# 416 practice questions (PQs)

1. **Goal**: give you some material to study for the final exam and to help you to more actively engage with the material we cover in class.

2. **Format**: questions that are in scope of what we covered in the lecture. Each question slide is followed by an answer slide.

# PQ 1

- I'm using three file systems: AFS, NFS, CODA. I go camping and disconnect from the network. Files in which file system remain accessible to me? [Choose one answer]

  A. AFS

  B. NFS

  C. CODA

  D. AFS and NFS

  E. NFS and CODA

  F. AFS and CODA

  G. AFS, NFS, CODA

# PQ 1

- I'm using three file systems: AFS, NFS, CODA. I go camping and disconnect from the network. Files in which file system remain accessible to me? [Choose one answer]

A. AFS

B. NFS

**C. CODA [Only CODA has support for disconnected operation]**

D. AFS and NFS

E. NFS and CODA

F. AFS and CODA

G. AFS, NFS, CODA

# PQ 2

- You want to extend assignment 2 with a set of secrets, each of which can only be used once. How do you change your existing implementation to accomplish this? [Answer in a couple of paragraphs]

# PQ 2

- You want to extend assignment 2 with a set of secrets, each of which can only be used once. How do you change your existing implementation to accomplish this? [Answer in a couple of paragraphs]

Take a sequence of secrets on the command line, parse these into a list. Whenever a client sends the fserver a client-hash, check if the client-hash matches some secret from the list by iterating through the list and performing the check:
$md5(nonce+secret_i) =?= client\text{-}hash$

If this is true for some i, delete $secret_i$ from the list.

# PQ 3

- Alice and Bob have laptops, each of which uses NTP to set the local time. Alice and Bob use a chat app. to communicate. This app. timestamps each message at the sender using the local clock and includes this clock value in the message. It shows a history of messages on the screen in an order based on these timestamps. Based on this description, can the scenario below occur? Why or why not? [Answer in a couple of sentences]

Alice: hi

Bob: hello

Alice: how is 416?

Bob: I like it

Actual exchange

Bob: hello

Bob: I like it

Alice: hi

Alice: how is 416?

Alice's history

Bob: hello

Bob: I like it

Alice: hi

Alice: how is 416?

Bob's history

# PQ 3

- Alice and Bob have laptops, each of which uses NTP to set the local time. Alice and Bob use a chat app. to communicate. This app. timestamps each message at the sender using the local clock and includes this clock value in the message. It shows a history of messages on the screen in an order based on these timestamps. Based on this description, can the scenario below occur? Why or why not? [Answer in a couple of sentences]

Yes, this is possible. We can generate this scenario by having Bob's laptop set its time to 3:00PM before the conversation with Alice, and Alice's computer set it's time to 3:01PM before the conversation with Bob.

Actual exchange

| |
|---|
| Alice: hi |
| Bob: hello |
| Alice: how is 416? |
| Bob: I like it |

Alice's history

| |
|---|
| Bob: hello |
| Bob: I like it |
| Alice: hi |
| Alice: how is 416? |

Bob's history

| |
|---|
| Bob: hello |
| Bob: I like it |
| Alice: hi |
| Alice: how is 416? |

# PQ 4

- Which file system can support more clients, given a server that runs on identical hardware? [Choose one answer]

A. NFS

B. AFS

# PQ 4

- Which file system can support more clients, given a server that runs on identical hardware? [Choose one answer]

A. NFS

**B. AFS** [AFS pushes client load from the server by caching entire files on the client side. It is strictly more scalable (in terms of number of clients) than NFS.]

# PQ 5

- Given two events **a** and **b**, where a's vector timestamp is [2,3,4] and b's vector timestamp is [4,3,2], which of the following statements are true? [Choose one answer]

A. a happened before b

B. b happened before a

C. a and b happened concurrently

D. Not enough information

# PQ 5

- Given two events **a** and **b**, where a's vector timestamp is [2,3,4] and b's vector timestamp is [4,3,2], which of the following statements are true? [Choose one answer]

A. a happened before b

B. b happened before a

C. **a and b happened concurrently** [The two vector timestamps cannot be ordered, therefore the corresponding events occurred concurrently]

D. Not enough information

# PQ 6

- Given two events **a** and **b**, where a's Lamport clock timestamp is 2 and b's Lamport clock timestamp is 3, which of the following statements are true? [Choose one answer]

A. a happened before b

B. b happened before a

C. a and b happened concurrently

D. Not enough information

# PQ 6

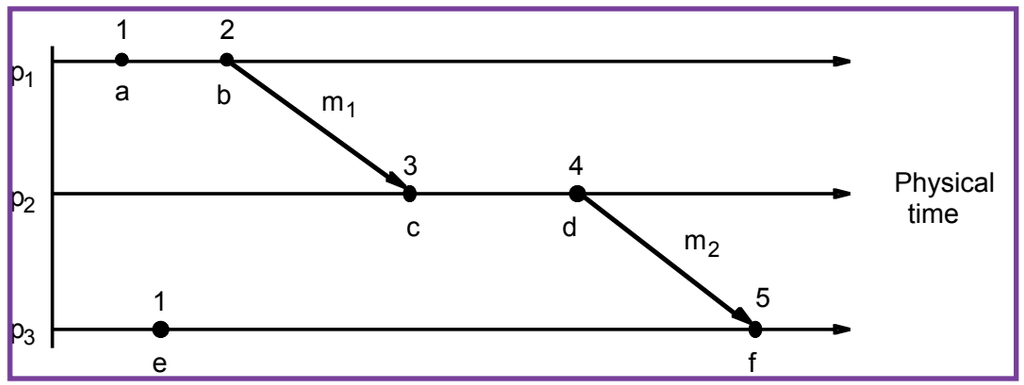
- Given two events **a** and **b**, where a's Lamport clock timestamp is 2 and b's Lamport clock timestamp is 3, which of the following statements are true? [Choose one answer]

A. a happened before b

B. b happened before a

C. a and b happened concurrently

**D. Not enough information** [For Lamport clock values $2 < 3$ does not imply that 'a happened before b'. Using the clock values alone we cannot tell which case we are in (see example diagram for counter-examples)]



# PQ 7

- When building a distributed system our first goal is to synchronize the clocks of the nodes in the system [Choose one answer]

A. True

B. False

# PQ 7

- When building any distributed system the first step is to synchronize the clocks of the nodes in the system [Choose one answer]

A. True

B. False [Synchronizing clocks is complex and many useful distributed systems do not require clock synchrony.]

# PQ 8

- In an asynchronous environment it is impossible to provide distributed mutual exclusion with fairness guarantees [Choose one answer]

A. True

B. False

# PQ 8

- In an asynchronous environment it is impossible to provide distributed mutual exclusion with fairness guarantees [Choose one answer]

A. True

B. False [We covered two algorithms that do this: dist. mutual exclusion with fairness in a ring topology and the Ricart Agrawala algorithm]

# PQ 9

- Ricart and Agrawala distributed mutual exclusion algorithm uses logical clocks [Choose one answer]

A. True

B. False

# PQ 9

- Ricart and Agrawala distributed mutual exclusion algorithm uses logical clocks [Choose one answer]

A. **True** [It uses lamport clock timestamps to decide whether or not a node should respond to a request]

B. False

# PQ 10

You are tasked with designing a new distributed mutual exclusion protocol based on voting. You come up with the following solution:

- A node requests permission (votes) from other nodes before proceeding to execute its critical section.

- The node does not proceed unless it receives a majority of replies from other nodes.

What are some problems with this solution?

# PQ 10

You are tasked with designing a new distributed mutual exclusion protocol based on voting. You come up with the following solution:

- A node requests permission (votes) from other nodes before proceeding to execute its critical section.
- The node does not proceed unless it receives a majority of replies from other nodes.

What are some problems with this solution?

1. Possibility of deadlock. Group of 6 nodes: node X receives 2 votes and node Y receives 2 votes. Neither can proceed.

2. Unfair. Node X issues requests before node Y, but node Y could collect majority of votes faster than X.

3. Cannot make progress when nodes fail (no “liveness” guarantee on failures). Group of 6 nodes: 3 nodes fail, then node X issues requests and receives 2 votes. But, it needs at least 3 votes to start executing critical section.

# PQ 11

- You are transmitting data packets over a link that can inject at most one bit error in each packet. What is the cheapest error detection solution that you can use to detect errors with perfect reliability?
  - md5 hash
  - Parity bit
  - Complement sums
  - CRC
  - md5 hash

# PQ 11

- You are transmitting data packets over a link that can inject at most one bit error in each packet. What is the cheapest error detection solution that you can use to detect errors with perfect reliability?
  - md5 hash
  - **Parity bit [detect a single bit flip perfectly]**
  - Complement sums
  - CRC
  - md5 hash

# PQ 12

- RAID uses complement sum for error detection

A. Yes

B. No

# PQ 12

- RAID uses complement sum for error detection

A. Yes

B. No [RAID uses Parity]

# PQ 13

- MTBF stands for

A. Mean time between failure

B. Mean time before failure

# PQ 13

- MTBF stands for

A. Mean time between failure  [Remember the timeline; MTBF captures all of the time the system was running]

B. Mean time before failure

# PQ 14

- Which of these raid levels has the lowest storage utilization (most disk space wasted on redundancy)

A. RAID 0

B. RAID 1

C. RAID 4

D. RAID 5

# PQ 14

- Which of these raid levels has the lowest storage utilization (most disk space wasted on redundancy)

A. RAID 0

B. RAID 1 [i.e., Drive mirroring. Wastes 1/2 of total storage capacity.]

C. RAID 4

D. RAID 5

# PQ 15

- Primary-backup replication is more fault-tolerant than quorum replication.

- True

- False

# PQ 15

- Primary-backup replication is more fault-tolerant than quorum replication.

- True

- False [If the primary dies, the entire system halts. Not so with a quorum-based system: as long as majority is alive, the system is available.]

# PQ 16

- You are a system architect. You decide to use Paxos as a core algorithm for fault tolerance. You want to design you system to survive f failures. How many replicas do you need in your system to satisfy this requirement?

A. f

B. f+1

C. 2f+1

D. 3f+1

E. 4f+1

# PQ 16

- You are a system architect. You decide to use Paxos as a core algorithm for fault tolerance. You want to design you system to survive f failures. How many replicas do you need in your system to satisfy this requirement?

A. f

B. f+1

C. 2f+1 [f fail => f+1 remain alive, which is enough for a quorum]

D. 3f+1

E. 4f+1

# PQ 17

- To use the Paxos protocol you must first elect a leader.
- True
- False

# PQ 17

- To use the Paxos protocol you must first elect a leader.

- True

- False [A leader-based Paxos is an optimization. You can have leaderless-Paxos. It's just slower.]

# PQ 18

- Like two phase commit, Paxos has two stages

- True
- False

# PQ 18

- Like two phase commit, Paxos has two stages

- True

- False [Paxos has three stages: prepare, accept, commit]

# PQ 19

- DNS is a globally distributed, strongly consistent, database
- True
- False

# PQ 19

- DNS is a globally distributed, strongly consistent, database

- True

- False [DNS is **not** strongly consistent]

# PQ 20

- A content distribution network provides a way for content providers to shed load from their servers

- True

- False

# PQ 20

- A content distribution network provides a way for content providers to shed load from their servers

- True [A CDN is fancy cache]

- False

# PQ 21

- Imagine that you wanted to build a CDN that not only cached static content, but also cached dynamically-generated results. Sketch out a high-level design for this kind of CDN. (hint: what properties must this kind of CDN provide?)

# PQ 21

- Basic design:
    - Server S computing the dynamically-generated content embeds a special hash H along with the akamai link to the content
        - H is a pointer to state necessary to generate the content, this state can be maintain at S until some timeout. Assumption: given two requests, if they resolve to the same hash H, then the dynamic content response is identical.
    - Client requests and downloads index.html containing akamai links. Client resolves akamai links to the akamai servers in the usual way.
    - Akamai server A sees the hash H, and first determines if the (dynamic) content corresponding to H is in its cache.
        - If content for H is in cache, A checks if this content has expired. If not expired then return the content to client.
        - If content is not in the cache, contact S, sending along H, and receive the generated content. S will also send along an expiration TTL for the dynamically generated content in its reply. Cache this content, then reply to client.

# PQ 22

- You are considering a startup that will use a peer-to-peer architecture. Your co-founder is a psychology major. Explain (all the possible) **advantages** of a peer-to-peer design to your business partner.

# PQ 22

- Elastic: scales with demand (peers bring resources to support more peers)
- Cheap: resources are not centrally provided/ administered
- Peer diversity = higher resilience to failures (e.g., geographical/network diversity)
- No single point of failure = more fault tolerant
- Resilient to gov. oversight/control

# PQ 23

- You are tasked with making a more reliable replication protocol based on primary-backup. You add **leases** to the backup so that a backup can reply directly to clients READS of data as long as the backup holds a lease. This works even when there is a partition! You believe that you've solved the CAP theorem. Where is your mistake?

# PQ 23

- Imagine a partition between primary and backup
- Consider the write path
- To provide C, primary not able to serve writes since the backup is unreachable. Backup, by definition, does not serve writes (that's the primary's job)
- Therefore, system sacrifices A. It's a CP-ish system.

# PQ 24

- Modify key-value service from assignment 4 to provide transactional semantics without durability (ACI). Do this by modifying the API and the front-end **only**. Describe your design in (non-Finnish) words.

# PQ 24

- Summary of expected answer
  - Change API: need either explicit or implicit txn begin/end boundaries and notification of commit/abort. Regardless, need to handle client disconnections.
  - Implementation: front-end performs concurrency control
    - simplest: allow one txn to execute at a time.
    - performs logging to be able to undo: operations logging or value logging
  - Aside: more efficient of replicas knew about transactions and could help abort transactions (e.g., keep logs of their own)

# PQ 25

- What are two difference between undo logging and redo logging in the context of a transactional database?

# PQ 25

- Difference 1
  - undo logging logs old values
  - redo logging logs new values
- Difference 2
  - undo logging writes data to disk continuously
  - redo logging writes data to disk all at once

# PQ 26

- Draw a time-space diagram in which the two-phase commit protocol blocks indefinitely because of a process failure.

# PQ 26

- Coordinator process fails after deciding outcome but before replying to any of the replica processes (i.e., all replicas are uncertain, therefore they cannot resolve the uncertainty on their own)

(see picture on whiteboard)

# PQ 27

- Draw a time-space diagram in which the Paxos protocol is not able to make progress (is not ‘live’) even though no nodes fail.

# PQ 27

- Two proposers, P1 and P2; 2n+1 acceptors
  - P1 issues prepare(k), receives n+1 promises
  - P2 issues prepare(k+1), receives n+1 promises
  - P1 issues accept(k,val), rejected since majority promised not to accept any proposal with proposal number <= k+1
  - P1 issues prepare(k+2), receives n+1 promises
  - P2 issues accept(k+1,val), rejected since majority promised not to accept any proposal with proposal number <= k+2
  - …
- (Key to progress is having one proposer at a time, i.e., electing a leader. But, note that election is itself not necessarily live.)

# PQ 28

- Let's say you want to change your non-bonus version of the A4 design to allow kv nodes to return results directly to clients. How does this change impact the rest of the design and what would you have to worry about in implementing this change?

# PQ 28

- Clients have to receive connections from kv nodes, which they do not know
- Clients have to match responses to requests (need a client-unique identifier per request)
- kv node might fail
  - Client would block waiting for a response
  - Have to implement reliability: on the client side, or at the front-end node
  - If we allow network failures, then clients may receive duplicate operations
- Clearly, interface between front-end and kv nodes has to change

# PQ 29

In A4 the client use RPC to connect to the system. What if instead of RPC they used UDP, sending one get/put/testset request in one UDP packet, and receiving one UDP packet in reply.

How does this change create extra flexibility in designing a solution to the A4 bonus (restartable front-end)?

# PQ 29

- Key insight:
  - RPC: when the front-end fails it exposes the failure to the client
  - UDP: does not expose failure to front-end
- Flexibility:
  - Could go with original A4 bonus design — expect front-end response in certain amount of time, consider it failed if it's too slow
  - Could make no expectations about front-end speed — expect a response eventually.
    - If we retain semantics (request always generates a response), then front-end now has new distributed state!
    - This state has to survive front-end failures (like unavailability info)…

# PQ 30

- Three phase commit protocol improves on two phase commit protocol in the following ways (check all that apply)
  - Performance
  - Availability
  - Consistency

# PQ 30

- Three phase commit protocol improves on two phase commit protocol in the following ways (check all that apply)
  - Performance
  - Availability
  - Consistency

# PQ 31

- A content distribution network implements a distributed cache
  - True
  - False

# PQ 31

- A content distribution network implements a distributed cache
  - True
  - False

# PQ 32

- Which of the following systems use client-side caching. Check all that apply.

- Networked file system (NFS)

- Andrew file system (AFS)

# PQ 32

- Which of the following systems use client-side caching. Check all that apply.

- Networked file system (NFS)

- Andrew file system (AFS)

# PQ 33

- It is impossible to provide **fair** distributed mutual exclusion in an asynchronous network.

- True

- False

# PQ 33

- It is impossible to provide **fair** distributed mutual exclusion in an asynchronous network.

- True

- False

  - A central server will do.

  - Use Ricart and Agrawala if you want to be fancy.

# PQ 34

- A peer-to-peer design has the following advantages over a traditional distributed system design. (Check all that apply)
  - Lower cost
  - Better scalability
  - Faster response time

# PQ 34

- A peer-to-peer design has the following advantages over a traditional distributed system design. (Check all that apply)
- Lower cost
- Better scalability
- Faster response time
  - Not the case as peers are typically distributed around the world.