

Practice questions

416 2018 W1 (Fall 2018)

PQ 1

- Imagine a scenario in which fdlib (in A1) at node M monitors a node F. At some point, the node F fails and restarts, which causes it to miss 1 heartbeat from M (so it does not reply with an ack to it).
- True/False: fdlib at M will always detect a failure of F in the above scenario.

PQ 1

- Imagine a scenario in which fdlib (in A1) at node M monitors a node F. At some point, the node F fails and restarts, which causes it to miss 1 heartbeat from M (so it does not reply with an ack to it).
- True/**False**: fdlib at M will always detect a failure of F in the above scenario.
- If `lost-msgs-thresh > 1`, then failure detection in this scenario is impossible, assuming no other hbeat loss. (And, `thresh <= 1` is unrealistic).

PQ 2

- The car steering wheel is analogous to which networking concepts that we reviewed in the last lecture?

PQ 2

- The car steering wheel is analogous to which networking concepts that we reviewed in the last lecture?
- **Layering:** the steering wheel hides the details of the driver from car and details of car from driver
- **Thin waist:** tiny interface to provide inter-op between many drivers types and many car types.

PQ 3

- Which of the following is/are an example of a *fate sharing* design?
 - A. Place program state into memory on local machine
 - B. Put keys to house under rug on the porch
 - C. Put baggage on same flight as the passenger

PQ 3

“lose state information for an entity if and only if the entity itself is lost”

- Which of the following is/are an example of a *fate sharing* design?
 - A. Place program state into memory on local machine
 - B. Put keys to house under rug on the porch
 - C. Put baggage on same flight as the passenger

PQ 4

- In A1 design fdlib sends a heartbeat immediately after receiving an ack message. Why is this a bad idea?

PQ 4

- In A1 design fdlib sends a heartbeat immediately after receiving an ack message. Why is this a bad idea?
 - High load for the monitor
 - High load for the node being monitored
 - With RTT of 1ms (LAN), a ~128byte heartbeat is generated every 1ms = 128 KB/s!
(state of the art dial-up throughput!)

Side note about course:

We will rarely focus on performance in the course.
Building distributed systems is challenging enough.
Having them be high-performance is another level.

PQ 4

- In A1 design fdlib sends a heartbeat immediately after receiving an ack message. Why is this a bad idea?
 - High load for the monitor
 - High load for the node being monitored
 - With RTT of 1ms (LAN), a ~128byte heartbeat is generated every 1ms = 128 KB/s!
(state of the art dial-up throughput!)

PQ 5

- In A1 design fdlib sends a heartbeat immediately after receiving an ack message. Why is this a bad idea?
 - High load for the monitor
 - High load for the node being monitored
- **How do we fix this?**

PQ 5

- In A1 design fdlib sends a heartbeat immediately after receiving an ack message. Why is this a bad idea?
 - High load for the monitor
 - High load for the node being monitored
- **How do we fix this?**
 - **Easy**: monitor less frequently (pause between ack recv and heartbeat send)
 - **Harder**: use other app-specific/implicit signals (e.g., don't hbeat if receiving a file from monitored node)
 - **Beastmode**: programmable networks, reliable watchdogs

PQ 6

- Which of the following appear(s) to clients as stateless?
 - TCP server
 - UDP server
 - NFS sever
 - fdlib A1 server that's responding to heartbeats

PQ 6

- Which of the following appear(s) to clients as stateless?
 - TCP server
 - UDP server
 - NFS sever
 - fdlib A1 server that's responding to heartbeats

Simple check: if the server fails and instantly recovers, would a client notice?

PQ 7

- 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 7

- 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 8

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


PQ 8

- 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 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
 - BitTorrent has $O(\text{)$ 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%

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)**
 - 75%

PQ 11

Session semantics means that the first person to close the file “wins” (their copy will persist, while copies generated by later close calls will not)

- True
- False

PQ 11

Session semantics means that the first person to close the file “wins” (their copy will persist, while copies generated by later close calls will not)

- True
- **False : session semantics = last close wins**

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
- 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

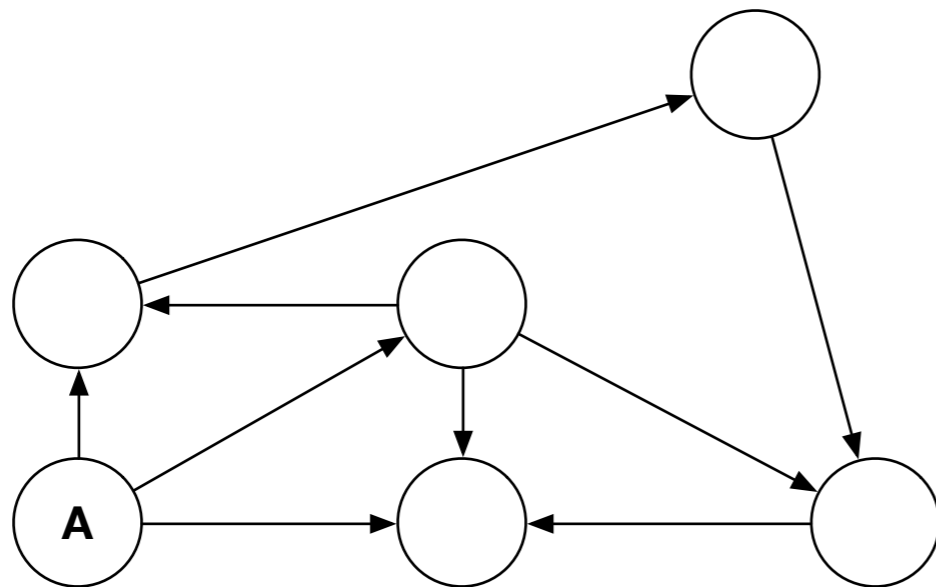
PQ 14

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

A block mined at node **A** in a project 1 deployment is flooded to a network of nodes that looks like this:



A. 1

B. 2

C. 3

D. 4

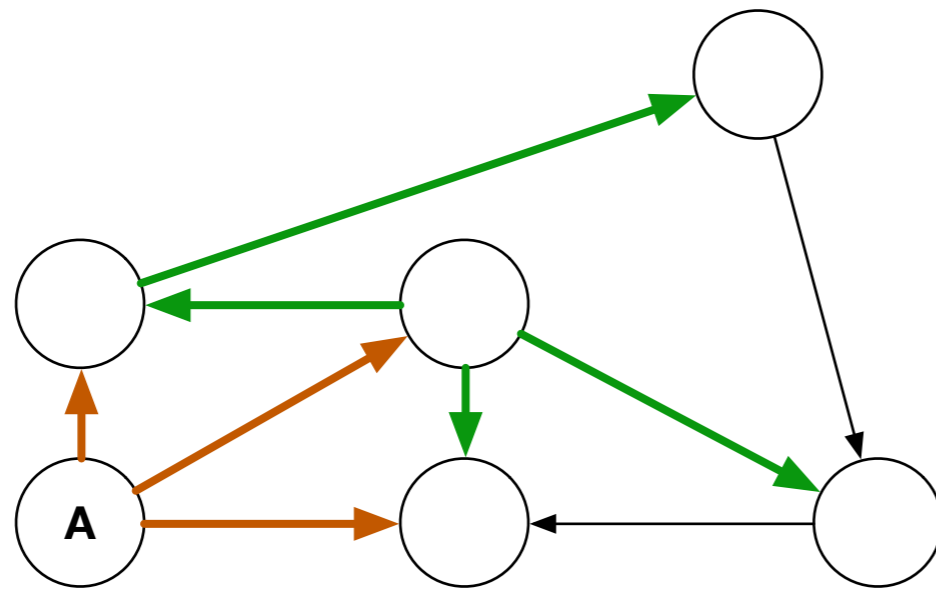
E. 5

F. 6

What is the minimum number of **rounds** before all nodes in the network receive the block?

PQ 15

A block mined at node **A** in a project 1 deployment is flooded to a network of nodes that looks like this:



A. 1

D. 4

B. 2

E. 5

C. 3

F. 6

What is the minimum number of **rounds** before all nodes in the network receive the block?

PQ 16

- The BitCoin protocol is a public ledger that maintains a set of accounts. Each block in the blockchain records the updated balance of bitcoins in each account that was part of a transaction.
 - True
 - False

PQ 16

- The BitCoin protocol is a public ledger that maintains a set of accounts. Each block in the blockchain records the updated balance of bitcoins in each account that was part of a transaction.
- True
- **False : each block records a set of transactions, the blockchain does not explicitly store account balances**

PQ 17

- 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 17

- 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 18

- Which of the following 2PC variants has the best (lowest) round efficiency?
 - Centralized 2PC
 - Linear 2PC
 - Decentralized 2PC
 - Tree-based 2PC

PQ 18

- Which of the following 2PC variants has the best (lowest) round efficiency?
 - Centralized 2PC : 3 rounds
 - Linear 2PC : $O(n)$ rounds
 - **Decentralized 2PC : 2 rounds**
 - Tree-based 2PC : $O(\log(n))$ rounds

PQ 19

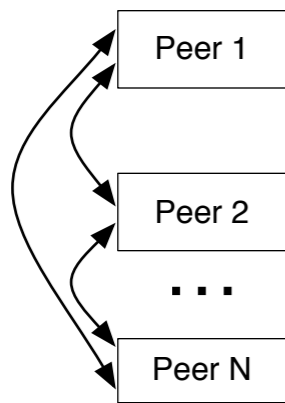
- Three-phase commit is a blocking protocol (blocks indefinitely during failures)
 - True
 - False

PQ 19

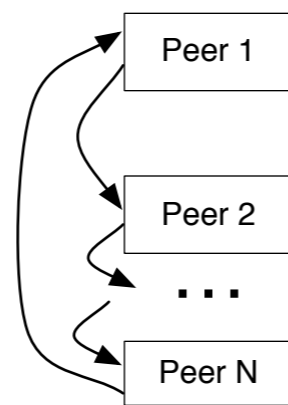
- Three-phase commit is a blocking protocol (blocks indefinitely during failures)
- True
- **False : 3PC trades off safety for liveness, it does not block (indefinitely) during failures. Waits for timeout and continues.**

PQ 20

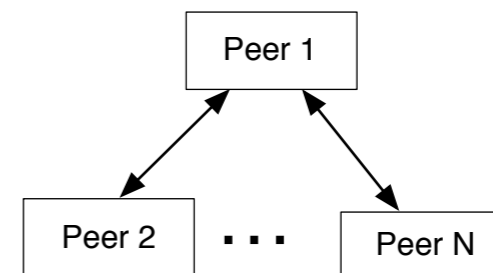
- Consider the following three topologies (e.g., in P1):



(a) all-to-all



(b) linked-list

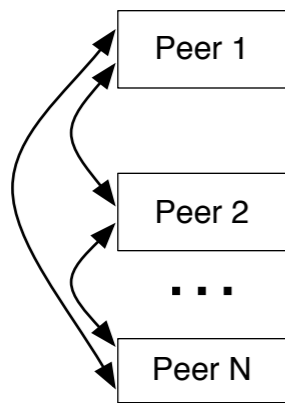


(c) star

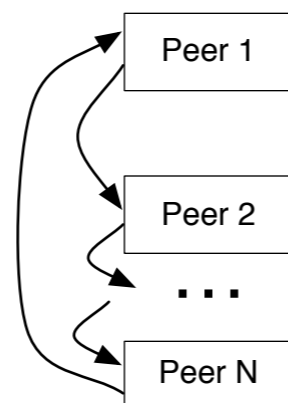
- Which topology makes it easiest for peers to detect peer failures?
- Assuming a large N , which topology (on average) impacts the fewest peers when a peer fails?

PQ 20

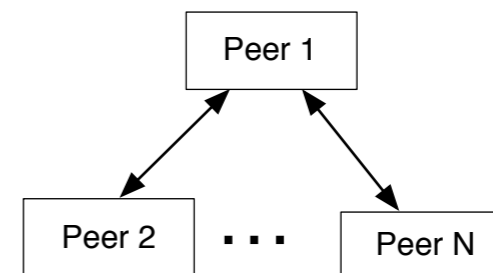
- Consider the following three topologies (e.g., in P1):



(a) all-to-all



(b) linked-list



(c) star

- Which topology makes it easiest for peers to detect peer failures? **A**
- Assuming a large N , which topology (on average) impacts the fewest peers when a peer fails? **C**

PQ 21

- 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

- 2

- 1

- 3

PQ 21

- 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 22

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

PQ 22

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

PQ 23

- You are figuring out the minimum number of paxos nodes you need to run in the initial configuration if you plan on later (1) removing n nodes from the system, and after that (2) to add n new nodes. And during all of this time you want the system to be available. What's the right answer?
 - $n-1$
 - n
 - $n+1$
 - $2n-1$
 - $2n$
 - $2n+1$

PQ 23

- You are figuring out the minimum number of paxos nodes you need to run in the initial configuration if you plan on later (1) removing **n** nodes from the system, and after that (2) to add **n** new nodes. And during all of this time you want the system to be available. What's the right answer?
 - $n-1$
 - n
 - $n+1$
 - $2n-1$
 - $2n$
 - **$2n+1$**
 - In this case majority is relative to initial set of nodes.
 - To be available between (1) and (2) must have at least $n+1$ nodes.
 - So, initial set must have at least $2n+1$ nodes

PQ 24

- When should you use SDN (a software-defined network)?
 - When you need to enforce a policy inside the network that requires global knowledge
 - When you need to route packets more efficiently
 - When you need higher network bandwidth
 - When you want more control over the network, but you are not the network operator

PQ 24

- When should you use SDN (a software-defined network)?
 - **When you need to enforce a policy inside the network that requires global knowledge**
 - When you need to route packets more efficiently
 - When you need higher network bandwidth
 - When you want more control over the network, but you are not the network operator

PQ 25

- In dataflow programs, such as MapReduce and Spark, the nodes in the dataflow graph represent individual nodes in the distributed system.
 - True
 - False

PQ 25

- In dataflow programs, such as MapReduce and Spark, the nodes in the dataflow graph represent individual nodes in the distributed system.
 - True
 - **False** (the dataflow graph is a logical graph)

PQ 26

- Two phase commit is more efficient (in the number of messages sent) than Paxos when there are no node failures and you are considering a best case execution.
 - True
 - False

PQ 26

- Two phase commit is more efficient (in the number of messages sent) than Paxos when there are no node failures and you are considering a best case execution.
 - True
 - **False** (both have two phases, identical number of messages; and, Paxos can continue with a quorum while 2PC cannot)