15-440/640 Distributed Systems
Midterm SOLUTION

Name:
Andrew ID:

October 18, 2018

- Please write your name and Andrew ID above before starting this exam.
- This exam has 20 pages, including this title page. Please confirm that all pages are present.
- This exam has a total of 100 points.

Question	Points	Score
1	15	
2	20	
3	9	
4	12	
5	10	
6	16	
7	16	
8	2	
Total:	100	

True/False

1. Circle whether each statement below is *true* or *false*. ***ALSO***, give a one to two sentence reason for your answer.

   Each correct answer is worth 3 points (1pt for the T/F and 2pts for the reason).

   (a) (3 points) [True, False] CODA uses pessimistic replication to ensure high-availability to users.

   > **Solution:** FALSE. CODA uses optimistic replication, where it assumes write conflicts are rare and clients can hoard files.

   (b) (3 points) [True, False] Write Ahead Logging for transactions is used to ensure both Atomicity and Isolation properties of ACID.

   > **Solution:** False: WAL is used for Atomicity and Durability properties, it does not ensure Isolation (i.e. transactions happen in some serial order, one after the other) of ACID.

   (c) (3 points) [True, False] In the context of data stores, having eventual consistency does not imply that the data store is also sequentially consistent.

   > **Solution:** True: eventually consistency is the most relaxed form of consistency, while sequential consistency is much stronger, requiring the same sequential order on all nodes.

   (d) (3 points) [True, False] Kerberos is a system used to throttle users who are abusing their access to a DFS (e.g., making too many requests, consuming too much bandwidth).

   > **Solution:** False, Kerberos is an authentication system used to authorize and authenticate users, for example in a DFS.

   (e) (3 points) [True, False] Under LSP protocol from P1, after the initial connection setup phase, we can optimize network bandwidth by not sending out ACK 0 messages when the epoch timer fires, as long as the network does not reorder messages. There would be no change in the behavior of LSP protocol.

   > **Solution:** Solution: False. ACK 0 message works as a heartbeat. It prevents LSP server and client dropping the connection when they are not receiving any message just because they have nothing to send out. ACK 0 message is still needed even if the network guarantees in-order delivery.

Short Answers

2. In the following, keep your answers brief and to the point (i.e. 2-3 sentences).

   (a) (4 points) You are responsible for building the successor to the popular Skype Voice-over-IP protocol that works across the internet (high RTTs). Your boss went to CMU but did not take 441 or 440, and thinks you should use TCP due to its reliable delivery feature. Can you explain to him why this is not a good idea?

   Solution: Use UDP (as discussed in class). Reliability is less important than reducing delay and jitter. TCP can cause delays for longer RTT links due to reliable delivery and need for ACKs. (One point for mentioning one of packet loss/UDP, delay and jitter).

   (b) (4 points) NoSQL systems like Amazon's Dynamo allows writes to continue to all machines during a network partition. Once reconnected, Dynamo nodes seek to reconcile inconsistent file versions. Which technique would you choose to detect and flag parallel events? Briefly explain why.

   Solution: Vector clocks. Only algorithm that can properly preserve causality, so Dynamo would know that two events happened in parallel. Lamport clocks can't do this, as they create arbitrarily ordered timestamps, which might give two parallel events different timestamps. (2 points for Vector clocks and 2 points for correct reason. Note: Totally Ordered Lamport clocks is incorrect)

   (c) (2 points) Name and briefly describe two differences between the time synchronization algorithm used in NTP and Christian's algorithm.

   Solution:

   1. NTP incorporates the server processing time. Actually a different equation.
   2. NTP sends multiple measurements to several time master servers in parallel.
   3. NTP is hierarchically organized. Christian's algorithm considers only a single time master server.

   (1 points for each point. If you elaborated only one point, you would not get extra points)

You want to design a file system that can tolerate up to f node crash failures. For each of the following scenarios, how many file servers do you need at least? State the values and explain in one sentence.

(d) (2 points) The file system is using Primary-backup to replicate the files.

Solution: f+1. In Primary-backup, we maintain n replicas (1 primary and n-1 backups). Each backup can take over if the primary node fails.

(1 point for the number of file servers and 1 point for explanation.)

(e) (2 points) The file servers are using Basic Paxos to elect a leader among themselves.

Solution: 2f+1. Paxos requires the majority of the acceptors for a successful transaction commit.

(1 point for the number of file servers and 1 point for explanation.)

Consider two proposers A and B in the Paxos algorithm. Suppose that A is starting phase 2 (accept) and B is starting phase 1 (prepare). Assume that participants will not crash throughout the process.

(f) (6 points) Sketch out a scenario where the Paxos algorithm does not achieve the liveness property (a labeled sketch is fine, you can annotate a part that is "repeated forever").

Solution: Solution: 1. B receives the majority in phase 1 with a propose number higher than A's. 2. Before A receives majority in phase 2, B enters phase 2 and starts sending accept messages. 3. A fails to get majority because participants who accepted B's prepare messages reject A's accept messages. Note that participants who accepted B's prepare messages must have formed the majority, so it is possible that A cannot get the majority in phase 2. 4. A returns to phase 1 as it fails to get majority in phase 2.

By swapping the role of A and B, we return to the initial state and can repeat the same steps above such that neither will be able to proceed.

(A sketch must capture the 4 steps above to get 6 points. We also accept the "dueling proposers" scenario as the answer. We give 4 points to a mostly correct answer with minor issues and 2 points to an answer that fails to show the key idea but is sensible. Otherwise, we gives 0 point.)

Communication and RPC

3. You are hired by the augmented reality startup AR^2 that needs to perform a complex image processing algorithm. Because mobile phones have little computing power, AR^2's algorithm gets unacceptably slow as image sizes increase. Your manager, Jaine, proposes to use RPC to perform the work on a remote, more powerful server.

   You measure the following numbers, assuming a base image scaling factor of N.

   - The running time of AR^2's algorithm is $M(N) = 3N\ ms$ on a mobile phone. On the server, its $S(N) = N/3\ ms$.
   - Marshalling and unmarshalling take $(N/6 + 20)\ ms$ in total.
   - The average RTT between a user and the server is 80 ms.
   - Bandwidth is infinite between the user and the server.

   (a) (4 points) For what value of N is Jaine right, i.e., AR^2 should be using an RPC instead of a local call?

   Solution: Want $3N \geq 80ms + (N/6 + 20)ms + N/3 = 100ms + N/2$. So, $N \geq 40$.
   We also accepted answers that considered $2(N/6 + 20)ms$ or $4(N/6 + 20)ms$

   (b) (2 points) With geo-replication, AR^2 is able to reduce the average RTT. How will N change? (Give a qualitative answer)

   Solution: It decreases, because the server's computation becomes as fast as the phone's for a lower value of N.

   (c) (3 points) As AR^2 acquires 1) more and more powerful servers and 2) places them closer to users, you notice that the total RPC time does not improve very much. Explain why, by pointing out the bottleneck, in two sentences or less.

   Solution: The bottleneck is the marshalling time. No matter how much everything else is improved, the marshalling time will keep the RPC time high.

Distributed File Systems: since really everything that was innovative in DFS's was invented @ CMU !!

4. Karan has decided to drop out of CMU, work with Peter Thiel since he believes that education is overrated. He wants to create the next great cloud file storage company called "SoftBox". However, his starting funding is very low, since no one wants to give money to him. This means his Softbox system has occasional network partitions due to node failures, and high network latency.

   (a) (3 points) Karan wants to market that he can provide two guarantees: (1) That all file changes will be *immediately* visible to all other clients. (2) Any client can access any file at any time. Explain whether or not this is possible? (Hint: Recall that his funding is very low.)

   Solution: Not possible by CAP Theorem because he wants to guarantee Consistency and Availability, but his system will have network partitions.

   (b) (3 points) Karan argues that companies like Dropbox and Google Drive are successful because they provide the same guarantees as he is aiming for in Softbox. Is Karan right? Briefly explain your answer.

   Solution: No, they prioritize 'availability' over 'consistency'. Partial credit also given for answers relating Google/DropBox's high budget.

   (c) (3 points) Karan decides to relax his requirements a bit to get to market faster. He decides to prioritize "Any client can access any file at any time". Should Karan implement his system with optimistic or pessimistic replication?

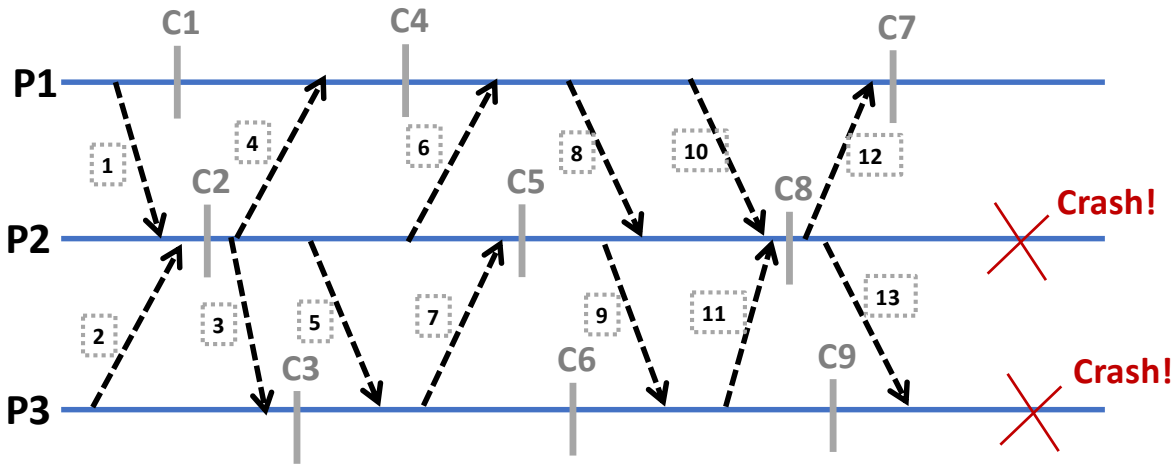
   Solution: Optimistic replication for high availability.

   (d) (3 points) Karan is worried that his users will experience extremely high latency every time they request a file. What is a simple technique to improve user experience?

   Solution: Mention: Client-side caching of whole files or LBFS.

Logging and Failure Recovery

5. You have designed a new distributed system, that is highly scalable and uses 3 nodes, P1, P2, P3 to send messages to each other (shown in square gray boxes). Since you are a smart CMU student you are convinced that node failures are a reality and you must prepare for the worst and bring back your entire application to a consistent state. You like the idea of checkpointing, and each node in your system takes independent snapshots after sending or receiving at most 3 messages. (C1 - C10 in the example below). Now, the following happens:



(a) (5 points) Process P2 and P3 both fail. Can your system recover from this failure and get back to a consistent state using the checkpoints? If so, calculate a consistent recovery line (listing the snapshots) for rollback. Note:

1. If there is no recovery line state why.
2. If there are multiple, list the ideal recovery line. Explain your choice.

Solution: We can't choose C7 for P1 since message 12 shows having been received in C7 by P1, but has not been sent by P2 in C8. So, we need to roll back P1 to C4 Instead. P2 also has to be rolled back to C5 since C8 won't work because of numerous messages that show P2 received in C8 (e.g. 10, 8 etc). P3 is similarly rolled back to C6.

Now C4, C5, C6 for a consistent recovery line based on the definition of a consistent checkpoint. 2,3,5,7 messages are both in C5 and C6. Similarly 1-4 messages are both in C4 and C5. Note, Message 6 is still OK since P2 shows it as sent in its checkpoint C5, but P1 does not show it being received (in C4). This is OK.

(2 points for the consistent recovery line found C4-C5-C6, 3 points for completely correct reasoning of why its consistent. We also generously gave 2 points for demonstrating general understanding of the concepts)

(b) (5 points) Next, you realize that both checkpoints C5 and C8 taken by P2 were actually corrupt on the disk. Can your system still recover to a consistent state by using the other stored checkpoints? If so, explain how. If not, explain why not, and give one efficient mechanism on how you could address this issue going forward.

Solution: Unfortunately, in this case it does not look like there is any other consistent recovery line. (1pt)

Reason, P2 rolls back to C2 now since C5 is corrupt. P1 cannot use C4 (message 4 shows received by P1 in C4, but not sent by P1 in C2). P1 rolls back to C1 then and P3 to C3. However, now P3 shows message 3 received, but P2 does not have it as sent in CHKPT C2. P2 has to roll back to the start. So, cascaded rollback – no good recovery line.

One mechanism: augment checkpointing with Logging and replay. That way, checkpoints don't have to be taken as frequently and their overhead can be minimized. When a failure happens, you can roll back the system to a consistent checkpoint and then replay the logs to get to a more recent consistent state.

(1 point for saying no recovery line. 3 points for clearly explaining why and mentioning cascaded rollback or its idea (2 points for explanations with some deficiencies). 1 point for mentioning WAL or coordinated checkpointing.)

BergerNet

6. BergerNet is a new social network started by the TAs and Professors of 15440 providing almost the same functionality as social networks do today. As a new engineering lead, it is your job to make some crucial design decisions that are either going to make or break the BergerNet.

   BergerNet uses RPCs. Identify the easiest-to-implement RPC semantic (at-least-once, at-most-once, exactly-once) for each of the following features. Give a *one sentence* reason for your choice. (1pt for semantic, 1pt for reason only if semantic correct).

   (a) (2 points) Transferring money to your friend on BergerNet

   > **Solution:** At-most once. Easiest to implement and can always retry sending on failure.

   (b) (2 points) Posting a message in the BergerNet messenger.

   > **Solution:** At-least or at-most once both work given appropriate reasoning.

   (c) (2 points) Deleting a friend on BergerNet

   > **Solution:** At-least once. Delete is idempotent. At-most once does not work because it is not the easiest to implement.

Next, BergerNet users want to upload group collages of their friends and colleagues. Privacy is important. So before a user can upload and publish a collage, the user needs to get approval from all their friends and can only upload the collage if all of them approve. Even if one friend rejects the proposal, the collage cannot be uploaded.

(d) (1 point) Name a protocol from lectures, with the least number of messages, to implement the collage feature.

Solution: 2PC. Token ring might be accepted based on reasoning below. (No partial credit)

(e) (2 points) Describe each stage of this protocol with a bullet point in the given context.

Solution: Voting phase - uploader asks and waits for votes Commit Phase - Uploader decides whether to commit or not and sends decision. OR resonable explanation for token ring. (-1pt if broadcasting decision is not mentioned)

(f) (1 point) If the computer of the user who uploads fails during this protocol what could be one negative outcome other than the upload failing?

Solution: Blocking. Consistency is not an issue because 2PC guarantees consistency by blocking. (No partial credit)

(g) (2 points) Is there a way of solving this problem? If yes, name a protocol that resolves the issue, if no, explain in one sentence why.

Solution: 3PC. (No partial credit)

BergerNet is very popular these days mainly because of how little it crashes.

(h) (2 points) Define reliability and availability (one sentence each).

- Reliability

- Availability

> **Solution:** Reliability is the frequency of errors in a system (how long can a system run before encountering an error.) Availibility is the ratio of the uptime to (uptime + downtime) (slide definition also accepted). 1 pt for each. No vague answers allowed.

Let's take the previous year as an example. BergerNet crashed once every hour and recovers within a second.

(i) (2 points) Would you say that BergerNet is highly available and/or highly reliable or neither ?

> **Solution:** Availability: 1-1/60/60 = 0.9997222. So, highly available. Reliability: time between crashes is less than an hour - that's bad. So not reliable. 1pt for highly avaiable, 1 pt for not reliable.

Hepp Dean's Hierarchical Mutual Exclusion

7. In 2027, Foodle is the world's leading Internet company with ten datacenters around the globe. This future version of the Internet has neither packet loss nor network partition nor sudden machine crashes, but the speed of light remains as a major constraint. A critical piece of Foodle's algorithms require mutual exclusion across all servers in all data centers.

   Foodle's design head, Hepp Dean, proposes a new hierarchical distributed mutex. Each datacenter has a unique coordinator, `M`. Across the ten datacenters, the ten coordinators uses Ricart & Agrawala's algorithm, which has the following thread-safe API: `BroadcastRequest()`, `GotGlobalLock()`, `BroadcastRelease()`.

   Within each datacenter, the coordinator manages datacenter-local mutual exclusion among the local client machines. Below is Hepp's pseudo code for coordinators.

```
 1  M.Init():
 2  # Initialize this leader machine
 3    M.hasRequestedGlobalLock = False
 4    M.hasGlobalLock = False
 5    M.Queue = NewQueue()
 6    M.mutex = 0
 7    M.cvar = NewCond(M.mutex)
 8
 9  M.GotGlobalLock():
10  # Triggered when M gets mutual exclusion among other leaders
11    M.mutex.lock()
12    M.hasRequestedGlobalLock = False
13    M.hasGlobalLock = True
14    M.cvar.SignalAll()
15    M.mutex.unlock()
16
17  M.Lock(id):
18  # Client with id calls this function using RPC when asking for mutual exclusion.
19    M.mutex.lock()
20    if not M.hasRequestedGlobalLock && not M.hasGlobalLock:
21        BroadcastRequest() // Broadcast to other leaders for mutual exclusion
22        M.hasRequestedGlobalLock = True
23
24    M.Queue.PushBack(id)
25
26    while not M.hasGlobalLock || M.Queue.Front() != id:
27        M.cvar.Wait()
28
29    M.mutex.unlock()
30
```

```
31  M.Unlock(id):
32  # Client with id calls this function through RPC when releasing mutual exclusion
33    M.mutex.lock()
34    M.cvar.SignalAll()
35    M.Queue.Pop()
36
37    if M.Queue.Empty():
38      M.hasGlobalLock = False
39      M.BroadcastRelease() //Broadcast to other leaders to release mutual exclusion
40    M.mutex.unlock()
```

(a) (4 points) Across different datacenters and in the absence of failures, explain briefly why Hepp's algorithm safely implements a mutex?

Solution: Ricart & Agrawala's algorithm enables only a single coordinator to hold the global lock. The coordinator gives a global lock to its clients within the datacenter only if it has a global lock. So, only one client in the datacenter gets a mutual exclusion and no two clients in different datacenter get the mutual exclusion.

- Mention "Ricart & Agrawala's algorithm" (or "global lock", explanation of Ricart & Agrawala) (+2)
- Mention only one "coordinator" (also can be "datacenter," "leader", "`M`") gets "global lock" (also can be "global mutual exclusion", "global mutex") or, gets into "critical section" etc. (+2)
- Walk through the code to explain one of the points above. Can cite line numbers or function calls and explain what they do (+2)
- Final score becomes $min(4, \texttt{sum})$

(b) (4 points) Within a single datacenter and in the absence of failures, explain in less than three sentences why Hepp's algorithm safely implements a mutex?

Solution: Within a datacenter, the coordinator performs centralized control over the lock among clients. `M.Lock` RPC returns (terminates) only if the coordinator has global mutual exclusion (line 26, following `M.GotGlobalLock` callback), this is the only client with `id` in mutex-protected queue (`M.Queue`, line 26). `M.Unlock` pops current client's `id` from the queue, wakes up other client's RPC using `M.Cvar`, and passes the local mutual exclusion to other client in the same datacenter.

- Mention "centralized control" by "single coordinator"(also can be "single machine," "`M`", "single leader") within a datacenter. Or, can say one

"queue" in datacenter but should say it's mutex, cvar protected queue – just saying queue does not count (+2)

- Mention "only one "client" (also can be "machine", "requestor", or similar, but not "`M`". "leader", "coordinator") can get "lock", "mutual exclusion", "mutex" or can get into "critical section" (+2)
- Mention coordinator's `mutex`, `cvar` protects/gurantees mutual exclusion. (+2)
- Walk through the code to explain one of the points above. Can cite line numbers or function calls and explain what to do. (+2)
- Final score becomes $min(4, \texttt{sum})$

(c) (2 points) Assume that many client machines simultaneously seek to acquire the mutex. In what order will they acquire it? Describe the order for both cases below.
All client machines in the same datacenter:

Solution: Within the same datacenter, each client will be given in First-come, First-served order of getting `M.Mutex`. Note that, there could be cases where client machine A calls `M.Lock` earlier than client machine B but client B's `M.Lock` gets `M.Mutex` first.

- Any answers implying "First-in-first-out", "First-come-first-served". Also allowing: "requested order", "timestamp of calling `M.Lock`" that can implicitly mentions FCFS order. Do not have to catch the case noted in the solution (the order of getting `M.Mutex`) (+1)

Client machines in different datacenters:

Solution: Client machines in different datacenters get mutual exclusion in the order their coordinator gets global mutual exclusion by Ricart & Agrawala algorithm.

- Any answers implying the order of Ricart & Agrawala's algotihm. such as "the order their coordinator gets lock", "TO Lamport timestamp order", etc. Do not allow just "timestamp order" because it is ambiguous whether it means "Lamport clock" or "physical clock" (+1)

(d) (3 points) State an advantage of Hepp's hierarchical distributed mutex, assuming that there are tens of thousands of servers in each datacenter?

Solution: Efficiency. Comparing to non-hierarchical distributed mutex where all client machines across datacenters use one distributed mutual exclusion algorithm such as Ricart & Agrawala, Hepp's algorithm is more efficient. Hepp's algorithm reduces the number of requests that should be sent through relatively poor inter-datacenter links, keeping a significant number of messages within datacenters. Also, it reduces the total number of messages.

- Any answers implying "efficiency", such as: "performance". Or, a descriptive answer that implies any part of the solution, such as "less number of message", "less inter-datacenter communication", etc (+3)

(e) (3 points) State a disadvantage of Hepp's hierarchical distributed mutex, in the context of the mutex design goals discussed in class? How would you resolve it?

Solution: Unfairness. From line 37, the coordinator releases its global lock only if there is no more local client in the queue. So, if the local clients of a coordinator are too demanding to get the lock, one coordinator will hold the lock forever. Clients in other datacenters will starve. The way to solve this problem could be setting timeouts for the time a coordinator can hold global mutual exclusion or setting the number of times a coordinator can give local mutual exclusion, etc.

- Any answers implying "unfairness". Any descriptive answer implying unfairness such as: "one datacenter can hold the lock forever" (+1.5)
- Any reasonable resolution. (+1.5)

Anonymous Feedback

8. (2 points) Tear this sheet off to **receive points**. We'd love it if you handed it in either at the end of the exam or, if time is lacking, to the course secretary.

   (a) Please list one thing you'd like to see improved in this class in the current or a future version.

   (b) Please list one good thing you'd like to make sure continues in the current or future versions of the class.