CMU 15-440/640 Midterm Solutions
Fall 2020

Q1 True/False

13 Points

Before you start, scroll through the parts of this exam to see the questions and the point breakdowns. Spend time on questions with bigger point balances. Don't spend too much time on the questions with small point breakdowns (e.g., an individual T/F or short answer question).

Good luck!

Q1.1

1 Point

In Write-Ahead Logging (WAL), logs are written to disk only after the corresponding page is written to disk.

False. They are written before.

Q1.2

1 Point

Paxos ensures liveness in the face of (n/2) - 1 node failures out of n total nodes.

True. Only a strict majority of nodes need to be up.

We realized the use of "liveness" was confusing considering the relevant lecture slide, so we also accepted False: Paxos does not guarantee liveness.

Q1.3

1 Point

The callbacks approach reduces network traffic of reads for valid cached objects (i.e., objects in cache which have not been invalidated by a callback) to zero.

True. The onus is on the master copy/server to inform the clients or other copies when the object is modified.

Q1.4

1 Point

For a server with a small number of clients and a workload where writes are rare, check-on-use has lesser network traffic than broadcast invalidation.

False. The client is forced to check with the main copy each time it reads, leading to frivolous traffic.

Q1.5

1 Point

Even if the proposers in Paxos do not wait for a majority of acceptors to respond to the Prepare() and Accept() messages, the Paxos guarantees still hold.

False. This could lead to multiple values being chosen (violating safety).

Q1.6

1 Point

It's better to use UDP rather than TCP to send game updates in fast-paced multiplayer games where low latency is a high priority.

True. TCP's retries and ordering cause latency spikes in case of packet loss.

Q1.7

1 Point

Transmission delay is proportional to the packet size, while propagation delay is portional to the length of the link.

True. Transmission delay is the time spent sending the packet by the router, while the propagation delay is the time it spends travelling on the link.

Q1.8

1 Point

In the basic Paxos algorithm that we saw in class, livelocks are impossible.

False. The nodes may repeatedly fail to reach majority.

Q1.9

1 Point

The non-blocking variant of remote-write primary backup guarantees sequential consistency.

False. Because it is non-blocking, clients apply writes before the server they contacted hears back from the primary, possibly causing clients to apply writes in a different order than the primary.

Q1.10

1 Point

Causal consistency is a weaker notion of consistency than sequential consistency.

True. Sequential consistency has stricter ordering requirements (all nodes must agree on ordering of events).

Q1.11

1 Point

If no error is detected at the receiver of a system that uses checksum, we can be certain that no errors have occurred.

False. A sufficient number of errors may convert the received message into a different valid combination of data + checksum.

Q1.12

1 Point

Parity disk becomes the bottleneck in RAID-5 (rotating parity) configuration.

False. Different blocks have different parity disks (the “rotating” in “rotating parity”) to avoid this.

Q1.13

1 Point

We can increase the write throughput of a RAID-4 (single parity) configuration by adding more disks.

False. The single parity disk becomes a write bottleneck.

Q2 Short Answer

12 Points

Answer these questions succinctly with at most one or two lines.

Q2.1

2 Points

Explain two major differences between standard TCP and UDP. *(Remember that P1 was a UDP protocol that added some functionalities of TCP.)*

TCP: reliable delivery, slower delivery, handles error, does have acknowledgement

UDP: non-reliable delivery, faster delivery, discards erroneous packets, does not have acknowledgement

(Any other correct responses were also accepted.)

Q2.2

2 Points

Suppose you want to modify the “Leases” approach to caching to mimic other approaches as closely as possible. What would you choose the lease duration to be in order to mimic

a.) Check on use

0. Check on use is equivalent to not having a lease.

b.) Callbacks

Infinity/very large number. The lease lasts indefinitely (until the server revokes it with a callback).

Q2.3

2 Points

What do cycles represent in a wait-for graph?

Deadlocks. They represent a cycle of transactions which are waiting for locks held by each other.

Q2.4

2 Points

Which variant of two-phase locking guarantees ACID properties?

Strong Strict 2PL. (See slide 38 in the Oct 1 lecture slides.)

Q2.5

2 Points

Suppose your heavily used Orange laptop crashes on average after one year (365 days in a year) of use and it takes 10 days for it to be repaired. What is the availability of this Orange laptop?

(365 days) / (365 + 10 days) = 365/375 ≈ 97.3%

Q2.6

2 Points

Consider you have a RAID-1 (mirroring) system with 100 disks in total. Let R denote the maximum throughput for reads as well as for writes for a single disk. What is the maximum throughput of this RAID-1 system for

a.) Reads?

100*R. You can read from each disk simultaneously.

b.) Writes?

50*R. Each write is sent to two disks (mirrors of each other), so you can get at most 100/2 = 50 disks' worth of write bandwidth.

Q3 Time Synchronization

14 Points

Impressed by how time synchronization works in distributed systems, John decides to rent a time server with an accurate GPS receiver installed. He plans to connect to the server with his personal computer, and there's one router between his PC and the server. Please answer the following questions.

Q3.1

2 Points

First, John wants to calculate the estimated minimum one way delay of sending a time sync request to the server. Assuming store-and-forward routing as well as the following scenarios listed below, please help John with the calculation (consider only one request message for one way):

- The link from John's PC to the router has a bandwidth of 5Mbps, and a latency of 10ms.
- The link from the router to the server has a bandwidth of 2Mbps, and a latency of 100ms.
- The time sync request has a size of 20Kb.

Note: Show your steps for full credit.

20Kb/5Mbps + 10ms + 20Kb/2Mbps + 100ms = 4ms + 10ms + 10ms + 100ms = 124ms

Q3.2

2 Points

John now decides to use Cristian's Time Sync algorithm to synchronize the clock on his PC with the server. He sends a sync request at 08:06:00.000. Then, at 08:06:00.400 his PC receives a response from the server, saying that the time is 08:08:10:000. What time should John set his PC to?

08:08:10.200

Q3.3

2 Points

Based on your answers to the above two sub-questions, what is the accuracy of the time estimate made in 3.2?

Note: Show your steps for full credit.

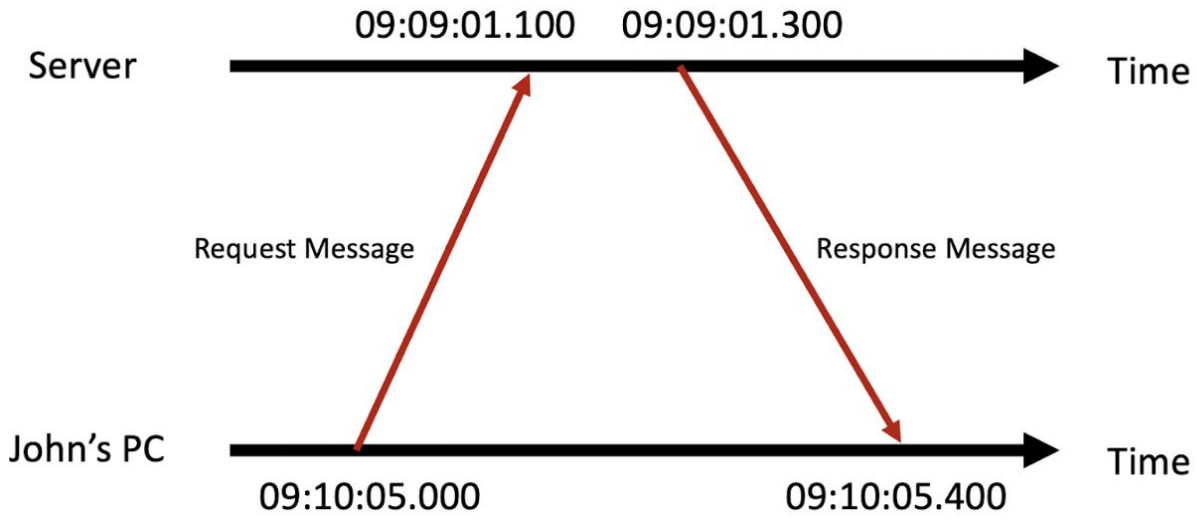
±(200 - 124) = ±76ms

Q3.4

4 Points

However, after using Cristian's Time Sync algorithm several days, John realizes that this algorithm may not be very accurate if it the RTT is large. Therefore, John decides to use NTP instead.

Consider the following diagram:



The four timestamps show when John sends the request, when the request is received at the server, when the server sends the response, and when John receives the response.

What is the RTT in this diagram (does NOT include server process time)?

200ms

Calculate the estimated time offset of NTP.

-1min 4s

Q3.5

4 Points

Besides algorithms dealing with real-world time, John also wants to try logical clocks, which only care about relative order of different events. John thinks that regular Lamport clocks have some drawbacks, so he'd like to choose between total-order Lamport clocks and vector clocks. Help him by clarifying some details about each algorithm.

How do total-order Lamport clocks achieve a "total-order"? Explain in 1-2 sentences.

Potential ties (possible due to concurrency) are broken using process ID, so that they become totally ordered.

What shortcoming of Lamport logical clocks do vector clocks overcome?

If one Lamport timestamp is less than another, it did not necessarily happen before the other (it may be concurrent). Vector clocks do not have this ambiguity.

Q4 Concurrency Control

6 Points

Q4.1

6 Points

Recall Gather.towns's simple Podium feature that allows whoever stands at the podium to have temporary global broadcasting rights to send messages to everyone in the room, but it does not give the speaker any feedback. In order to collect students' responses, TA Ray proposes a new feature to enable the speaker standing at the podium to create a poll among students in the room for simple binary responses: "go" or "no-go" for deciding when to progress in the recitation slides.

Ray decides to implement this feature using two-phase commit (2PC). Whoever stands on the podium also acts as a coordinator who sends a proposal to all students. Each student will

choose "go" or "no-go" and respond to the coordinator. Unless the coordinator receives unanimous “go” responses, it would not tell everyone to move forward with the next slide.

Assumption: Each student is indefinitely patient: he/she never asks the coordinator about a past decision or whose vote the coordinator is waiting on. Each student also tracks how many slides she/he has seen.

a.) **Scenario 1:** Ray is standing on the podium. One student suddenly left the classroom after Ray created the poll, and then came back ten minutes later. The student didn’t get a chance to respond to Ray. Can Ray move forward? Explain why.

Ray can’t move forward. Ray would have to wait indefinitely since he would never get a response from one of the students.

b.) Ray implemented a timeout mechanism to account for Scenario 1 above, and he set the timeout to be 5 minutes for everyone including himself. Can the student in Scenario 1 still vote when he comes back? Explain what would happen.

Since the timeout has been set as 5 minutes, Ray will ABORT the poll after 5 minutes since he did not get a response from one of the students. So, even if the student votes after he/she comes back, Ray will ignore the vote.

c.) **Scenario 2:** Standing on the podium, Ray created a new poll, and he set the timeout for everyone to be 15 minutes. Unfortunately, Ray's internet accidentally dropped off while he was broadcasting the message to let everyone move to the next slide. Some students got the message, some didn’t. What will happen to the students who got Ray's message? What will happen to the students who didn't get Ray's message?

In this scenario, the students who got the poll request and responded with a "go" will wait forever to get a response from Ray. The students who did not get Ray’s message will do nothing, but they could follow the CTP(cooperative termination protocol) by asking their neighbours.

Q5 Ye Olde Distributed Mutual Exclusion

10 Points

The year is 1850. You and a group of colleagues are collaborating on a programming assignment for Charles Babbage’s Analytical Engine, an early mechanical computer. However, you are all in different locations and can only communicate by postal mail. This makes just figuring out how to collaborate a challenge of its own! Assume that letters are never lost in the mail but may take a few days to arrive.

Q5.1

4 Points

You first try having a single paper copy of the assignment. Whenever someone has the single copy, they make any changes they want, then mail it to the next group member (in some fixed ordering of group members, with the last person then mailing it to the first and repeating the process).

a.) What distributed mutual exclusion algorithm does this correspond to?

Token ring. (The paper copy is the token.)

b.) Describe one **disadvantage** of this approach.

Possible answers: lost time if lots of people are not actually working during their turn; someone may forget to do it and then the token is lost forever.

Q5.2

3 Points

How would you instead implement centralized mutual exclusion in this setting?

Everyone mails requests to acquire the copy to a chosen coordinator, who mails it out to a winner of their choosing. The winner makes their changes and mails it back to the coordinator.

Q5.3

3 Points

Over time, the assignment becomes a thick manuscript that is expensive to mail. Your group decides that to save money, everyone will keep a replica (copy) of the current assignment and only mail descriptions of their changes. Given that everyone has their own copy that they can edit, is a mail-based distributed mutual exclusion algorithm still necessary? Why or why not?

Yes: they need to avoid making conflicting changes. If multiple people make changes simultaneously, the replicas may go out of sync and no longer have the same values, or they may violate some invariant.

Q6 Write-ahead Logging and ARIES

20 Points

Recall the simplified version of ARIES logging and recovery protocol as discussed in lecture. Recall that the log record entries have the following format

```
LSN: [prevLSN, TID, type]                          # All log records have these fields
LSN: [prevLSN, TID, Update, pageID, newVal, oldVal] # “Update” type log records
```

where LSN stands for Log Sequence Number.

Consider the following sequence of log entries followed by a crash:

LSN	Log
1	[-, 1, Update, P1, A = 10, A = 11]
2	[-, 2, Update, P1, B = 20, B = 22]
3	[2, 2, Update, P2, C = 30, C = 33]
4	[1, 1, Update, P2, D = 40, D = 44]
5	[4, 1, Commit]
6	[3, 2, Update, P1, B = 24, B = 20]
	CRASH

At the time of the crash,

Pages in memory: P1: pageLSN = 6 P2: pageLSN = 4

Pages on disk: P1: pageLSN = - P2: pageLSN = -

Q6.1

8 Points

Analysis Phase: Fill in the transaction table and dirty page table after the Analysis phase of the recovery protocol. Recall that in the simplified version of the protocol that you learned in lecture, a transaction is removed from the transaction table once the commit log entry is written. For your answer using the following format:

TT: (transactionID1, lastLSN1), (transactionID2, lastLSN2) …

DPT: (pageID1, recoveryLSN1), (pageID2, recoveryLSN2) …

The first two steps are shown below. Please add the rest:

TT:(1, 1)
DPT:(P1, 1)

TT:(1, 1), (2, 2)
DPT:(P1, 1)

TT:(1, 1), (2, 3)
DPT:(P1, 1), (P2, 3)

TT:(1, 4), (2, 3)
DPT:(P1, 1), (P2, 3)

TT:(2, 3)
DPT:(P1, 1), (P2, 3)

TT:(2, 6)
DPT:(P1, 1), (P2, 3)

Note that the transaction table has only one entry corresponding to a transaction ID and the lastLSN corresponding to that transaction is updated in place.

For this exam, we have given credit to solutions which have appended the new entries to the transaction table rather than updating in place as long as the new entries are correct.

Q6.2

4 Points

Redo Phase: Write down LSNs of log records to be reapplied in the Redo phase using a comma separated list.

1, 2, 3, 4, 6

Even though a transaction has been committed, log entries corresponding to it might have to be redone (if they fall in the range identified by the analysis phase) since the memory pages corresponding to the actions of that transaction might not have been flushed to disk when the crash occurred. This is the case in this problem. Also, there is nothing to redo in LSN 5 since it is just a log entry saying that transaction was committed.

Q6.3

8 Points

Undo Phase: Recall that for each undo operation, a log record called “compensation log record (CLR)” is added to the log.

Write down the first two log records for the undo phase. Note that the LSN of the next log record will start from 7.

Format of compensation log record: LSN: [prevLSN, TID, Comp, redoTheUndo, undoNextLSN] #compensation

7: [6, 2, Comp, B = 20, 3]
8: [7, 2, Comp, C = 33, 2]

(not asked for but it is fine if the third entry is also provided which is 9: [8, 2, Comp, B = 22, -])

Q7 RPC

10 Points

In p0partA, we asked you to implement a KVStore with a client-server-kvstore model, where the client sends add/delete/get requests to the server, the server executes the requests on the kvstore, and the server then sends back the result. After learning about RPC, Bob thinks to himself: “The server we implemented looks like a RPC server which parses requests and executes them”.

Q7.1

4 Points

In the test environment for p0partA, we expected your implementation to have an exactly-once semantics because there are no packet drops and kvstore actions always succeed. However, in a real environment, packet drops, action failures, etc. are common. Assuming our clients wait for results indefinitely and don’t retry in case of failures, what are the actual semantics of the implementation? Explain your answer.

At-most-once. Because the client and the server don't have a failure detection and retry mechanism, each request is executed at most once.

Q7.2

3 Points

In p0partA it is possible to associate duplicate values to the same key. Bob doesn’t like duplicate values and decided to rewrite the implementation for kvstore with the following pseudocode.

```
/* a map from string to int array */
map[string]int[] kvs

void put(string key, int value):
  for v in kvs[key]:
    if v == value:
      return
  kvs[key].append(value)
  return

int[] get(string key):
  return kvs[key]
```

```
void delete(string key):
  kvs.delete(key)
  return
```

Bob wants to publish this new kvstore to clients using a RPC framework without writing a server by himself. Which RPC semantics should he choose? Why?

At-least-once. With the new implementation, all three operations are idempotent because we disallowed duplicates. Putting the same values multiple times will result in the same final state.

Q7.3

3 Points

After publishing his kvstore service, Bob notices that the get(key) function is often slow when it has a large array to return. Bob remembers the pass-by-reference trick he often uses and changes his get() function to return the memory address of the result array:

```
*int[] get(string key):
  return &kvs[key]
```

Will his approach work? Why or why not?

No, because the client machine doesn't share the address space with kvstore and can't access the memory location returned.

Q8 "Semi-coordinated" checkpointing

10 Points

Crystal is designing a new distributed system that has 3 nodes P1, P2, P3 which perform complex, long-running computations related to climate modelling and communicate with each other by sending messages. In order to not lose computed results in the event of node failures, Crystal has decided to use checkpointing. Recall from the lecture about coordinated checkpointing and its implementation based on the 2-phase blocking protocol where a coordinator sends "do_checkpoint" request and all the nodes block until they receive a "checkpoint_done" or an "abort" message from the coordinator.

Q8.1

4 Points

What is a disadvantage of using coordinated checkpointing with the 2-phase blocking protocol given? (Please provide your answer in one sentence).

Large overhead due to large blocking time.

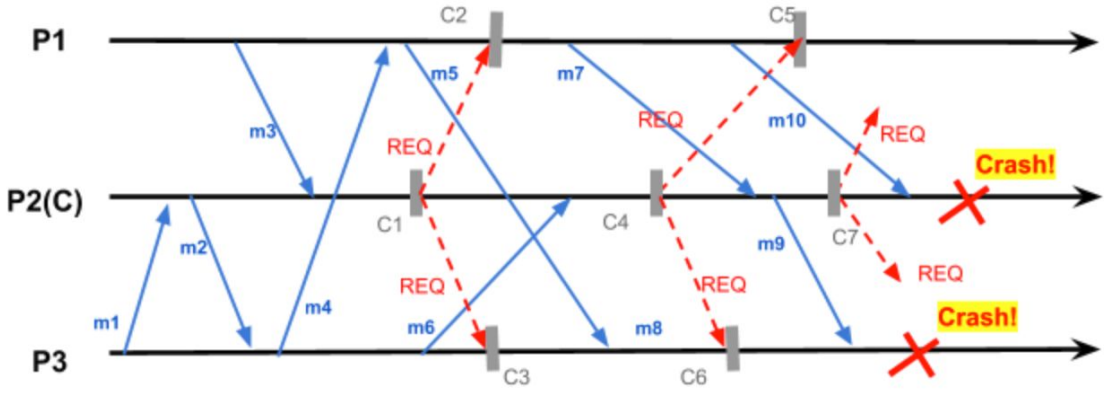
Q8.2

6 Points

Crystal is a bit worried about this downside of the blocking approach to coordinated checkpointing. She comes up with a simple “semi-coordinated” approach whose protocol is as follows:

- P2 plays the role of a coordinator. It takes a checkpoint and sends a checkpoint request (REQ) to the other two nodes.
- Each node will take a checkpoint as soon as it receives the checkpoint REQ.

Consider the timeline of events depicted in the following graph:



The blue solid arrows indicate application messages (m1-m10). The red dashed arrows indicate checkpoint request messages (REQs). Each grey bar represents a checkpoint taken at the corresponding process node.

a.) Now nodes 2 and 3 both fail as shown. Crystal decides to use the latest checkpoints at each node to form her recovery line. Is this consistent? Justify your answer.

The latest checkpoints at each node are C5-C7-C6. This is a consistent recovery line because in these checkpoints, there are no messages that are acknowledged by the receiver but not captured by the sender.

Please note that:

1. We want consistent checkpoints for the recovery line, so it is OK if they are uncoordinated.
2. Definition for consistent checkpoint: “In a distributed snapshot, if a process P has recorded the receipt of a message, then there should also be a process Q that has recorded the sending of that message. After all, it must have come from somewhere.” (Source: course textbook.)

For this exam question, we have accepted answers that say “C5-C7-C6 do not form a consistent recovery line because m9 is shown to be sent by C5, while it is not acknowledged by C7, thus a consistent recover line is C2-C4-C6” since one of the previous years’ exam solutions had discrepancies leading to potential confusion.

b.) Can you help Crystal find a consistent recovery line?

C5, C6, C7 is the most up-to-date recovery line.

For answers which said “no” for part (a), the most up-to-date recovery line is instead C2-C4-C6.

Q9 BONUS: Comparing a familiar distributed mutual exclusion algorithm with an unfamiliar distributed mutual exclusion algorithm

10 Points

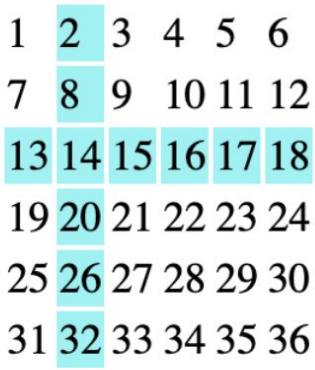
Now let’s consider another modern-day mutual exclusion algorithm. Below we highlight the rules behind Maekawa’s Algorithm. Don’t worry, you only need to understand it on a high enough level to compare it to other algorithms we discussed in class. Here we are simplifying it for the sake of the exam:

At its core, Maekawa’s is very similar to a distributed mutual exclusion algorithm like Ricart-Agarwala (R&A). The major differentiating factor is that unlike R&A, Maekawa’s is a quorum based algorithm. Lamport’s and R&A are permission based algorithms that need to request every other node in the system, whereas quorum based algorithms only do so for a subset of the nodes.

There are a couple of rules to how quorum sets work:

1. There is at least one common node between any 2 quorums. This just means if you are to represent a quorum as a set of nodes, then the intersection of two quorums would be non-null.
2. All nodes must be a part of some quorum set.
3. All quorum sets have some size KKK. (Note: generally KKK is much smaller than NNN)
4. Each node has its own quorum set that it makes requests to, but it appears in other nodes’ quorum sets as well. In total, each node is contained in exactly KKK quorum sets.

Here's an example: for each node, the quorum set it sends a request to is all the machines in the same row or column as itself (in blue you can find node 14's quorum set). Observe that 14 is also in the quorum set of other nodes (2,8,...)(2, 8, ...)(2,8,...), and this satisfies all the rules above with quorum set size K=2\sqrt{N} - 1.



There are two other kinks about Maekawa's. First, at every node there exists a boolean VOTED to denote whether it has given permission to another node to enter the critical section. This is initialized to false. Second, at every node you also have a FIFO queue that holds "pending" node requests in-order, if the node already gave permission to another.

When any node wants to enter the CS, it will send a REQUEST message to all the other nodes in its quorum set. When any node receives a REQUEST message, if it has VOTED already, it queues up the request, otherwise it sends a REPLY and sets VOTED to true.

Once a node has received a REPLY from every node in its quorum it enters the CS. After it is ready to exit the CS, it will send a RELEASE message to all the nodes in its quorum set. Upon receiving a RELEASE message, a node will attempt to send a REPLY message to the next pending request in its queue, otherwise, it will set VOTED to false.

Q9.1

4 Points

In a distributed system with N nodes, how many messages are sent in the interval from a node entering the CS to it exiting the CS? Assume it is the only node that is waiting and there are no further requests for entering the CS. You can assume the size of a quorum set is K.

3*(K-1) messages (K-1 messages to REQUEST, K-1 messages for REPLY, K-1 messages for RELEASE).

2*(K-1) messages (K-1 messages for REPLY, K-1 messages for RELEASE) was accepted with the assumption that requests are already sent.

Q9.2

3 Points

List at least one advantage of Maekawa's **compared to** R&A. Be sure to include a brief explanation (1-2 sentences) about what makes it better.

Much fewer messages need to be sent, since generally 3*(K-1) << 2*(N-1).

Q9.3

3 Points

List at least one disadvantage of Maekawa's **compared to** R&A. Be sure to include a brief explanation (1-2 sentences) about what makes it worse.

Possible answers:

- Not as fair as R&A since it does not prioritize requests by timestamp.
- Possibility of deadlock due to conflicting votes.

(However, there are more complex forms of Maekawa's that solve these issues.)

Not accepted: failure of a node could lead to starvation (R&A also has this issue).