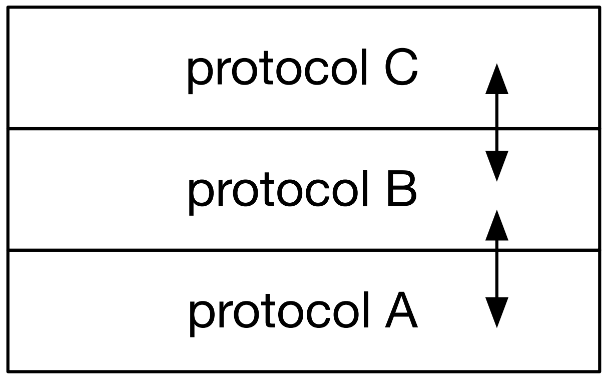
Practice questions

416 2020 W2 (Winter 2021)

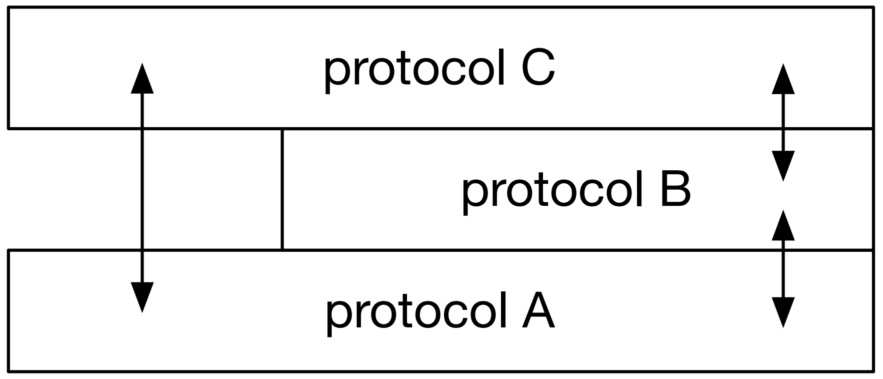
These questions are intended to simulate the final exam. They cover previous lecture/assignment material that is fair game for the final exam.

PQ 1

- You are designing a protocol stack. You have narrowed down your design to two choices. And, you know that the specification for protocol C is likely to change. Which stack design should you use?



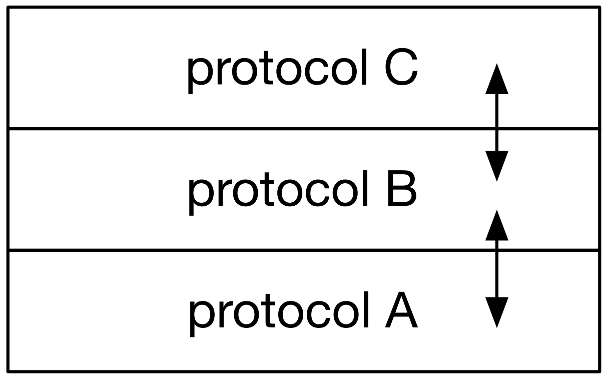
(a)



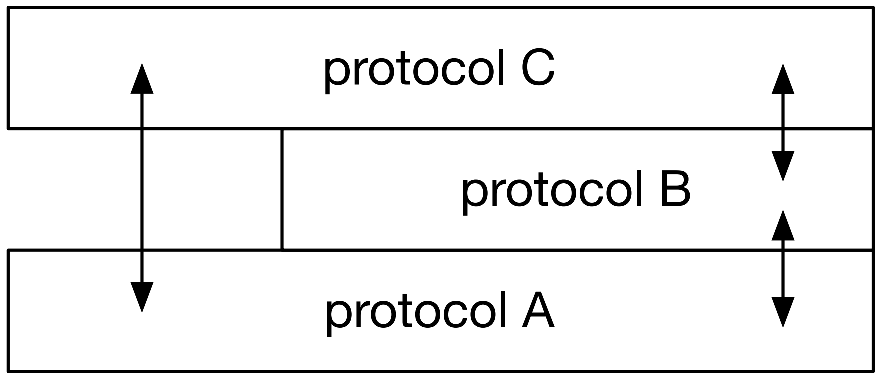
(b)

PQ 1

- You are designing a protocol stack. You have narrowed down your design to two choices. And, you know that the specification for protocol C is likely to change. Which stack design should you use?



(a)



(b)

- If protocol C changes then only protocol B would need to adapt, and not A

PQ 2

- A network element can inspect any of the protocols present in the packet. So, **why not** build e.g., a switch that is aware of HTTP and have it route packets based on HTTP information that it can extract from the packet?

PQ 2

- A network element can inspect any of the protocols present in the packet. So, **why not** build e.g., a switch that is aware of HTTP and have it route packets based on HTTP information that it can extract from the packet?

- Expensive! Line-rate HTTP processing requires more memory/cpu. Also requires interpreting the protocol below HTTP (e.g., TCP/IP)

- Higher-level protocols change, often more frequently (HTTP 2.0)

- More software can access/manipulate HTTP content (not just your OS TCP/IP stack). *Requires more robustness/more security considerations.*

- But, it's not impossible! See "software middleboxes" or "network function virtualization"

PQ 3

- You plan to disrupt the RPC concept by not only sending arguments to the remote procedure, but also sending the *procedure* itself (as a lambda fn). What challenges do you expect with this idea?

PQ 3

- You plan to disrupt the RPC concept by not only sending arguments to the remote procedure, but also sending the *procedure* itself (as a lambda fn). What challenges do you expect with this idea?
    - Defining invocation semantics: at most once/at least once. What happens on failures?
    - Defining the capabilities of the procedure: can it read files? Can it open connections/send data?
    - Related: defining allowable side effects, if any
    - Guaranteeing determinism (procedure should probably run identically regardless of host server). Have to determinize calls to random, env state like files. Have to provide environment that is identical across machines (e.g., runtime language-based VM like a JVM).
    - Encoding of arguments (as in RPC)
    - Encoding of the procedure + env state that it needs besides args

PQ 5

- On the client-side NFS caches only those parts of the file that the local processes have read/written. *What are the pros and cons of NFS caching files in their entirety?*

PQ 5

- On the client-side NFS caches only those parts of the file that the local processes have read/written. *What are the pros and cons of NFS caching files in their entirety?*

- Pros: **performance** (more reads handled locally, don't hit the server), **scales better** (can support more clients), **file access is faster** (assumes: client cache faster than server cache; client memory faster than server disk), **robust to server crashes** (client can read cache instead of going to server)

- Cons: **more resources at client** (client needs space for the cache), **scales worse** (with larger files), **worse\complex consistency** (there is less sync. between clients), **security** (trust clients with *all* file data),

PQ 5

- On the client-side NFS caches only those parts of the file that the local processes have read/written. *What are the pros and cons of NFS caching files in their entirety?*
    - Pros:
        - Great performance when client workload has file locality (e.g., sequentially read the file)
        - Disconnected operation!?
    - Cons:
        - Uses up client memory/disk space
        - Without other changes, this will likely make consistency worse..

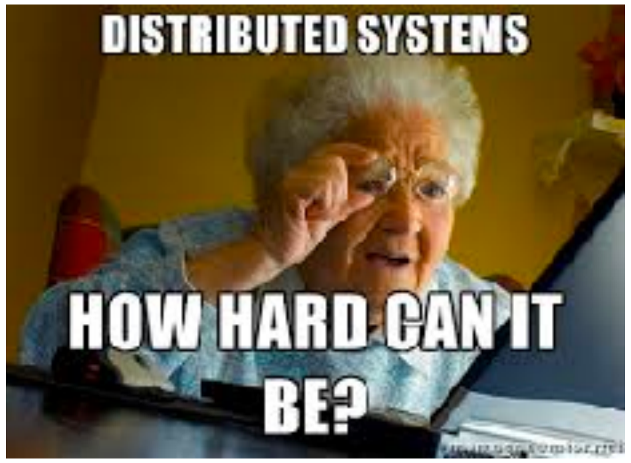
PQ 4

- True/False: *In a practical system that uses proof-of-work, the checking of the proof has to be easy relative to the generation of the proof.*

PQ 4

- True/False: In a practical system that uses proof-of-work, the checking of the proof has to be easy relative to the generation of the proof.

- True!



PQ 5

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

A. NFS

B. AFS

PQ 5

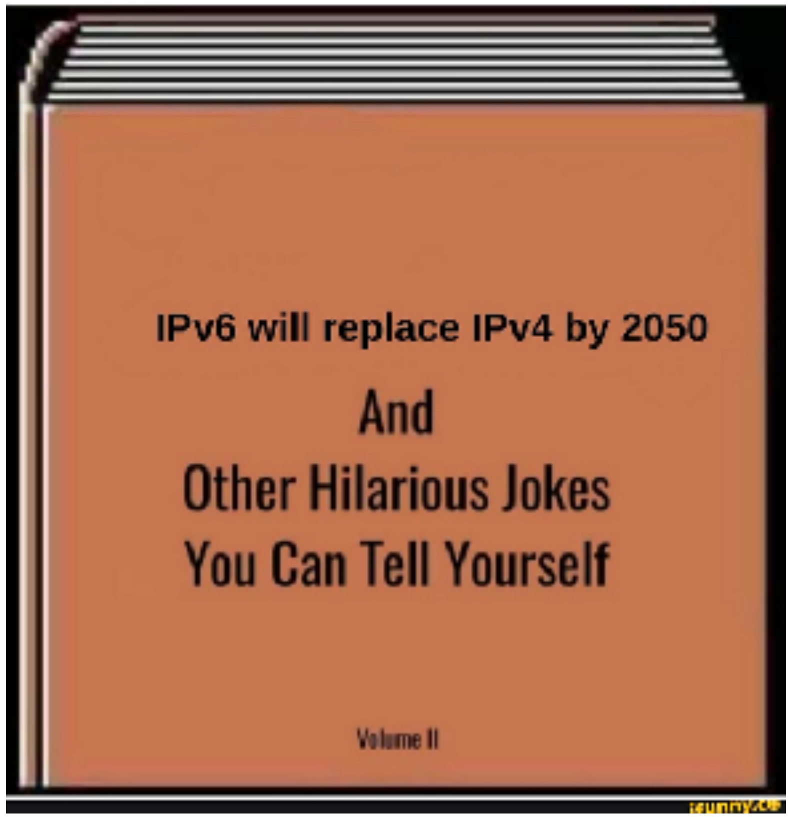
- Which file system can support more clients, given a server that runs on identical hardware? [Choose one answer]

  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 6

- A CDN can improve which of the following for the client:
  - Latency
  - Security
  - Throughput
  - Consistency
  - Availability



PQ 6

- A CDN can improve which of the following for the client:
  - Latency (CDNs nodes closer to client)
  - Security (2 administrative domains)
  - Throughput (CDN nodes more lightly loaded)
  - Consistency (nope)
  - Availability (CDN nodes in same region as client)

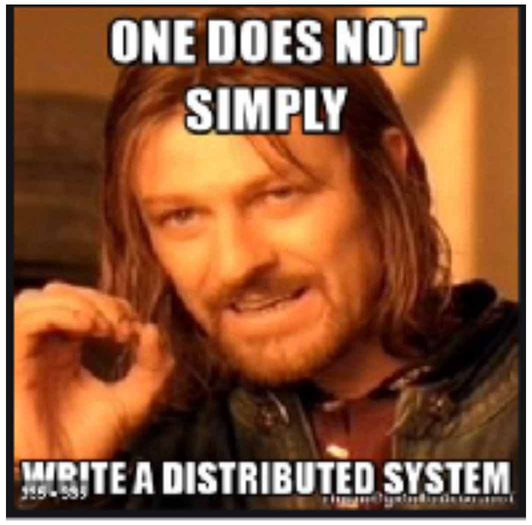
PQ 7

- Compared to a central file hosting server, a BitTorrent swarm has which of the following features:
  - Higher scalability
  - Higher availability
  - Higher performance

PQ 8

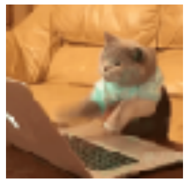
- Compared to a central file hosting server, a BitTorrent swarm has which of the following features:
  - Higher scalability (supports more clients)
  - Higher availability (can survive more failures)
  - Higher performance (perf scales with peers)

PQ 9



- Which of the following statements is false?
  - Gnutella has O(N) search scope
  - Napster has O(1) search scope
  - BitTorrent has O(1) search scope

PQ 9

- Which of the following statements is false?
  - Gnutella has O(N) search scope
  - Napster has O(1) search scope
  - BitTorrent has O(1) search scope <— False
    - BitTorrent has O() scope since it's outside the model of the system! It doesn't provide a lookup function.

PQ 10

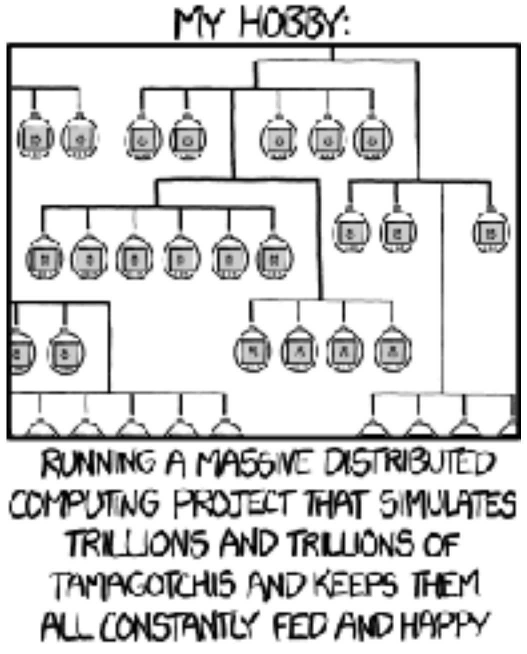
- *"To take control over the BitCoin ledger (e.g., to double spend) an attacker needs to control at least X of the processing power in the network."* What's the right X?
    - 10%
    - 25%
    - 33%
    - 50%
    - 75%

When your boss thanks you for staying late to work but you were just watching the price of Bitcoin and lost track of time

PQ 10

- *"To take control over the BitCoin ledger (e.g., to double spend) an attacker needs to control at least X of the processing power in the network."* What's the right X?
  - 10%
  - 25%
  - 33%
  - **50% (Controlling longest chain requires majority processing power, the "51% attack")**
  - 75%

PQ 11

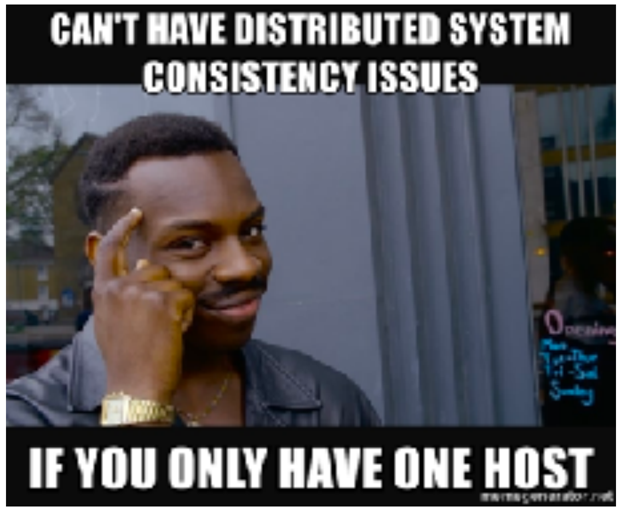

MY HOBBY:
RUNNING A MASSIVE DISTRIBUTED COMPUTING PROJECT THAT SIMULATES TRILLIONS AND TRILLIONS OF TAMAGOTCHIS AND KEEPS THEM ALL CONSTANTLY FED AND HAPPY

- Event A with vector clock timestamp [1,2,3] happened before event B with timestamp [3,2,1].
    - True
    - False
    - Can’t tell from timestamps alone

PQ 11

- Event A with vector clock timestamp [1,2,3] happened before event B with timestamp [3,2,1].
  - True
  - **False: A and B are concurrent according to their vector clocks**
    - [1,2,3] <=> [3,2,1]
    - **0: 1 < 3**
    - 1: 2 == 2
    - **2: 3 > 1**
    - Because of index 0 and 2, the v timestamps are not comparable. No comparable **=>** underlying events associated with these timestamps are concurrent.
  - Can't tell from timestamps alone

PQ 12


CAN'T HAVE DISTRIBUTED SYSTEM CONSISTENCY ISSUES
IF YOU ONLY HAVE ONE HOST

You are attempting to subvert your co-worker's machine by selecting a RAID level that wastes the most amount of physical space. Which RAID level should you choose?

- RAID 0
- RAID 1
- RAID 4
- RAID 5

PQ 12

You are attempting to subvert your co-worker's machine by selecting a RAID level that wastes the most amount of physical space. Which RAID level should you choose?

- RAID 0 : N
- **RAID 1 : mirroring (N/2 capacity)**
- RAID 4 : N - 1
- RAID 5 : N - 1

PQ 13

Ricart-Agrawala uses which of the following mechanisms?

- Atomic clocks
- NTP-synchronized clocks
- Lamport clocks
- Vector clocks
- Other special ninja type of clock not listed above

PQ 13

Ricart-Agrawala uses which of the following mechanisms?

- Atomic clocks
- NTP-synchronized clocks
- **Lamport clocks : provides mut. exclusion + fairness; requires total order: $e < e'$ implies $L(e) < L(e')$**
- Vector clocks
- Other special ninja type of clock not listed above

PQ 14

What is a blockchain?

A. An eventually consistent data structure

B. A set of distributed protocols

C. A fault tolerant data storage system

D. An implementation of an immutable ledger

E. All of the above

PQ 14

What is a blockchain?

A. An eventually consistent data structure

B. A set of distributed protocols

C. A fault tolerant data storage system

D. An implementation of an immutable ledger

E. All of the above

PQ 15

- Your distributed system was running for 30 days during which time you had two outages: a disk failed and you had to replace it (outage of 3 days), and a faulty OS update had to be reverted (outage of 2 days). How many 9s of availability did your system achieve during this time?

- 0
- 1
- 2
- 3

PQ 15

- You were operating your distributed system for 30 days during which time you had two outages: a disk failed and you had to replace it (outage of 3 days), and a faulty OS update had to be reverted (outage of 2 days).

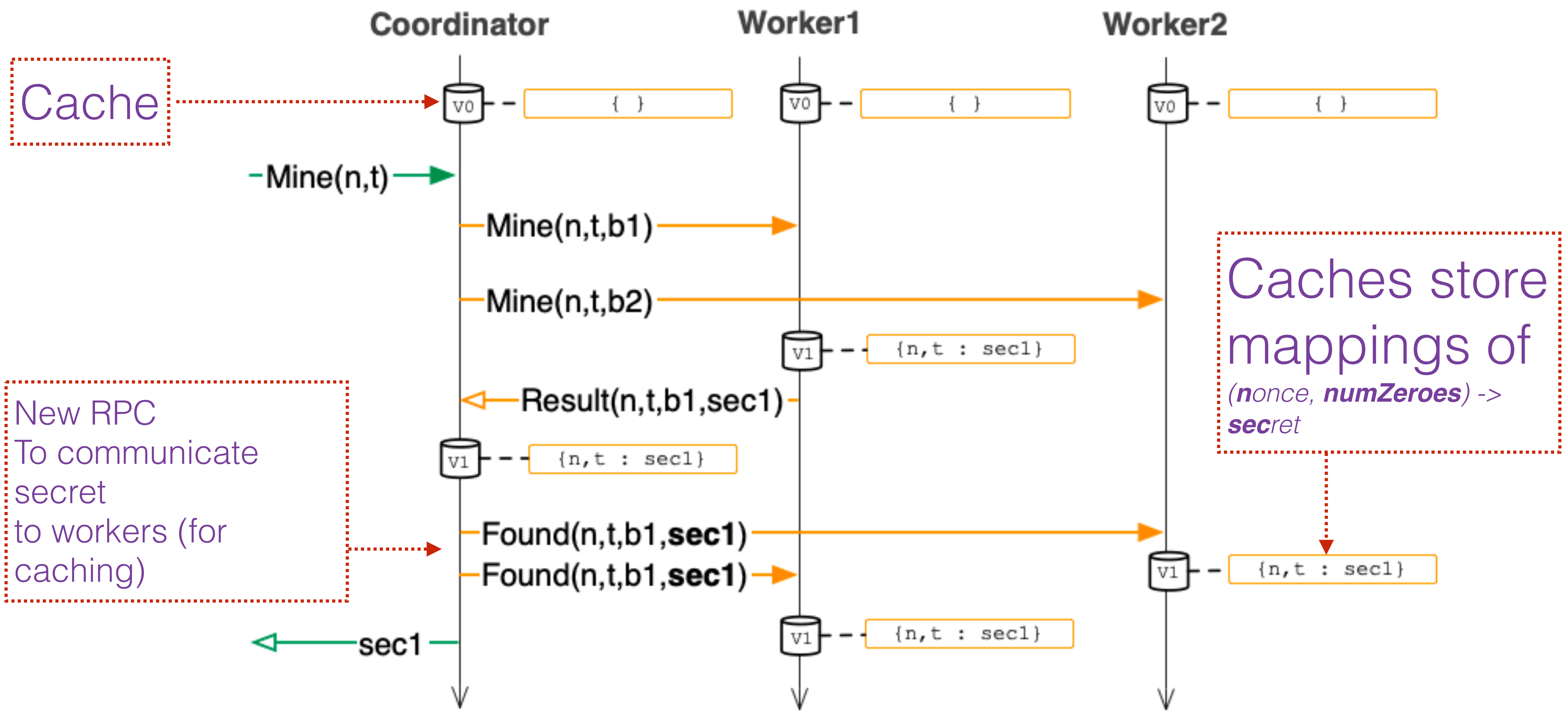
  - How many 9s of availability did your system achieve during this time? => What was your system's availability during this time?

  - Availability = time running / time should have been running

  - = (30-2-3) / 30 = 25 / 30 = 83% => **0 9s of avail.**

PQ 16

- Consider this diagram from A4:

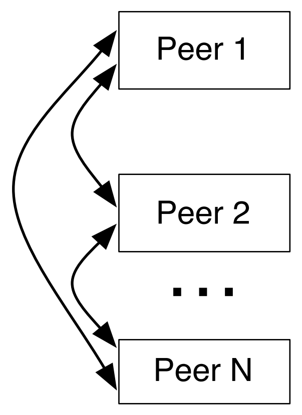


- Q1: How would you define a cache hit and miss?

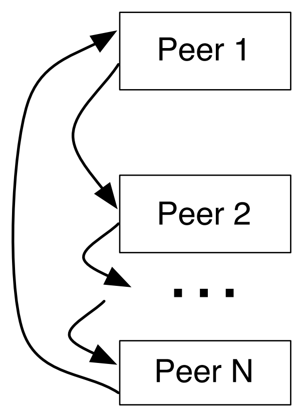
- Q2: How could you use the worker caches to recover from coordinator restarts? How would you have to change the design to enable this?

PQ 17

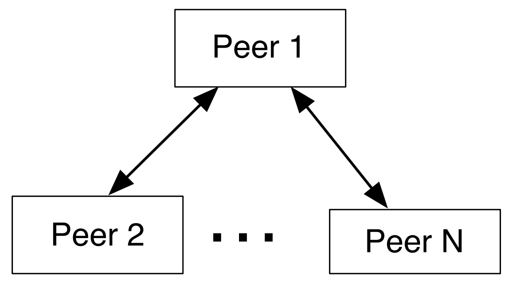
- Consider the following three topologies (e.g., in P1):



(a) all-to-all



(b) linked-list



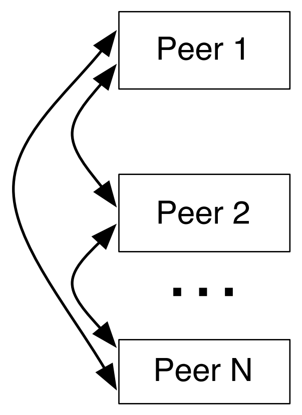
(c) star

- Q1: Which topology makes it easiest for peers to detect peer failures?

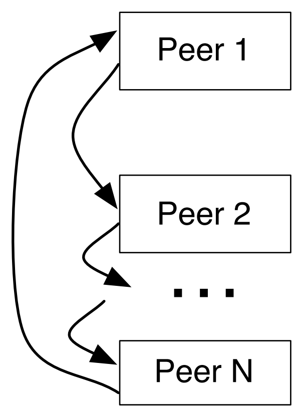
- Q2: Assuming a large N, which topology (on average) impacts the fewest peers when a peer fails?

PQ 17

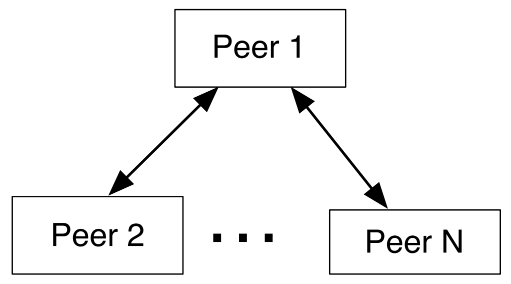
- Consider the following three topologies (e.g., in P1):



(a) all-to-all



(b) linked-list



(c) star

- Q1: Which topology makes it easiest for peers to detect peer failures? **A**

- Q2: Assuming a large N, which topology (on average) impacts the fewest peers when a peer fails? **C**

PQ 18

- RAID uses complement sum for error detection

  A. Yes

  B. No

PQ 18

- RAID uses complement sum for error detection

  A. Yes

  B. No [RAID uses Parity]

PQ 19

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

PQ 19

- 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 20

- Redo logging writes new values to a log
    - True
    - False

PQ 20

- Redo logging writes new values to a log
    - True [redo logs what must be re-done after a failure: committed txn will commit in recovery]
    - False

PQ 21

- Because two phase commit is distributed, it does not require logging at individual nodes
    - True
    - False

PQ 21

- Because two phase commit is distributed, it does not require logging at individual nodes
  - True
  - False : Still txn processing, so need logging to provide txn atomicity. And, nodes might fail and recover, so need logging to remember 2pc state.

PQ 22

- Three-phase commit is a blocking protocol (blocks indefinitely during failures)
    - True
    - False

PQ 22

- Three-phase commit is a blocking protocol (blocks indefinitely during failures)
  - True
  - **False : 3PC trades off safety for liveness, it does not block (indefinitely) during failures. Waits for timeout and continues.**

PQ 23

Given two recorded A4 actions with the following vector clocks:

X: {“worker11":10, "worker3":10, "worker9":10, “coordinator”:409, “client2”: 5}

Y: {“worker11":5, "worker3":7, "worker9":15, “coordinator”:230, “client1”: 10}

Which of the following is correct?

a) X happens before Y

b) Y happens before X

c) X concurrent with Y

d) cannot determine

PQ 23

Given two recorded A4 actions with the following vector clocks:

X: {“worker11":10, "worker3":10, "worker9":10, “coordinator”:409, “client2”: 5}

Y: {“worker11":5, "worker3":7, "worker9":15, “coordinator”:230, “client1”: 10}

Which of the following is correct?

a) X happens before Y

b) Y happens before X

c) X concurrent with Y : simultaneously larger and smaller clock values for individual entries (e.g., X.worker11 < Y.worker11 and X.worker9 > Y.worker9)

d) cannot determine

PQ 24: discuss in a group

The coordinator in A4 blocks until it hears back from each worker: either via a Result RPC or a reply to a Found RPC. This delays the coordinator's reply to the client (especially with many workers!).

You decide to optimize A4 replying to the client when the coordinator has received one Result RPC.

Is this safe to do? What, if anything, could go wrong? And, how would you fix it?

PQ 24: discuss in a group

Rm 1: Safe? Caches may be not up to date. Workers may not be cancelled.

Introduce a cache update/worker cancel *signal* (RPC?). Have workers wait on this signal *before* starting any new computation.

——

Rm 3: Safety concern — client receives response, can now issue another Mine(n,t) and therefore multiple (n,t) in the system (at workers)… isn’t that a problem?

Thought: (n,t1) will be cached at coordinator, so the second (n,t2) will be a hit at the coordinator and will never reach the workers. Except… if t2 > t1. But.. this raises issues with unique nonce assumption at workers.

——

Rm 11: client does not receive the most dominant secret.. but they don’t receive one now either! They just receive the first secret.

Max: but caches could diverge.. is that safe?

PQ 24: discuss in a group

The coordinator in A4 blocks until it hears back from each worker: either via a Result RPC or a reply to a Found RPC. This delays the coordinator's reply to the client (especially with many workers!).

You decide to optimize A4 replying to the client when the coordinator has received one Result RPC.

Is this safe to do? What, if anything, could go wrong? And, how would you fix it?

- Safety: depends on how you define safe. Violates the A4 spec (actions at workers may be concurrent with CoordinatorSuccess). And, after reply to clients with this optimization, the caches in the system may be different/ diverge.
- On node failure this could be a problem (e.g., not all results are replicated). But, since the cache is a perf. optimization.. maybe that's okay: just re-do the computation.
- Complexity: coordinator has to deal with Result RPCs and Found replies for Mine tasks it already responded to: should it still replicate new Results? Yes, if we want caches to converge.
- For worker: nothing changes! Since a worker can't tell when the coordinator replies to client, they continue operating as usual.
- Workers have to worry about concurrent Mine(n,t) requests with different trailing zeroes
- Optimization? Doesn't really save much time if most of the time is spent on mining. For a large t, this is a small optimization. Have to think holistically - how much time is this going to save in the *overall* system (end-to-end latency).

PQ 25

- Paxos is always **safe**, but not always **live**
    - True
    - False

PQ 25

- Paxos is always **safe**, but not always **live**
  - **True : has an execution that never terminates, but will never violate safety conditions**
  - False