification of Bakery algorithm:

```
<Code for entry section>:
        Choosing[i] := true
        ~~Number[i] := max{Number[0], …, Number[n-1]} + 1~~
        Number[i] := 1
        Choosing[i] := false
        for j := 0 to n-1 (except i) do
                wait until Choosing[j] = false
                wait until Number[j] = 0 or
                 (Number[j],j) > (Number[i],i)
        endfor
<Code for exit section>:
        Number[i] := 0
```

- No deadlock __Yes But mutual exclusion may be violated_____
- No lockout ___No____

(d) Ricart-Agrawala algorithm

- No deadlock ___Yes____
- No lockout __Yes_____

8. (10 points) Consider a **synchronous** system with five processes P1, P2, P3, P4 and P5. The one-way delay of the link between each pair of processes is at least 10 ms and at most 200 ms. Process P1 may potentially crash, but the remaining processes are always reliable. You may ignore delays for performing computation below.

Note that parts (a) and (b) below are independent of each other.

(a) Process P2 sends a message to process P1 every 100 ms, and expects a reply from process P1 at least once every **k** seconds. When no reply is received within a **k** second interval, P2 assumes that P1 has crashed.

What constraint must **k** satisfy such that P2 never incorrectly concludes that P1 has crashed?

If P1 is still alive, then the maximum interval between two messages that are received by P2 is: 100 + (200 * 2 – 10 * 2) = 480ms. Specifically, consider the case when P2 sends out two consecutive messages m1 and m2, with an interval of 100ms. All messages before and including m1 suffer a round-trip delay of 20ms, and all messages after and including m2 suffer a round-trip delay of 400ms. Then the reply of m2 arrives at P2 480ms after the reply of m1.

So in order to guarantee that P2 never incorrectly concludes that P1 has crashed, k > 0.48 second.

(b) Design a mechanism that allows processes P2, P3, P4 and P5 to detect the failure of process P1 within 500 ms after P1 crashes. Your mechanism must use the least possible number of messages per second to achieve this goal; additionally, the processes must not conclude that P1 has failed before P1 actually fails. Present your mechanism and explain why you believe that it uses the least possible number of messages.

Sending message to all processes is expensive, instead, P1 heartbeats to P2 only, and the heartbeat interval is 100ms. When P2 detects P1's failure, it broadcast to P3, P4 and P5. The worst case delay for P2 to detect P1's failure is 100ms + 200ms = 300ms, and for P3, P4 and P5, it is 100ms+200ms+200ms = 500ms. The number of messages is 1msg/100ms (i.e., 10 messages per second) – when P1 fails, 3 additional messages are required from P2 to P3,P4,P5.

9. (10 points) Transactions T and U execute on a single server. In the table below, time proceeds from top to bottom, and the relative position of the operations indicate their relative order in time. Assume that a different lock is used for each shared variable w, x, y and z.

Assume that two phase locking is used, and that each transaction acquires locks only when needed. For instance, in part (i) below, transaction T acquires lock for variable z right before it performs Write(z,1). All locks held by a transaction are released when it performs closeTransaction (but not before that).

Determine whether the interleaving of operation as shown in parts (i) and (ii) below can occur under the assumption stated in each part (state YES or NO).

(i) ***Assume that Exclusive locks are used with two phase locking****:*

If you answer NO, delete a minimum number of operations <u>from transaction U only</u> so that the modified execution becomes possible while using exclusive locks. Assume that the relative order of remaining operations stays unchanged, even if some of the operations are deleted.

T	U
	OpenTransaction **Read(x) ←** **Write(z,1) ←**
OpenTransaction Read (y) Write(z,1) Read (z) Write(x,3) CloseTransaction	
	Read(z) Write(y,7) CloseTransaction

NO. Delete the operations marked in red above.

(ii) Assume that Read-Write locks are used with two phase locking:

If you answer NO, **change** a minimum number of **write** operations to **read** operations so that the modified execution becomes possible while using read-write locks.

T	U
openTransaction a = read (z) e = read (z) write(z, a*2) **write(y, 3+e) ←** closeTransaction	openTransaction c = read (y) write (z, c+1) write(y, 7*c) closeTransaction

NO. Change the operations marked in red above.

10. (10 points) State True or False:

(a) A causally consistent execution satisfies the requirements of sequential consistency.

False

(b) Serial equivalence for concurrent transactions can be achieved with the use of strict two-phase locking (Assume no deadlock).

True

(c) It is impossible to solve Byzantine Generals problem with one faulty node in an asynchronous system with four nodes.

True

(d) The last-writer-wins rule ensures that the shared memory system achieves linearizability.

False

(e) The CAP theorem implies that consensus is impossible in an asynchronous system in presence of crash failures.

False