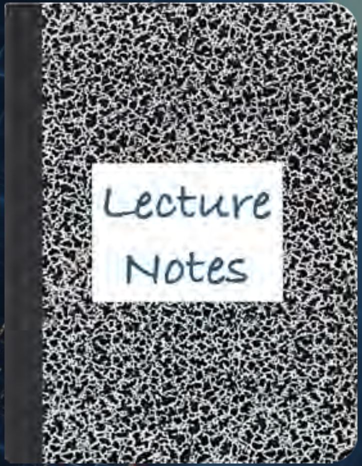
CS 417 – DISTRIBUTED SYSTEMS

Week 4: Exam 1 Past Q&A and Info



Paul Krzyzanowski

Exam 1: February 23, 2026

Expect around 25 multiple-choice questions on content from the first four weeks of class:

1. **Introduction & Networks**
2. **Remote procedure calls and web services**
3. **Clock synchronization & logical clocks**
4. **Group communication, mutual exclusion & leader election**

Past exams are posted on the website (*Exam Info* tab)

– But some content and terminology have changed over time

Review

This is not a review of all the material

- It highlights some of the key topics you should be familiar with

- Goes over some sample Q&A from old exams to give you an idea of what kind of questions to expect

Introduction & Networks

Key concepts you should know

- Properties and key issues of a distributed system
  - Definitions
  - Vertical vs. horizontal scaling, Moore's law
  - Transparency
  - Caching vs. replication
- Failures
  - Availability vs. reliability, series vs. parallel systems
  - Types of failure: fail-stop, fail-restart, partitions, omission, byzantine)
- Networks
  - Layers, Internet design principles
  - Latency vs. throughput, asynchronous networks
  - TCP, UDP, QUIC: key distinctions

Question 1

A *single system image* is:

(a) Deploying the same operating system across all computers in a distributed system.

(b) An environment that hides the fact that there is a distributed system.

(c) Running multiple processes on one computer instead of using a distributed system.

(d) Having a single operating system distributed across multiple computers and coordinating all activity.

Single System Image:

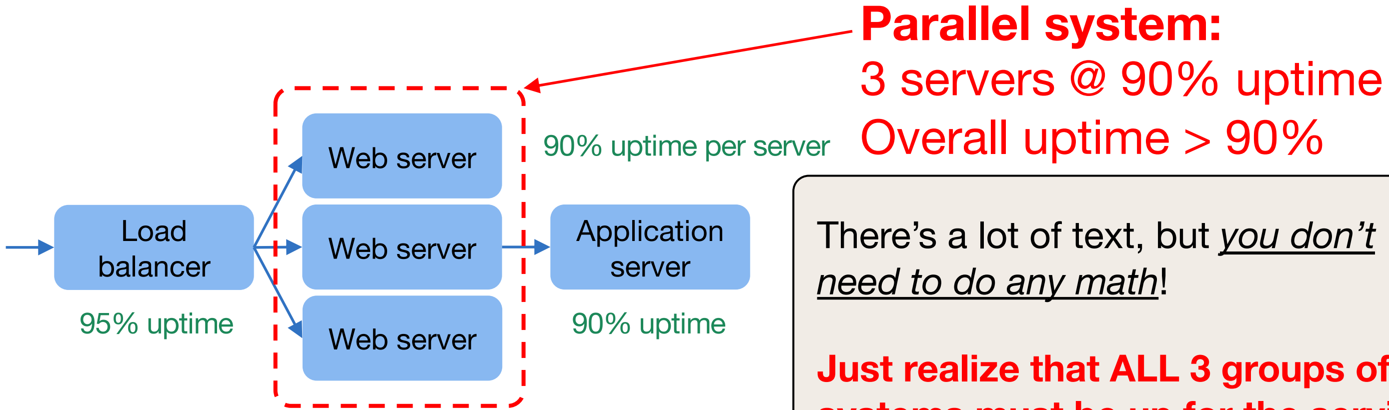
A collection of independent computers that appears as a single system to the user(s)

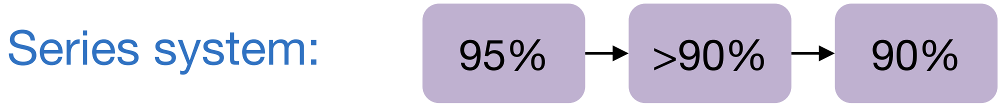
Users aren’t aware that components are remote or replicated.

Question 2

In an attempt to improve fault tolerance, a company deployed three replicated web servers. Requests enter through a load balancer and are routed to the web servers, which contact a back-end application server. The uptime of all the servers is 90% while the uptime of the load balancer is 95%. What is the expected uptime of the overall environment?

(a) 85%
(b) 92.5%
(c) 94.9%
(d) 99.9%





There's a lot of text, but *you don't need to do any math*!

Just realize that ALL 3 groups of systems must be up for the service to work

The worst availability is the application server (90%), so the **overall availability will be <90%**

Question 3

The main challenge with *fail-restart* behavior is:

(a) A node may resume operations with stale data.
(b) The node might not give advance warning that it will fail.
(c) The node may fail again after restarting.
(d) A failed node can create a network partition.

Stale state means the node will have outdated info:

Old leader, outdated group membership, missed updates, outdated computation progress.

Other systems may have already taken over for the node, but it won’t know that.

Question 4

Why might UDP be favored over TCP for real-time applications such as online gaming or voice over IP?

(a) UDP ensures in-order delivery of all packets.
(b) UDP establishes a long-lasting connection for data transfer.
(c) The occasional loss of packets may be acceptable in exchange for reduced latency.
(d) UDP has mechanisms to provide constant bandwidth for streaming data.

- UDP does not have the delay incurred in the 3-way handshake TCP uses to set up a connection – packets can be sent immediately
- UDP does not wait to transmit packets in the hope that a process will provide more data since it's more efficient to send a smaller # of bigger packets
- UDP doesn’t retransmit corrupt/lost packets, but some applications wouldn’t want to wait for retransmitted data since it would arrive too late

Question 5

Which layer of the OSI model is responsible for routing packets between different networks?

(a) Transport layer.

(b) Network layer.

(c) Data link layer.

(d) Session layer.

Transport layer: enables application-to-application communication (sockets)

Data link layer: enables communication between devices on a local area network

Session layer: establish, maintain, and terminate communication sessions between two devices – manage the exchange of data

RPC and Web Services

Key concepts you should know

- Remote procedure calls
  - Stub functions, interface definition language (IDL), stub generation
  - Marshalling, serialization, challenges, versioning, timeouts/deadlines
  - Idempotent operations
  - First gen RPC (ONC & DCE), Object RPC (DCOM, RMI, RPyC)
  - Garbage collection
- Web Services
  - REST vs. XML-RPC and SOAP vs. gRPC
  - Benefits of HTTP/2
  - Request IDs and observability

Question 6

What role does the *client stub* play in remote procedure calls?

(a) It acts as a server that hosts the remote procedure.

(b) It provides a local interface to the remote function and handles communication.

(c) It is an automatically generated template for remote procedures that programmers fill in with their own code.

(d) It provides encryption and security measures for RPC data.

Client stub = client-side proxy to remote services

Provides a local interface for remote functions.

Question 7

Why are IDL (Interface Definition Language) compilers important in an RPC framework?

(a) They translate the interface specifications into client and server stubs for a specific environment.

(b) They provide a runtime environment for executing remote procedure calls.

(c) They allow server functions to be defined and compiled in a language-neutral manner.

(d) They enable data to be delivered securely to the correct server process.

An IDL defines remote interfaces and their data types.

It enables the automatic creation of client and server stubs so that clients can communicate with the remote implementations on the server.

Question 8

What is a significant advantage of RESTful web services over SOAP-based web services?

(a) REST is more consistent, requiring the use of XML for all its responses.

(b) RESTful services are stateful, making interactions more flexible.

(c) REST is generally simpler, using standard HTTP methods, and can return multiple data formats (like XML or JSON).

(d) REST mandates a strict communication contract defined via WSDL.

RESTful services are usually simpler to implement since they rely on standard HTTP methods (GET, POST, PUT, DELETE)

Data formats are not defined in REST, so it's up to applications, which may use XML, CSV, JSON, protocol buffers, …

JSON dominates for Internet services; protocol buffers are common for internal services

Clocks

Key concepts you should know

- Clock Synchronization
  - Drift, offset, adjustments: slewing vs. stepping
  - Cristian’s algorithm
  - NTP: goals, strata, NOT the formula
  - PTP (precision time protocol): goals, NOT the formula
- Logical Clocks
  - Happened-before, causal ordering (partial ordering), total ordering, concurrent events
  - Lamport timestamps
  - Vector clocks (comparisons)
  - Hybrid logical clocks: goals, role of the two components

Question 9

Which of the following best defines *clock drift* in the context of distributed systems?

(a) The difference in time between two clocks at a single instant.

(b) The rate at which a clock gains or loses time relative to a reference clock.

(c) The delay between requesting and receiving the time from a server.

(d) The cumulative adjustment made to a clock after synchronization.

Clock drift is the growing discrepency between two clocks, typically caused by variations and slight inaccuracies in the clock's oscillator.

Question 10

Machine A sends a time request to B at *time=11*. B receives it at *time=6* and returns a response containing a timestamp of 8 at *time=9*. A receives the response at *time=17*. Using Cristian's algorithm, to what value does A set its clock?

(a) 7.5
(b) 10.5
(c) 11
(d) 20

Event	A's time	B's time
Request sent	11	
Request received		6
Returned timestamp		8
Response sent		9
Response received	17	

Cristian's algorithm:

Set time to ½(delay) + server_timestamp

Time = ½(17 – 11) + 8 = ½(6) + 8 = 3 + 8 = 11

Question 11

With no additional information at A, what would Cristian's algorithm state is the error of the new timestamp?

(a) ±0.5

(b) ±1.5

(c) ±3

(d) ±6

Event	A's time	B's time
Request sent	11	
Request received		6
Returned timestamp		8
Response sent		9
Response received	17	

Error = ±½ (round_trip_time – best_case_round_trip_time)

We don't know the best-case round-trip time, so assume 0 for worst-case error

Error = ±½(17 – 11) = ±½(6) = ±3

Question 12

What does the term *stratum* refer to in the Network Time Protocol (NTP)?

(a) An NTP server's position in the hierarchy of time sources, representing its distance from the reference clock.

(b) The rate at which the local clock drifts from the reference time.

(c) The specific time zone in which the NTP server operates.

(d) The security mechanism used to encrypt time synchronization packets.

Stratum = level of hierarchy in the synchronization network.

Stratum 0 = actual time source

Stratum 1 = server that syncs from stratum 0
(`time.google.com`, `time.apple.com`, `time.facebook.com`, ...)

Stratum 2 = server that syncs from stratum 1

Question 13

Three processes, x, y, and z exchange vector timestamps. Which event is concurrent with the timestamp <x:5, y:2, z:1>?

(a)	<x:4, y:2>	<x:4, y:2, z:0>
(b)	<x:5, y:3, z:1>	<x:5, y:3, z:1>
(c)	<x:1, z:0>	<x:1, y:0, z:0>
(d)	<x:6>	<x:6, y:0, z:0>

Vector timestamps may be positional (e.g., P0 in element 0, P1 in element 1, …).
In real life, they are usuallya list of (process:timestamp tuples)

If a node is missing in a vector timestamp, it is implicitly 0: we haven't seen any messages from that node or any messages from some process that knows about it.

Causal dependencies: element-by-element comparison

(a) Every element in <x:4, y:2, z:0> ≤ the corresponding element in <x:5, y:2, z:1>

(b) Every element in <x:5, y:3, z:1> ≥ the corresponding element in <x:5, y:2, z:1>

(c) Every element in <x:1, y:0, z:0> ≤ the corresponding element in <x:5, y:2, z:1>

(d) **<x:6, y:0, z:0>** is neither ≤ nor ≥ than **<x:5, y:2, z:1>**

Groups, Elections, and Mutual Exclusion

Key concepts you should know

Groups

- **IP Multicast:** IGMP vs PIM, PIM Dense Mode vs. PIM Sparse Mode
- **Multicast Reliability**: unreliable, best-effort reliable, reliable, durable
- **Multicast ordering**: unordered, single-source FIFO, causal, total, synchronous
- **Failure detection**: push/pull heartbeating, accrual failure detector
- **Group membership** (virtual synchrony): view, view change operations

Mutual Exclusion

- Algorithms: Centralized, Lamport's, Ricart-Agrawala, Token Ring

Leader Election

- Algorithms: Bully & Ring

Question 14

What is the primary function of the *Internet Group Management Protocol* (IGMP)?

(a) Assigning unique multicast IP addresses to nodes in that need to initiate a multicast stream.

(b) Allowing hosts to report their multicast group memberships to their routers.

(c) Ensuring efficient delivery of multicast traffic to the public internet.

(d) Dynamically establishing multicast routing paths between the source and the destination.

IGMP is used only on the local area network for hosts to tell routers on the network that they are interested in specific multicast addresses.

Routers then talk with each other using PIM (Protocol Independent Multicast)

Question 15

In ***dense mode IP multicast***, what mechanism prevents unnecessary traffic from flowing in areas of the network with no interested receivers?

(a) Explicit *Join* request messages.

(b) Use of a Rendezvous Point (RP).

(c) *Prune* messages.

(d) Reverse Path Forwarding (RPF).

Dense mode IP multicast (PIM-DM) floods the network.

Routers can send *Prune* messages to other connected routers to tell them they're not interested in certain multicast addresses.

(c) Join and *rendezvous points* are used in sparse-mode multicast (PIM-SM).

(d) Reverse Path Forwarding is used to prevent forwarding loops.

Question 25

Why is the ***flush*** operation important in virtual synchrony?

(a) It enables processes to discard duplicate messages.

(b) It ensures that all processes receive messages in the same sequence and at the same time.

(c) It allows nodes to process messages in parallel.

(d) It ensures that all messages are delivered within a group view and not across multiple views.

A ***flush*** is a way for a process to tell other group members that it has no more unstable messages queued up.

When a process sent a *flush* and received a *flush* from all group members, it knows that there are no more unstable messages and the view change is complete.

Question 26

What is a key difference between *Lamport*'s mutual exclusion algorithm and the *Ricart and Agrawala* algorithm?

(a) Lamport's algorithm uses a logical clock, while Ricart and Agrawala's uses physical clocks.

(b) Lamport's algorithm is token-based, while Ricart and Agrawala's algorithm is not.

(c) All processes immediately acknowledge a request in Lamport's algorithm, but not always in Ricart and Agrawala.

(d) Lamport's algorithm relies on a coordinator, while Ricart and Agrawala is distributed.

(a) They both use logical clocks – unique Lamport timestamps

(b) Neither use tokens

(d) Neither requires a coordinator

Ricart & Agrawala is based on holding back an acknowledgment if you are in a critical section (or if you want it and have an earlier timestamp)

Question 27

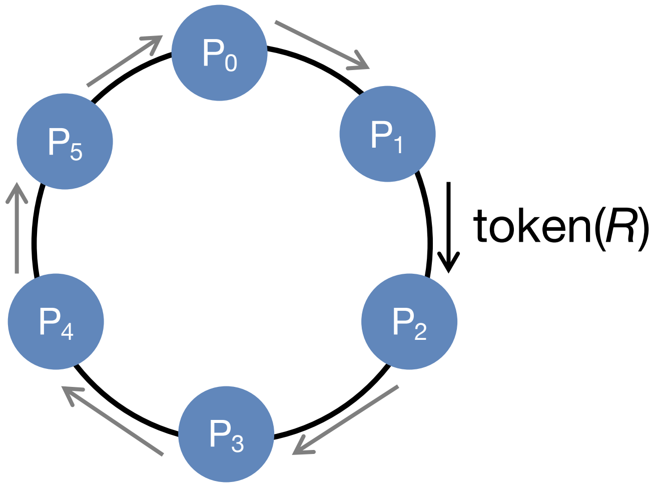
How does the ***token ring algorithm*** for distributed mutual exclusion work?

(a) A process passes a message to its neighbor that grants access to a resource.

(b) A central coordinator assigns tokens to processes based on their priority.

(c) Processes send requests to all other processes in the system to request a token for a resource.

(d) Processes use timestamps to determine the order in which tokens should be granted.



- A message, called a "**token**" for a resource *R* is passed from one system to its neighbor
- When a system receives the token for *R*:
  - If the process needs to access the *R*, it forwards the token when it's done
  - If it doesn't need to access *R* then it forwards the token immediately

Question 28

What is the primary communication pattern used in *Ricart and Agrawala*'s *mutual exclusion algorithm*?

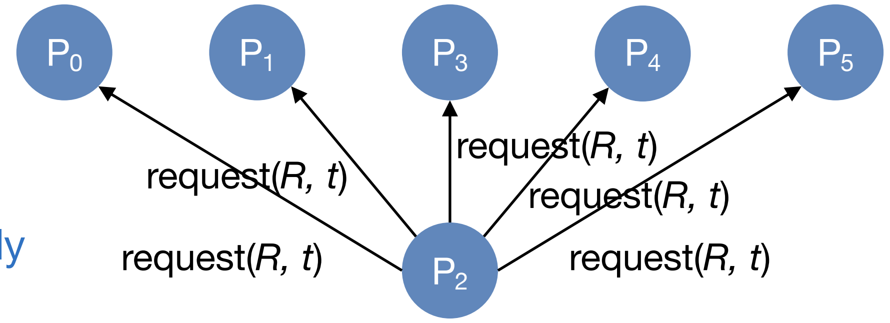
(a) Processes send requests to a central coordinator.

(b) Processes communicate directly only with the process holding the shared resource.

(c) Processes communicate only with their immediate neighbors in the system.

(d) Processes send requests to all other processes in the system.

- Ricart & Agrawala mutual exclusion
  - Send a request for a resource *R* (with a timestamp) to the entire group
  - Wait for everyone to respond with "ok"
  - If a process is using *R*, it will respond only when it is done



Question 29

Which of the following best describes the *Bully* election algorithm?

(a) Processes with higher IDs win over those with lower IDs to become the leader.

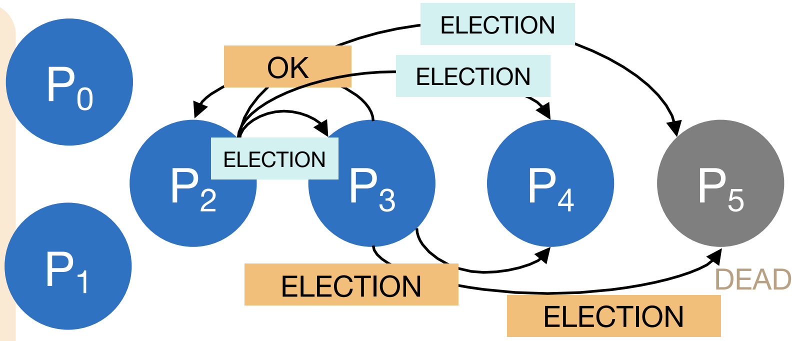
(b) Processes forward election messages to their neighbor to select a leader.

(c) Processes use timestamps to determine the order of election messages.

(d) Processes solicit votes from other processes and the one with the most votes wins.

Bully election algorithm

- If a process P_n detects a dead coordinator:
  - Send a message to all higher-numbered processes
  - If any of them respond, P_n will not be the leader
  - If none respond, that means P_n is the highest-numbered live process and announces itself as the leader
- If a process P_n receives an *ELECTION* message:
  - It sends an OK response back
  - It holds an election by sending an *ELECTION* message to all higher-numbered processes.



The End