Distributed Systems

Exam 1 Review

Paul Krzyzanowski

Rutgers University

Fall 2018

1

Question 1

A defining characteristic of a *distributed system* is that:

a) The computers have shared memory so they can share state.
b) Computers are connected with a high-speed network.
c) The systems do not have a shared clock.
d) All of the above.

(a) Computers in a distributed system DO NOT have shared memory

(b) They are connected with a network but it does not have to be a high-speed one

6

Question 2

Caching has the following advantage over replication:

a) A cache will contain data that is more up-to-date than a replica.
b) A cache is designed to contain authoritative data while a replica may contain stale data.
c) A cache enables a system to be more fault tolerant than using a replica.
d) A cache can be smaller in size than a replica.

(a) No. All replicas should have the definitive versions of data. In a bad implementation (no write-through cache), a cache may contain up-to-data data if it was generated locally.

(b) No. That would be an invalid design. This assumes an incoherent cache.

(c) Possible in some cases – where a cache hit doesn't involve contacting the replica but caches are not designed for fault tolerance.
This is a useful side-effect of a cache but NOT an advantage over replication

(d) **Yes**. The purpose of a cache it to keep frequently-reference data close to where it is used. We can place full replicas close to where they are needed but may end up copying a lot of data we don't need.

7

Question 3

A *cache* is primarily used to improve:

a) Latency.
b) Concurrency.
c) Fault tolerance.
d) Cost.

Caches are designed to reduce latency by bringing the data close to where it is used.

The answer does not imply only distributed systems, but in-process caches, the OS buffer cache, CPU L1/L2/L3 caches, disk caches, …

8

Question 4

With a *network partition* fault:

a) Messages may take longer to arrive than expected.
b) Two systems might each think the other one is dead.
c) Some messages between two systems might get lost or corrupted.
d) Messages might be sent to the wrong system.

Network partition = a link between components is broken, resulting in segmented sub-networks that cannot communicate

(c) A partition is when systems cannot communicate: *all* messages are lost

9

Question 5

A problem with *Byzantine faults* is that a system may:

a) Restart with old versions of data.
b) Suddenly stop responding.
c) Continue running with no network connectivity.
d) Generate faulty data.

(a-c) Generic problems that have nothing to do with Byzantine faults

(d) Byzantine fault = a component appears to function but generates invalid data

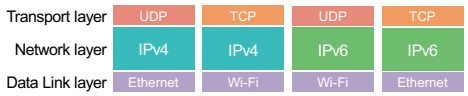
10

Question 6

Which statement is accurate?

a) TCP Is a transport layer protocol while UDP is a network layer protocol.
b) TCP Is a network layer protocol while UDP is a transport layer protocol.
c) Both TCP and UDP are network layer protocols.
d) Both TCP and UDP are transport layer protocols.

Both TCP & UDP are transport protocols on top of IP



11

Question 7

UDP has the following advantage over TCP:

a) In-order message delivery.
b) Reduced latency.
c) Ability to send a message over multiple physical networks.
d) Reliable message delivery.

(a) That's TCP

(b) Yes
- No handshake to set up a session
- Immediate message delivery instead of byte-stream delivery

(c) That's a feature of IP

(d) That's TCP

12

Question 8

Pipelining of messages refers to:

a) Sending the same message to multiple hosts.
b) Sending messages through a coordinator.
c) Sending multiple messages without waiting for responses.
d) Relaying messages through multiple routers.

Pipelining = send a stream of messages instead of (request-response), (request-response), …

(a) Broadcasting, Multicasting

(d) Routing

13

Question 9

A user program must do this to a socket in TCP but not in UDP:

a) Accept connections.
b) Assign a local address and port.
c) Acknowledge received data.
d) All of the above.

(a) UDP has no concept of a connection. This does not make sense in UDP

(b) A socket needs to be associated with an address & port in both TCP & UDP.

(c) A program doesn't *need* to do this with either. TCP provides reliable delivery. You *might* do this in UDP if you need to implement reliable delivery (but you'd probably use TCP then).

14

Question 10

Remote procedure calls are a service implemented at:

a) The operating system.
b) The programming language.
c) The network protocol.
d) All of the above.

RPC: programming language construct

Sockets: operating system construct

- RPC is a feature of the programming language – it redefines how a procedure call is implemented
- The OS does not implement procedure calls.
  RPC frameworks use OS services to access the network (sockets).
- The network protocol does not implement RPC either
  The OS uses it to send & receive the messages.

15

Question 11

To provide a *transparent interface* on the client side, remote procedure calls (RPC):

a) Provide a client stub function per remote function.
b) Require the program to first marshal all parameters.
c) Allow a programmer to specify the type of transport that is used.
d) Must handle failures.

(a) Stubs provide transparency:
A client-side stub is a local procedure that mirrors the remote one

(b) The program doesn't have to do this; the stub does

(c) This has nothing to do with providing transparency

(d) This is where transparency breaks

16

Question 12

An *Interface Definition Language* (IDL) is used to:

a) Allow programmers to define server functions in a portable manner so they can run on any system.
b) Serialize parameters into a network message.
c) List remote functions and their parameters so stubs could be generated.
d) Communicate with the network interface to send and receive messages.

(a) An IDL isn't a programming language. You don't use it to define server functions.

(b) The generated stubs do the marshaling.

(c) Yes – the entire purpose of the IDL is the generation of stubs.

(d) The stubs do that via the OS.

17

Question 13

The advantage of a *multi-canonical* marshaling standard is that:

a) Pointers and object references can be supported.
b) At least one system can use its native format without having to convert the data.
c) It is architecture independent and the same data can be sent to multiple servers.
d) Server functions can be executed on different hardware, language, and OS platforms.

- (a, c, d) Nothing to do with multi-canonical marshaling.
- The goal of a multi-canonical approach to marshaling is to minimize the amount of data conversion needed.

18

Question 14

A purpose of the Windows 10 *COM Surrogate process* is to:

a) Enable a client to locate the server that is hosting a remote object.
b) Load and run objects on a server in response to client requests.
c) Enable a client to locate a remote service on a server.
d) Provide a client-side interface to remote services.

(a) No – it's not a name server to locate servers

(b) Yes

(c) No – it's not a name server to locate services

(d) No – stub generation does that

19

Question 15

Reference counting is a common technique for garbage collection and has been used in languages such as Perl, Rust, and Python. What is a specific problem with using reference counting for remote objects?

a) Multiple references to the same object.
b) Attempts to access an object after it has been deleted.
c) Mismatch in data representation formats between the client and server.
d) Client crashes.

(a) Reference counting can handle this. Each assignment increments the reference count.

(b) This would be a bug with any form of garbage collection.

(c) This is not a problem because of reference counting or any storage management.

(d) Yes – abnormal client termination doesn't allow the client to decrement its reference counts on the server.

20

Question 16

The Web Services Description Language (WSDL) is used to:

a) Define the interface to a specific web service.
b) Locate a particular web service on the Internet.
c) Enumerate all the web services available on a server.
d) Define the implementation of a service so it can be compiled to a target platform.

- WSDL describes the messaging format for interfacing with a web service.

(b) Other services (e.g., UDDI), database, or manual transmission needs to be used

(c) No – WSDL identifies one service

(d) It only describes the interface, not the implementation

21

Question 17

Google Protocol Buffers are best described as:

a) A web-friendly, text-based data representation format.
b) A data buffering layer to avoid sending or receiving many small messages.
c) A transport layer protocol that abstracts out the underlying network interfaces.
d) An efficient binary data serialization format.

22

Question 18

Which clock synchronization algorithm starts off by having the group select the best clock source?

a) Cristian's
b) Berkeley
c) PTP
d) NTP

The Precision Time Protocol starts off by having the peers select the *Best Master Clock*

(a) Cristian's algorithm is the basis for SNTP but doesn't define clock selection

(b) Berkeley doesn't assume any system has the best source – it seeks an average

(d) NTP encourages each system (not group) to synchronize from multiple servers continually and pick the best result it gets each time

23

Question 19

Which clock synchronization algorithm encourages a computer to synchronize with multiple clocks and select the best?

a) Cristian's
b) Berkeley
c) PTP
d) NTP

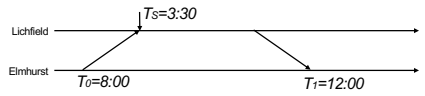
NTP encourages frequent synchronization with multiple servers to pick the clock source with the lowest total dispersion.

24

Question 20

David Brown has a clock that is not showing the correct time. He leaves his home in Elmhurst at **8:00** (on his clock) to go see what time it is on the Lichfield Clock Tower. He arrives at the tower in exactly one hour and notes the time is exactly **3:30**. On his way home, he stops at a pub for two hours. He arrives home at **12:00** (according to his clock). Using Cristian's algorithm, to what value should he set his clock?

a) 4:30
b) 5:30
c) 6:30
d) 7:30



A bunch of extraneous data here

For Cristian's algorithm, all we care about is the time the request was sent, the time the response arrived, and the server's timestamp

$$T_{new} = T_S + (T_1 - T_0)/2$$

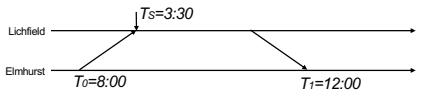
$$T_{new} = 3:30 + (12:00 - 8:00)/2 = 3:30 + (4:00)/2 = 3:30 + 2:00 = 5:30$$

25

Question 20

David Brown has a clock that is not showing the correct time. He leaves his home in Elmhurst at **8:00** (on his clock) to go see what time it is on the Lichfield Clock Tower. He arrives at the tower in exactly one hour and notes the time is exactly **3:30**. On his way home, he stops at a pub for two hours. He arrives home at **12:00** (according to his clock). Using Cristian's algorithm, to what value should he set his clock?

a) 4:30
b) 5:30
c) 6:30
d) 7:30



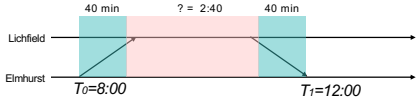
6:30 would be the most accurate value but that's not how Cristian's algorithm works

26

Question 21

Suppose David Brown knows that, with a steady and brisk walk, he can get to Lichfield in 40 minutes. What is the error of his clock calibration?

a) ±1:00
b) ±1:20
c) ±1:40
d) ±2:40



Round-trip Time (RTT) = 12:00 – 8:00 = 4:00
Best-case RTT = 0:40 + 0:40 = 0:80 = 1:20

Error = ± ½ (RTT – Best-case-RTT) = ± ½ (4:00 – 1:20) = ± ½ (2:40) = ±1:20

Or Error = ± ½ ($T_1 - T_0 - 2T_{max}$) = ± ½ (12:00 – 8:00 – 2*0:40)
= ± ½ (4:00 – 1:20) = ± ½ (2:40) = ± 1:20

27

Question 22

Free answer: 18 of you got this question wrong!

What can you *definitively* state about two events, *a* and *b*, if their Lamport timestamps are *A* and *B* respectively?

a) If *A* = *B* then a and b are concurrent.
b) If *A* > *B* then a and b are causally related and $b \rightarrow a$.
c) If *A* < *B* then a and b are causally related and $a \rightarrow b$.
d) You cannot make any conclusions about the concurrency of *a* and *b*.

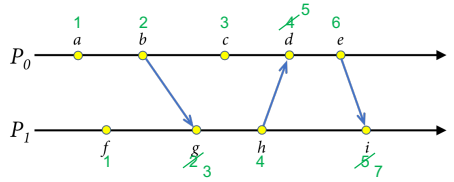
- If $b \rightarrow a$ then *A* > B but the converse is not necessarily true
  - If a and b are concurrent, it may be the case that *A* > *B*
- If $a \rightarrow b$ then *A* < *B* but the converse is not necessarily true either
  - If a and b are concurrent, it may be the case that *A* < *B*
- If a and b are causal then $a \rightarrow b$ OR $b \rightarrow a$
  - If $a \rightarrow b$ then *A* < *B*
  - If $b \rightarrow a$ then *B* < *A*
  - Therefore, if a and b are causal then $A \neq B$
  - **Therefore, if *A* = *B* then *a* and *b* must be concurrent**

28

Question 23

Assume clocks for P_0 and P_1 in the diagram are initialized to 0 (i.e., event *a* gets a Lamport timestamp of *1*).
What is the Lamport timestamp of event *i*?

a) 4
b) 6
c) 7
d) 8

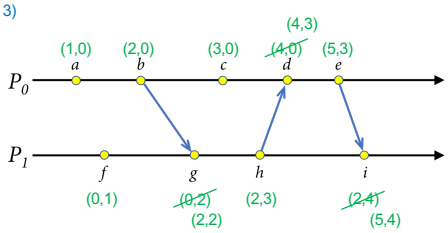


29

Question 24

Clocks are again initialized to 0 but get vector timestamps (event *a*=(1,0)). What is the vector timestamp of event *i*?

a) (5, 4)
b) (6, 1)
c) (6, 4)
d) (7, 3)



30

Question 25

What set of events is *concurrent* with event *c*?

a) { *f, g, h* }
b) { *g, h* }
c) { *f, g* }
d) { *h, i* }

- Let's compare timestamps with event c =(3, 0)
  - c = (3, 0) > a = (1, 0)
  - c = (3, 0) > b = (2, 0)
  - c = (3, 0) < d = (4, 3)
  - c = (3, 0) < e = (5, 3)
  - [c = (3, 0) ≥ f = (0, 1)] ∧ [c = (3, 0) ≤ f = (0, 1)]
  - [c = (3, 0) ≥ g = (2, 2)] ∧ [c = (3, 0) ≤ f = (2, 2)]
  - [c = (3, 0) ≥ h = (2, 3)] ∧ [c = (3, 0) ≤ f = (2, 3)]
  - c = (3, 0) < i = (5, 4)

31

Question 26

Unlike an atomic multicast, a *reliable multicast*:

a) Needs to account for the sender dying during the multicast.
b) Does not need to get acknowledgements from recipients.
c) Can give up on sending to non-responding members.
d) Must make sure messages are delivered in order.

(a) No: an atomic multicast needs to ensure the *all-or-none* property even if the sender dies partway through the multicast

(b) No: a reliable multicast needs to get confirmation of delivery

(c) Yes: unlike an atomic multicast, a reliable multicast may fail.

(d) No: ordering is a separate requirement from delivery guarantees

32

Question 27

Feedback implosion occurs when:

a) A faulty network link causes the sender to keep retransmitting the same message.
b) A routing loop causes the same message to be replicated repeatedly.
c) A group of receivers responds to the sender.
d) A message is received by systems that have not requested it.

Feedback implosion is an *amplification* problem: you send one message out but get lots of responses

33

Question 28

Implementing *total ordering* in group communication generally requires:

a) Synchronized clocks at all group members.
b) Synchronized clocks among all the senders to the multicast group.
c) A sequence number generator for each process.
d) A global sequence number generator.

Total ordering means that all processes see all received messages in the same order

(a, b) Synchronized clocks are useless because we can still have concurrent message delivery with bad ordering

(c) A per-process sequence # allows us to have unique sequence numbers per message but delivery is difficult to implement because a receiver will not know if there are missing messages from other processes

(d) A global sequence # generator makes it clear to a receiver if there are any missing messages and the current one needs to be placed on a hold-back queue

34

Question 29

IGMP, the *Internet Group Membership Protocol*:

a) Allows a multicast sender to send a message to all recipients that are members of that multicast group.
b) Allows a computer to tell its connected router that it wants to receive messages for a certain multicast group.
c) Keeps track of membership for each multicast group.
d) All of the above.

- IGMP is only used at multicast receivers to communicate with their connected router(s).
- Multicast routing within the Internet is handled via PIM (Protocol Independent Multicast)

(a) IGMP does not concern itself with the sender
(c) Nobody keeps track of the membership of the entire multicast group

35

Question 30

In Sparse Mode multicast (PIM-SM),

a) Only network segments that have explicitly requested multicast data will be forwarded the multicast traffic.
b) Multicast traffic is initially flooded to all segments of the network, which can then send prune messages to stop it.
c) Multicast traffic is sent to only a subset of the group members.
d) Only one host can act as a multicast sender.

(a) Sparse Mode PIM has routers send messages to the Rendezvous Point (RP) asking each router along the path to forward a multicast stream

(b) Dense Mode PIM floods the network
Sends messages along every route
Routers later send *prune* messages to cut off the multicasts on segments

36

Question 31

In Isis *virtual synchrony*, if the sender dies partway through sending a message:

a) It resumes sending from where it left off after restarting and reading a persistent log.
b) It re-sends the message to the group after restarting, causing some members to receive duplicates.
c) Other group members take over and send copies of that message to their peers.
d) No action needs to be taken; the sender is simply removed from the group.

- A message is not stable unless a group member has been told that all group members received it.
- A dead sender will cause a view change to take place
  - The sender will no longer be in the new group … but the message still must be delivered
- During a view change:
  - Each group member multicasts all of its unstable messages to all live group members

37

Question 32

The purpose of the *Group Membership Service* (GMS) is to:

a) Serve as a root coordinator for group messaging to handle the multicasting.
b) Provide central services, such as time, to all group members.
c) Allow clients to find out which groups provide which services.
d) Ensure all processes have a consistent knowledge of who is in the group.

The GMS is told when a group member is not responding
- It removes the member from the group
- If a member is added, the GMS is informed of the new members
- Tells everyone of the new group members
- Triggers view changes

38

Question 33

In Isis virtual synchrony, a message is considered *unstable* if:

a) It is not certain whether all group members received it.
b) Its receipt has not been acknowledged by the Group Membership Service (GMS).
c) It has not been logged to persistent storage.
d) If it has not yet been sent and its contents can still change.

39

The end

40