Department of Computer Science

Distributed Systems

Exam 2

March 30, 2026

Discussion

Prepared with tender loving care by Paul Krzyzanowski

100 POINTS – 25 QUESTIONS – 4 POINTS EACH – For each statement, select the *most* appropriate answer.

1. With *Raft consensus*, a client receives a response to a command it sent:
   a. As soon as the leader appends the command to its own log.
   b. After the leader receives acknowledgment from every server it contacted.
   c. After all live servers have applied the command to their state machines.
   d. After a majority of servers have acknowledged that they have appended the command to their logs.

Correct answer: (d) Once a majority of servers acknowledge the entry, the leader commits it, applies it to its state machine, and returns the result to the client.

Incorrect answers:

a. The leader does not respond until it has received majority acknowledgment and committed the entry.
b. Raft requires only a majority acknowledgment to commit, not a response from every contacted server.
c. Followers apply entries after learning the commit index from a subsequent AppendEntries; the leader does not wait for this.

2. Which of the following best describes the *FLP Impossibility Result*?
   a. Consensus is impossible in any system where the network can drop messages.
   b. No consensus algorithm can guarantee a decision within a bounded number of round trips.
   c. Consensus cannot be achieved if more than half the nodes in a system crash.
   d. No deterministic algorithm can guarantee consensus in a fully asynchronous system if even one process may crash.

Correct answer: (d) FLP proves that in a purely asynchronous model, the inability to distinguish a crashed process from a slow one makes it impossible to guarantee both agreement and termination deterministically.

Incorrect answers:

a. FLP's result is about asynchrony and crash failures, not message loss specifically.
b. FLP is about termination, not the number of round trips; it says a decision may never be reached, not that it takes too long.
c. FLP requires only one potential crash failure to apply; it is not a statement about majority failures.

3. In Raft, why are *election timeouts* randomized rather than fixed?
   a. To give the server with the most up-to-date log a better chance of winning.
   b. To prevent a predictable heartbeat interval that could be exploited by an attacker.
   c. To reduce the chance that multiple followers start elections simultaneously.
   d. To allow followers to adjust their timeouts dynamically based on observed network latency.

Correct answer: (c) If all followers used the same timeout, they would likely start elections at the same time, repeatedly splitting votes and preventing any candidate from winning a majority.

Incorrect answers:

a. Randomized timeouts are independent of log state; log recency affects who wins an election, not who starts one.
b. Raft's timeout randomization is a correctness mechanism, not a security measure; heartbeat intervals are separate.
d. Raft election timeouts are randomized at startup, not adjusted based on network conditions.

4. A Raft follower receives an *AppendEntries* RPC but its log does not match the leader's log at the preceding index. What does the follower do?
   a. Accepts the entries and flags the inconsistency to the leader for later repair.
   b. Rejects the RPC, causing the leader to back up and retry from an earlier log index.
   c. Contacts other followers to determine the correct log state before responding.
   d. Truncates its log at the mismatch point and overwrites it with the leader's entries.

Correct answer: (b) A follower rejects any AppendEntries RPC that fails the consistency check. The leader then decrements its nextIndex for that follower and retries until it finds a matching point, after which the follower accepts and overwrites any conflicting entries.

Incorrect answers:
a. A follower that detects a mismatch rejects the RPC outright; it does not accept entries with a flag.
c. Followers never consult each other during log repair; all coordination goes through the leader.
d. The follower rejects the RPC on mismatch; it is the leader that must back up and find the agreement point before the follower will accept and overwrite anything.

5. A Chubby cell has five replicas, one of which is the master. What do the other four replicas do?
   a. They replicate state and back up the master.
   b. They serve client reads while the master handles writes.
   c. They balance client requests across the cell.
   d. They increase the storage capacity of the cell.

Correct answer: (a) Non-master replicas participate in consensus replication to maintain durable copies of all state, and redirect client requests to the master.

Incorrect answers:
b. All client requests, including reads, are directed to the master; non-master replicas do not serve clients directly.
c. Chubby is not designed for load balancing; it is a strongly consistent coordination service where all requests go through the master.
d. Replicas provide fault tolerance, not additional storage capacity; adding replicas does not expand the amount of data the cell can hold.

6. *ZooKeeper* is *not* suited for:
   a. Reading a portion of a large file.
   b. Coordinating a lock shared among multiple clients.
   c. Implementing leader election in a server cluster.
   d. Storing configuration data in a tree-structured namespace.

Correct answer: (a) Like Chubby and etcd, ZooKeeper reads and writes entire znode contents and is not designed to hold large files or to address portions of stored data.

Incorrect answers:

b. Distributed locking is one of ZooKeeper's primary use cases, typically built using sequential ephemeral znodes.

c. Leader election is a canonical ZooKeeper use case.

d. ZooKeeper's tree of znodes is inherently hierarchical and well-suited to configuration namespaces.

7. Which file system does not provide access transparency?
   a. NFS
   b. SMB
   c. GFS
   d. AFS

Correct answer: (c) GFS exposes a custom application-level API and is not accessible through standard file system calls; applications must use the GFS client library directly.

Incorrect answers:

a. NFS integrates with VFS so applications use standard file system calls with no awareness of remote storage.

b. SMB provides access transparency through Windows' file system driver model on Windows clients and through VFS on Unix clients; applications use standard file operations without any awareness of remote storage..

d. AFS integrates with VFS so applications use standard file system calls with no awareness of remote storage.

8. Why did the original NFS design have no *CLOSE* procedure?
   a. Files were closed automatically when the client disconnected.
   b. The server did not track which files clients had open.
   c. A server-side timer closed files; clients could reset it by issuing a read or write.
   d. A file handle gave the server enough information to clean up without an explicit close.

Correct answer: (b) NFS was stateless; the server held no per-client open-file state, so there was nothing for a CLOSE to act on.

Incorrect answers:

a. A stateless server has no knowledge of which clients are connected and therefore cannot close files on disconnect.

c. NFS had no server-side timers for file state; the design avoided all per-client state entirely.

d. File handles are opaque identifiers for locating a file, not records of client session state; the server has no session to clean up.

9. AFS was designed to scale to larger environments than NFS because:
   a. Read requests are load-balanced across multiple file servers.
   b. Entire files are cached at the client until the server indicates that the file has changed.
   c. A uniform global namespace allows the directory tree to span multiple servers.
   d. Large file contents can be split across multiple servers.

Correct answer: (b) Whole-file caching means most reads are satisfied locally with no server contact, dramatically reducing per-server load as the number of clients grows.

Incorrect answers:

a. AFS does not load-balance individual reads; a client fetches a file from one server and then reads from the local cached copy.

c. The global namespace improves usability and supports volume migration but does not itself reduce server load.

d. Splitting file contents across servers describes GFS and HDFS, not AFS.

10. *Opportunistic locks* (oplocks) in SMB allow:
    a. Clients to cache writes locally until they explicitly release the lock.
    b. Clients to lock byte ranges within a file.
    c. The server to push file contents to a client before it requests them.
    d. The server to grant and revoke client caching rights.

Correct answer: (d) The server grants an oplock giving the client caching rights, then sends an oplock break when a conflict arises, requiring the client to flush writes before the server grants a competing open.

Incorrect answers:

a. The server can revoke an oplock at any time by issuing an oplock break.

b. Byte-range locking is a separate SMB feature; oplocks apply to an entire open file handle.

c. Oplocks govern caching rights and invalidation, not prefetching or server-initiated data transfer.

11. NFS and SMB have both evolved over time. Which feature did not become part of both systems?
    a. Stateful operation.
    b. Disconnected operation.
    c. Compound requests.
    d. Server-initiated cache invalidation.

Correct answer: (b) Neither NFS nor SMB supports disconnected operation. That was in the Coda design.

Incorrect answers:

a. NFSv4 tracks open files and delegations; SMB 2 tracks open files and locks; both are stateful.

c. NFSv4 introduced Compound RPC and SMB 2 introduced compounding, so both support packing multiple operations into a single message.

d. NFSv4 uses delegations (recalled on conflict) and SMB uses oplocks/leases (broken on conflict), so both support server-to-client cache invalidation.

12. Without a finger table, Chord routing requires *O(n)* hops. What does the *finger table* add to reduce this?
    a. Pointers to nodes at exponentially increasing distances around the ring.
    b. Direct pointers to every node in the group.
    c. A local cache of recently resolved keys to avoid routing entirely.
    d. Pointers to a fixed set of well-known root nodes that handle all lookups.

Correct answer: (a) Pointers at exponentially spaced intervals mean each hop can skip roughly half the remaining ring distance, reducing total hops to O(log n).

Incorrect answers:

b. A table with a pointer to every node would scale as O(n), defeating the purpose of the finger table.

c. Caching is not part of the Chord finger table; the table stores routing pointers, not resolved values.

d. Chord has no root nodes; it is fully decentralized, and the finger table points to peers on the ring.

13. In Dynamo, when a write cannot reach its preferred replica, the coordinator sends it to another node. What does the receiving node do with the data?
    a. It merges the write with any conflicting version it already holds.
    b. It holds the write temporarily and delivers it to the intended replica once that node recovers.
    c. It immediately replicates the write to additional nodes to maintain *N* copies.
    d. It updates its own position on the ring to cover the unavailable node's key range.

Correct answer: (b) The receiving node stores the write as a temporary stand-in and delivers it to the intended replica when that replica comes back online, restoring full replication.

Incorrect answers:

a. Conflict detection uses vector clocks and is handled at read time; the receiving node stores the write, it does not merge it.

c. The coordinator already selected the receiving node to maintain availability; adding yet another replica (or replicas) is not part of the hinted handoff mechanism.

d. Nodes do not reassign ring positions on failure; the ring topology changes only through explicit joins and departures.

14. What property of *consistent hashing* minimizes disruption when a node joins or leaves a DHT?
    a. Keys are distributed uniformly across all nodes, so no single node is a hotspot.
    b. The hash function maps each key to exactly one node, regardless of the cluster's size.
    c. Nodes store redundant copies of neighboring ranges to handle node departures.
    d. Only keys in the range affected by the membership change need to move.

Correct answer: (d) A new node takes over a contiguous range from its neighbor; a departing node transfers its range to a neighbor. Only the keys in that range move, leaving all other assignments unchanged.

Incorrect answers:

a. Uniform distribution is a desirable property of consistent hashing, but it does not explain why membership changes are minimally disruptive.

b. Deterministic mapping describes how a key is located, not why few keys need to move when the cluster changes.

c. Redundant storage of neighboring ranges is not a property of consistent hashing; replication is a separate layer added by systems like Dynamo.

15. Dynamo attaches vector clocks to stored values. What problem do *vector clocks* solve in this context?
    a. They allow the system and application to detect when concurrent writes have produced conflicting versions.
    b. They prevent two clients from writing the same key at the same time.
    c. They establish a global write order so that the most recent version can always be identified.
    d. They replace quorum voting by letting replicas independently agree on the correct version.

Correct answer: (a) Vector clocks track the causal history of each version, allowing Dynamo to detect divergence when concurrent writes cannot be ordered and flag the conflict for the application to resolve.

Incorrect answers:

b. Dynamo does not prevent concurrent writes; it detects and surfaces the resulting conflicts after the fact.

c. Vector clocks capture causality, not wall-clock order; they identify that two versions are in conflict, not which one is newer.

d. Quorum parameters R and W are independent of vector clocks; they govern how many replicas must respond, not how conflicts are resolved.

16. Why did Dropbox introduce a dedicated *notification server*?
    a. Clients polling for changes caused version conflicts across devices.
    b. Polling introduced long synchronization delays that block-level transfers could not fix.
    c. Most poll responses were empty, wasting server capacity for no useful work.
    d. Polling overwhelmed the block store before changes could be confirmed.

Correct answer: (c) With many clients polling simultaneously and most responses returning nothing new, a notification server was introduced so clients hold a persistent connection and are pushed a message only when a change actually occurs.

Incorrect answers:

a. Polling does not cause version conflicts; deduplication via content hashing addresses storage consistency, not polling overhead.

b. Synchronization delay is bounded by poll interval, not transfer speed; the notification server eliminated unnecessary round trips regardless of block size.

d. Polls went to the metadata server, not the block store; the problem was the volume of empty responses, not block store load.

17. The GFS master does not persistently store chunk locations. Why not?
    a. Chunk locations change too frequently for disk writes to keep up.
    b. The operation log records chunk locations as a side effect of logging writes.
    c. Chunk locations are rebuilt from chunkserver reports at startup, making persistence unnecessary.
    d. Storing chunk locations on disk would create a consistency hazard with the in-memory state.

Correct answer: (c) At startup, the master polls all chunkservers to reconstruct the chunk location map, so there is nothing to persist or recover.

Incorrect answers:

a. Chunk locations change when chunkservers fail or are added, not continuously; this is not the reason persistence is avoided.

b. The operation log records namespace mutations, not chunk-to-server mappings.

d. The master's in-memory state is authoritative; there is no consistency hazard in not persisting a value that is always rebuilt from live sources.

18. In GFS's *two-phase write protocol*, why is data transfer separated from the write request?
    a. To send data once and let the primary order the write.
    b. To let replicas verify checksums before writing.
    c. To keep metadata and file data on separate channels.
    d. To give secondaries time to reserve disk space.

Correct answer: (a) Pipelining in phase 1 ensures each network link carries the data exactly once; the primary's lease in phase 2 serializes concurrent mutations so all replicas apply them in the same order.

Incorrect answers:
b. Checksum verification is a separate fault detection mechanism and is not part of the two-phase write protocol.
c. Separating metadata from data is a core GFS design principle, but it describes the master/chunkserver split, not the two-phase write protocol.
d. Disk space management is handled independently by chunkservers and is not part of the write protocol.

19. What rule does *two-phase locking* impose on lock acquisition and release?
    a. A transaction must acquire all the locks it will need before any operation begins.
    b. A transaction must hold all locks until it commits.
    c. A transaction may not acquire any new locks after releasing any lock.
    d. A transaction may hold at most one write lock at a time to prevent deadlock.

Correct answer: (c) In 2PL, once a transaction enters the shrinking phase by releasing a lock, it cannot acquire new ones; this prevents the interleavings that produce non-serializable schedules.

Incorrect answers:

a. Acquiring all locks upfront is not required by 2PL; locks are acquired as needed during the growing phase.

b. 2PL imposes no rule about the ordering of read and write lock acquisition relative to each other.

d. 2PL places no limit on the number of write locks a transaction may hold simultaneously.

20. Why do reads under MVCC never block?
    a. Reads return data from a committed version, regardless of ongoing writes.
    b. Read locks are granted immediately because MVCC allows multiple transactions to hold read locks simultaneously.
    c. MVCC routes all reads to a dedicated read replica that receives writes only after they are fully committed.
    d. Reads are served from the transaction's private workspace, isolating them from concurrent activity.

Correct answer: (a) Because each transaction reads from a stable snapshot rather than the current live state, there is no need to wait for concurrent writers to finish.

Incorrect answers:

b. MVCC avoids locks for reads entirely; it does not rely on shared read locks.

c. Read replicas are a separate architectural choice; MVCC operates on a single store by maintaining multiple versions of each item.

d. Private workspaces describe OCC; under MVCC, reads go directly to the versioned store, not a private copy.

21. In 2PC, a participant has voted YES, but the coordinator crashes before sending its decision. What must the participant do?
    a. Abort after a timeout, since the coordinator's failure means the transaction cannot complete.
    b. Contact the other participants and commit if a majority of them also voted YES.
    c. Commit, since a YES vote means the participant has already verified that they can commit.
    d. Block with locks held until the coordinator recovers.

Correct answer: (d) A participant that has voted YES is in the uncertain state: it has promised to commit but cannot do so without the coordinator's decision, so it must block with locks held until the coordinator recovers.

Incorrect answers:

a. Aborting unilaterally would violate atomicity if the coordinator had already sent COMMIT to other participants before crashing.

b. 2PC requires unanimous agreement, not majority; and participants in the uncertain state do not know each other's votes.

c. Unilateral commit would violate atomicity if the coordinator had decided to abort due to another participant's NO vote.

22. Which of the following is *not* an ACID property?
    a. Atomic.
    b. Concurrent.
    c. Isolated.
    d. Durable.

Correct answer: (b) The ACID properties are Atomicity, Consistency, Isolation, and Durability; Concurrent is not one of them.

Incorrect answers:
a. Atomicity is the A in ACID: a transaction either commits fully or leaves no changes behind.
c. Isolation is the I in ACID: concurrent transactions do not observe each other's partial results.
d. Durability is the D in ACID: committed changes survive crashes.

23. What is the main reason *Three-Phase Commit* is not used in practice?
    a. A network partition can still lead to inconsistent outcomes.
    b. It adds too many messages to each transaction.
    c. It requires synchronized clocks on all nodes.
    d. It does not scale to a large number of participants.

Correct answer: (a) PC assumes bounded message delay; a partition during the PreCommit phase can cause nodes to reach different decisions, producing split-brain behavior.

Incorrect answers:

b. The extra phase adds overhead but this is not the disqualifying reason; the split-brain problem under partitions is.

c. 3PC does not require synchronized clocks; it assumes bounded message delay, which is a different and equally unrealistic assumption in real networks.

d. 3PC's limitations are about network assumptions, not participant count.

24. What does *consistency* mean in the CAP theorem?
    a. Serializability: concurrent transactions produce results equivalent to some serial execution.
    b. The database enforces integrity constraints between valid states.
    c. Linearizability: every read returns the most recently committed write.
    d. All replicas converge to the same value once updates stop arriving.

Correct answer: (c) The C in CAP means linearizability, the guarantee that reads always reflect the latest committed write in real-time order.

Incorrect answers:

a. Serializability is a property of transaction schedules and is independent of linearizability; the two can hold separately.

b. This is the meaning of consistency in ACID, which refers to application-level data integrity, not the distributed systems definition used in CAP.

d. Convergence after updates stop is eventual consistency, the weakest consistency model, not the C in CAP.

25. Why was PACELC created?
    a. CAP treats consistency as binary; PACELC recognizes varying degrees of consistency.
    b. CAP focuses on partitions; PACELC accounts for node failures separately.
    c. CAP applies only to replicated systems; PACELC extends it to non-replicated systems.
    d. CAP does not address normal operation; PACELC adds the latency-consistency tradeoff for a healthy network.

Correct answer: (d) CAP only describes the consistency-vs-availability tradeoff during a partition; PACELC observes that even when the network is healthy, choosing stronger consistency requires coordination that adds latency.

Incorrect answers:

a. CAP does not treat consistency as binary in this sense; PACELC's contribution is the normal-operation latency tradeoff, not a richer consistency scale.

b. PACELC does not introduce a separate treatment of node failures; it extends CAP specifically by addressing normal-operation behavior.

c. CAP applies broadly to distributed systems; PACELC does not change the scope, it adds the latency dimension for normal operation.

The end.