Problem Set 4

15-440/15-640 Distributed Systems Spring 2025

Assigned: Thursday, April 17, 2025

Due: Thursday, April 24, 2025 at 11.59 pm

Submission procedure:

- Create a **.pdf** of your answers and upload to `Gradescope`.
- Here are some tips to make your submission easy to read and to grade. Remember, the easier you make this, the less likely we are to make grading errors. Following these guidelines will help us to focus on the technical content of your answers rather than trying to understand what you have written.
    - Don't hand write your answers. Use Latex or Google Docs or some similar input mechanism. If you use Latex, a template can be found on the course web page.
    - Put the answer to each question on a separate page.
    - Carefully tag your pdf pages to each question on `gradescope`. You can use the SHIFT key to select multiple pages and associate them with a single question.
- Assume SI notation
    - 1 KB = 10^3 bytes, 1 MB = 10^6 bytes, 1 GB = 10^9 bytes
    - 1 Kbps = 10^3 bits per second (bps), 1 Mbps = 10^6 bps, 1 Gbps = 10^9 bps
    - but a byte is still 8 bits (not 10 bits) :-)
- **Remember that you have a limit of 2 grace days totaled over all four problem sets.** You can use at most one of those grace days per problem set. Although Gradescope does not track grace days, we will. Exceeding your grace days will result in a zero grade for that problem set.

Question 1 (20 points)

Michael had just launched his new social app, PingMe, which lets users send real-time pings to friends worldwide. To ensure low latency and high fault tolerance, he set up **6** replica servers in major cities: Pittsburgh, Boston, Los Angeles, Rome, London, and Mumbai. Each server holds a **replica of the app's user status database**, which updates whenever someone changes their "mood ping" and can be read by friends across the globe. It is absolutely vital that each user's status is updated with perfect one-copy semantics, even in the presence of transient networking failures and partitions.

Mahika, PingMe's CTO, implemented a **Gifford-style voting protocol** to manage consistency between these replicas. Assume that for a given attempt to access a file, each server has a probability of **p=0.6** of being available. Assume further that the availability does not change for the duration of an operation (i.e., retransmission won't help). For all the questions below, show your reasoning and intermediate steps.

A. All servers have equal votes, and you wish to **maximize read availability**.
   a. What read and write quorum sizes would you assign?

   b. What is the probability of a successful read using these quorum sizes?

   c. What is the probability of a successful write using these quorum sizes?

B. All servers have equal votes, but you now wish to **maximize write availability.**
   a. What read and write quorum sizes would you assign?

   b. What is the probability of a successful read using these quorum sizes?

   c. What is the probability of a successful write using these quorum sizes?

C. Each app user determines which servers to access during a read operation. Assume that read quorums are of size 3, and that replicas are accessed in parallel. You have obtained the following table of operation latencies for clients in Philadelphia, PA and Athens, Greece to each of the servers described above:

Server	Latency from Philadelphia (ms)	Latency from Athens (ms)
Pittsburgh	100	400
Boston	200	400
Los Angeles	350	500
Rome	400	100
London	300	200
Mumbai	500	350

a. For optimal performance to access a small (e.g., 1-byte) file in the absence of failures, which servers should the Philadelphia client choose in its read quorum? Which servers should the client in Athens choose?

b. After deploying the server selection described in (a) above, you find that even in the absence of failures, the load on one of your servers is significantly larger than on others even though all servers consist of identical hardware. Assume that most app users for PingMe originate in North America or Europe, with read-heavy requests. Using your answer to (a), state which server is likely to be most heavily loaded and why. Explain what you would do to eliminate this uneven load across servers. Be sure to state any assumptions you make.

D. A new engineer on PingMe's team has modified the number of votes for each replica, and has chosen a quorum size of 3 for both read and write operations. With the new vote assignments shown below, is one-copy semantics preserved? If yes, explain why. If no, provide a scenario in which one-copy semantics is broken.

Server	Votes
Pittsburgh	2
Boston	1
Los Angeles	1
Rome	2
London	3
Mumbai	1

Question 2 (20 points)

Each of the following subproblems shows a sequence of messages in a system using Paxos. Identify if the sequence could be a valid sequence in a correct Paxos implementation or not. (Note: these sequences are not necessarily from the start of a new consensus / election period) If it is invalid, identify the line number of the first line that is not correct, and explain what is wrong and what possible bugs could arise from it.

A. Assume there are 3 total nodes (S1, S2, S3)
   1. S1➔S1: Prepare(101)
   2. S1➔S1: Promise(101, null)
   3. S1➔S3: Prepare(101)
   4. S3➔S1: Promise(101, null)
   5. S1➔S1: PleaseAccept(101, v1)
   6. S2➔S3: Prepare(101)
   7. S1➔S1: Accept_OK()
   8. S1➔S3: PleaseAccept(101, v1)
   9. S3➔S1: Accept_OK()
   10. S3➔S2: Promise(101, null)
   11. S2➔S3: PleaseAccept(101, v2)
   12. S3➔S2: Accept_OK()

B. Assume there are 10 total nodes (S1, S2, S3, S4, S5, S6, S7, S8, S9, S10)
   1. S1➔S1: Prepare(101)
   2. S1➔S1: Promise(101, null)
   3. S1➔S2: Prepare(101)
   4. S1➔S3: Prepare(101)
   5. S2➔S1: Promise(101, null)
   6. S3➔S1: Promise(101, null)
   7. S1➔S5: Prepare(101)
   8. S1➔S7: Prepare(101)
   9. S5➔S1: Promise(101, null)
   10. S1➔S8: Prepare(101)
   11. S1➔S10: Prepare(101)
   12. S8➔S1: Promise(101, null)
   13. S7➔S1: Promise(101, null)
   14. S1➔S1: PleaseAccept(101, v1)
   15. S10➔S1: Promise(101, null)
   16. S1➔S2: PleaseAccept(101, v1)
   17. S1➔S1: PleaseAccept(101, v1)
   18. S1➔S5: PleaseAccept(101, v1)
   19. S1➔S1: Accept_OK()
   20. S1➔S4: PleaseAccept(101, v1)
   21. S5➔S1: Accept_OK()

C. Assume there are 2 total nodes (S1, S2)
    1. S1➔S1: Prepare(101)
    2. S1➔S1: Promise(101, null)
    3. S2➔S2: Prepare(102)
    4. S2➔S2: Promise(102, null)
    5. S2➔S1: Prepare(102)
    6. S1➔S2: Promise(102, already_accepted(101, v1))
    7. S2➔S1: PleaseAccept(102, v1)
    8. S1➔S2: Accept_OK()

D. Assume there are 3 total nodes (S1, S2, S3)
    1. S1➔S1: Prepare(101)
    2. S1➔S1: Promise(101, null)
    3. S1➔S3: Prepare(101)
    4. S3➔S1: Promise(101, null)
    5. S2➔S2: Prepare(102)
    6. S2➔S2: Promise(102, null)
    7. S2➔S3: Prepare(102)
    8. S3➔S2: Promise(102, null)
    9. S2➔S3: PleaseAccept(102, v1)
    10. S1➔S3: PleaseAccept(101, v2)
    11. S3➔S2: Accept_OK()
    12. S3➔S1: Accept_OK()
    13. S2➔S2: PleaseAccept(102, v1)
    14. S2➔S2: Accept_OK()

Question 3 (20 points)

Helen, Monica, Mahika, and Michael each have an account at a different branch of PNCC Bank. Each branch handles local transactions independently. For transactions involving accounts at different branches, PNCC uses the two-phase commit (2PC) protocol to handle the distributed transaction. Each branch handles updates to its local accounts and contacts account owners (by text message) to verify if they approve of each transaction.

Account Owner	Branch name
Helen	Aspinwall
Monica	Bloomfield
Mahika	Carrick
Michael	Duquesne Heights

Example transaction:
Consider a transaction of the following form:

Bloomfield: deposit 100 to Monica's account
Carrick: withdraw 100 from Mahika's account

This means we want to transfer 100 dollars from Mahika's account to Monica's account. The Bloomfield branch (B) will check by sending a text message to verify that Monica approves of this transaction before voting yes. The Carrick branch (C) checks if there is enough money in Mahika's account and whether she approves before voting yes. The transaction commits only if both branches vote yes. Note that any of the branches can act as the coordinator.

A. Helen wants to withdraw $200 from Monica and Mahika. She will send a transaction request involving:

   Aspinwall: deposit 400 to Helen's account
   Bloomfield: withdraw 200 from Monica's account
   Carrick: withdraw 200 from Mahika's account

   Suppose Monica turns off her phone during her midterm exam, but forgets to turn it back on until after the weekend. Describe how the 2PC protocol will progress in this situation and what the results will be at each node. State any assumptions clearly.

B. Currently, the 2PC implementation for the bank stops resending decisions after 20 minutes in the COMMIT phase. In 1-2 sentences, describe a potential negative consequence for the scenario above and how to fix it.

C. Every day, each branch reboots its system at 5 am. After rebooting, each system reads its write-ahead log to resume all processes and ensure ongoing transactions complete

successfully. Careful review of the recovery code has confirmed that it is bug free. Despite this, a bug of the following form often manifests itself:

Aspinwall: withdraw 100 from Helen's account
Carrick: withdraw 100 from Mahika's account
Duquesne Heights: deposit 200 to Michael's account

The bug is that although Michael is credited with $200, only Helen's account has $100 deducted – Mahika's account is unchanged.

a. What implementation bug could have caused this?

b. Which ACID property is violated?

c. Propose a method to fix this problem.

Question 4 (20 points)

Melody and Ruiqi are trying to book tickets to see the newest anime star: Rika Furude. The venue keeps a database of available seats, prices, and revenue by seating section. The initial contents are as follows:

Section	seats_remaining	price	revenue
Main	220	200	8000
Balcony	180	150	3000
Wing	80	40	2000
Mezzanine	60	50	1000

A. The ticket purchase system uses intention lists for the database to help ensure consistency and fault tolerance.

Suppose three concurrent transactions with the following pseudocode are in progress (starting with initial values above):

time	T1	T2	T3
1	BEGIN		BEGIN
2	Balcony.seats_remaining = Balcony.seats_remaining - 1	BEGIN	
3		Main.seats_remaining = Main.seats_remaining - 10	
4	Balcony.revenue = Balcony.revenue + Balcony.price		Wing.seats_remaining = Wing.seats_remaining + 50
5		Main.revenue = Main.revenue + Main.price * 10 * 0.9	Wing.price = Wing.price - 20
6	COMMIT	Main.price = Main.price + 10	Wing.seats_remaining = Wing.seats_remaining - 25
7		ABORT	Wing.revenue = Wing.revenue + Wing.price * 25 * 0.9

a. Write the intention lists for each of the given transactions after time = 7.

b. Suppose the system crashed at time 8. Immediately after the crash and before recovery, what are the values stored in the database for each section?

c. During recovery, what operations would recovery redo (if any)? What operations would recovery discard?

B. Ruiqi believes the ticket purchase system can be much more efficient if implemented with a Write-Ahead Log (WAL), and proceeds to build such a system. Anything to ensure she can see Rika!

Assuming the system starts in the same initial state as specified above, consider the following sequence of operations from two concurrent transactions:

time	T1	T2
1	BEGIN	
2	SR = Main.seats_remaining	BEGIN
3	SR = SR - 2	P = Balcony.price
4	Main.seats_remaining = SR	
5	R = Main.revenue	
6	P = Main.price	P = P + 50
7	R = R + P * 2	Balcony.price = P
8	Main.revenue = R	
9	COMMIT	P = Main.price
10		P = P - 10
11		Main.price = P

a. After time = 11, show the contents of the WAL. Clearly distinguish between the portions on disk and the portions in volatile memory. Clearly indicate for each log record which transaction it belongs to.

b. Melody suggests to Ruiqi that we can perform log truncation for all the logs made before time = 10 because the T1 has committed by then. Is she correct? Why or why not?

c. Suppose the system crashes after time = 11, which operations will be redone? Why? What if the system crashes again during recovery?

Question 5 (20 points)

You are hired as a systems architect for ScottyNet, a new social media app designed for sharing short voice messages. The app is expected to scale to 10 million users with 10 MB of stored data per user and 50 requests per day per user.

A. You find servers available with 2 TB of storage that are able to process 1,000 requests a second. Each server costs $8,000 a month to rent, including unlimited networking. Assuming all clients' requests come in uniformly during the day, how many servers would you need to support your users and what would the cost be to rent them?

B. You consider a completely different implementation using a peer-to-peer (P2P) distributed hash table (DHT) using 3 replicas for each data item. Each user's device becomes a part of the P2P system, and contributes resources. Assuming read requests and barring any failures, how many nodes will a user need to visit (i.e. how many hops are needed) to process a request in the **best** and **worst case**?

C. Assuming the worst case in part B, if each hop takes 30 ms, what is the latency be per request?

D. **Half** of your users are mobile users. They have metered data plans and are charged $1 per GB of data transferred (in either direction). Each read request returns 10 KB of data. In a 30-day month, what is the total data transfer cost charged to all of your mobile users due to requests? (You can assume the request messages are negligible, so only the result payloads need to be considered; assume all requests are reads, and assume no failures).

E. After conducting some research, you expect that 2.7% of mobile users will churn daily. This means that each day 2.7% of the mobile users leave, but are replaced with new users, so the total user population stays stable. What is the total churn cost to your users over the period of a month?

F. What are some things you can do to reduce these costs to the mobile users in a P2P implementation?