Distributed Systems

Exam 1 Review

Paul Krzyzanowski

Rutgers University

Spring 2020

Question 1

Metcalfe's law is based on:

a) The steadily increasing rates charged for network connectivity.
b) The number of possible point-to-point links between users in a network.
c) The exponential decrease in the cost of networking.
d) The increased efficiency of communicating with large groups of users.

Metcalfe's Law

The value of a telecommunications network is proportional to the square of the number of connected users of the system.

Question 2

Byzantine failures may be more difficult to deal with than fail-silent ones because:

a) Failures are intermittent and difficult to diagnose.
b) Multiple nodes can fail at once.
c) The receiver must know if a received message is legitimate.
d) You cannot tell if the failure is in the system or in the network.

Byzantine failure = component produces faulty data

(a) The failure does not have to be intermittent

(b) Multiple nodes failing is an unrelated problem

(d) That has no bearing on dealing with the problem

Challenge is to detect that there even is a failure:

- You're receiving requests and they appear legitimate
- They might be old (replays), altered, or malicious
- How can you tell?

Question 3

Suppose your parallel system has two components, each with a 20% chance of failure. What is the probability of system downtime?

a) 4%
b) 20%
c) 36%
d) 40%

Series system: system fails if ANY component fail

Parallel system: system fails if ALL components fail

The downtime of a parallel system < downtime of any component

(a) is the only answer that is < 20%

Or do the math:

$P(\textit{system failure}) = P(\textit{component}_1 \textit{ fails}) \times P(\textit{component}_2 \textit{ fails})$

$P(\textit{system failure}) = 0.2 \times 0.2 = 0.04 = 4\%$

Question 4

Fail-restart components must deal with:

a) Byzantine failures.
b) Stale state.
c) Network partitions.
d) Omission faults.

When a computer (or process) restarts, it does not know what happened to the rest of the system during the time it was down – it missed getting updates ⇒ stale state

Question 5

Which layer of the OSI reference model is responsible for delivering data to applications?

a) Network.
b) Data Link.
c) Transport.
d) Presentation.

Network layer (3)

- handles routing of data packets: machine-to-machine communication

Data link layer (2)

- handles communication within the same network (LAN)

Presentation layer (6)

- handles data representation

Transport layer (4)

- provides an interface for application-to-application communication

Question 6

Sockets were designed to be compatible with files in that they:

a) Allow the programmer to use a hierarchical namespace to identify resources.
b) Enable an administrator to define access permissions to specific resources.
c) Support the use of the same read & write system calls as files.
d) Access a network device that is part of the file system namespace.

(a) The namespace is up to the communication provider. Sockets do not expose a hierarchical filesystem namespace.

(b) Sockets do not enable an administrator to control access on a resource basis

(c) Once a socket is set up, *read* and *write* system calls may be used

(d) In Linux, network devices are not part of the file system namespace

Remote Procedure Calls

Question 7

The purpose of a *client stub* (proxy) function is to:

a) Generate the template code for a remote procedure that the user can fill in.

b) Create a local function that has the same interface as the remote function but marshals and sends parameters.

c) Contact a remote server to download the necessary code.

d) Locate the remote server and initialize a network connection to it.

(a) Client stubs are automatically generated

(c) Remote code runs on the remote system & isn't downloaded

(d) Locating & connecting to the remote server is done before any RPCs are invoked

(b) The client stub looks like the remote procedure but instead:

- Marshals the function request & parameters into a network message
- Sends them to the server
- Waits for a responds
- Unmarshals the response and returns it to the caller

Question 8

Idempotent functions have the advantage that:

a) Marshaling parameters is simplified.
b) There is no need to marshal parameters.
c) They do not generate any return data.
d) They can easily support at-least-once RPC semantics.

Idempotent function can be called multiple times with no side effects

Examples:

```
set_email(user, "user@example.com")
look_up("Leslie Lamport")
sin(1.112)
```

Non-idempotent functions have side effects

```
transfer(account_a, account_b, 20000.00)
add_user("Leslie Lamport", "Microsoft");
```

Question 9

Protocol buffers were designed to:

a) Serialize structured data into a binary format.
b) Allow the programmer to specify network protocols in a human-friendly manner.
c) Implement a buffering scheme to allow the rapid transmission of messages.
d) Convert between incompatible network protocols.

Protocol buffers are a portable, efficient serialization mechanism

(b) They do not specify network protocols

(c) They don't do buffering

(d) They don't deal with network protocols

Question 10

An *interface definition language* (IDL) allows a programmer to:

a) Implement remote procedures in a platform-neutral manner.
b) Define the network protocol for an application.
c) Serialize a data structure into a byte array.
d) Enumerate remote functions, their parameters, and return values.

An IDL allows a programmer to define remote interfaces

RPC compilers can then read the IDL to create client & server stubs

Question 11

A *cell directory server* in DCE RPC allows a program to:

a) Look up and download a remote interface definition.
b) Send a procedure call to a group of servers.
c) Marshal parameters prior to making a remote procedure call.
d) Discover which server is hosting a specific set of remote procedures.

Cell = grouping of computers – centrally managed

Services register themselves with the cell directory server

Client programs can query the cell directory server to locate the server on which the service is running.

Question 12

An advantage to a *multi-canonical* data marshaling format is:

a) Systems may have to do less data conversion and can communicate more efficiently.
b) More data types can be supported.
c) Object references can be sent as parameters.
d) A text-based format that makes it easy for humans to inspect the data stream.

- Multi-canonical = multiple encoding formats are supported
- Systems negotiate on which one works best for them
- Avoids the need for both sides to convert data to some “neutral” format
- Ideally – both systems can do minimal data conversions
- Example
  - Both systems may agree to use little endian encoding since they support it natively (ARM64, intel x86-64) rather than converting a big endian “network standard” encoding

Clock synchronization

Question 13

What is the worst-case error that could be obtained with Cristian's algorithm?

a) Network round-trip time.
b) Network round-trip time + clock offset.
c) ½ network round-trip time.
d) Twice the network round-trip time.

Cristian's algorithm sets the time to

$$T_{server} + ½(\text{network round-trip time})$$

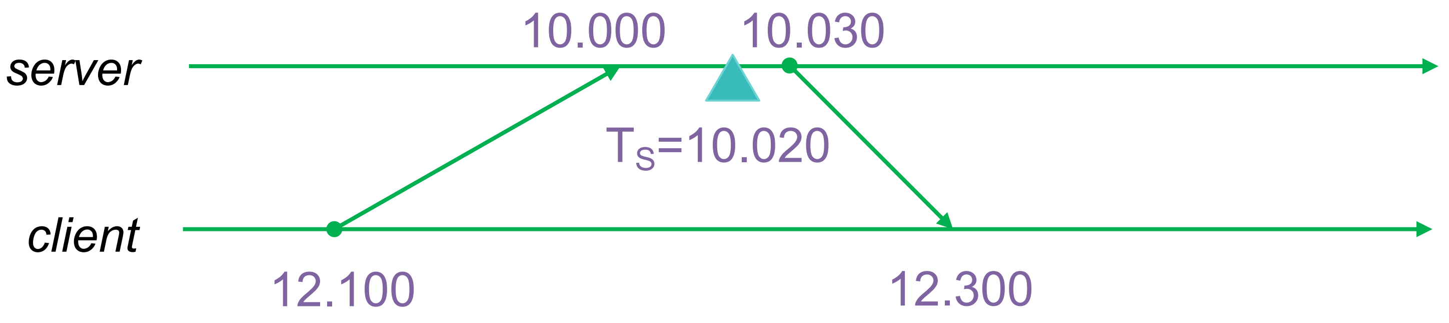
Worst-case error is when we don't know the best-case network time
= ± ½(network round-trip time)

Question 14

A system *X* wants to set its time from system *Y* using Cristian's algorithm. To what value does *X* set its clock?

a) 10.110
b) 10.115
c) 10.120
d) 10.220

X sends a request to *Y* at time 12.100
X receives a response from *Y* at time 12.300
Y receives *X*'s request at 10.000
Y sends back a message at 10.030 containing a timestamp of 10.020



Cristian's algorithm does not care about message arrival or dispatch times at the server

$$T_{new} = T_S + ½(\text{round-trip time})$$

$$= 10.02 + ½(12.3-12.1) = 10.02 + ½(0.2) = 10.02 + 0.1 = \mathbf{10.12}$$

Question 15

Which clock synchronization algorithm does not expect the presence of an authoritative time source?

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

The Berkeley algorithm synchronizes a group of clocks to their average time

Question 16

For two events *A* and *B* to be concurrent means that:

a) A does not causally precede B and B does not causally precede A.
b) The systems on which these events occur do not directly exchange messages with each other between these events.
c) The Lamport timestamps of A and B, L(A) and L(B) are identical.
d) All the above.

(a) This is simply the definition of concurrency

(b) Systems may exchange messages *indirectly* and be causally related

(c) Messages may be concurrent and have different Lamport timestamps

Question 17

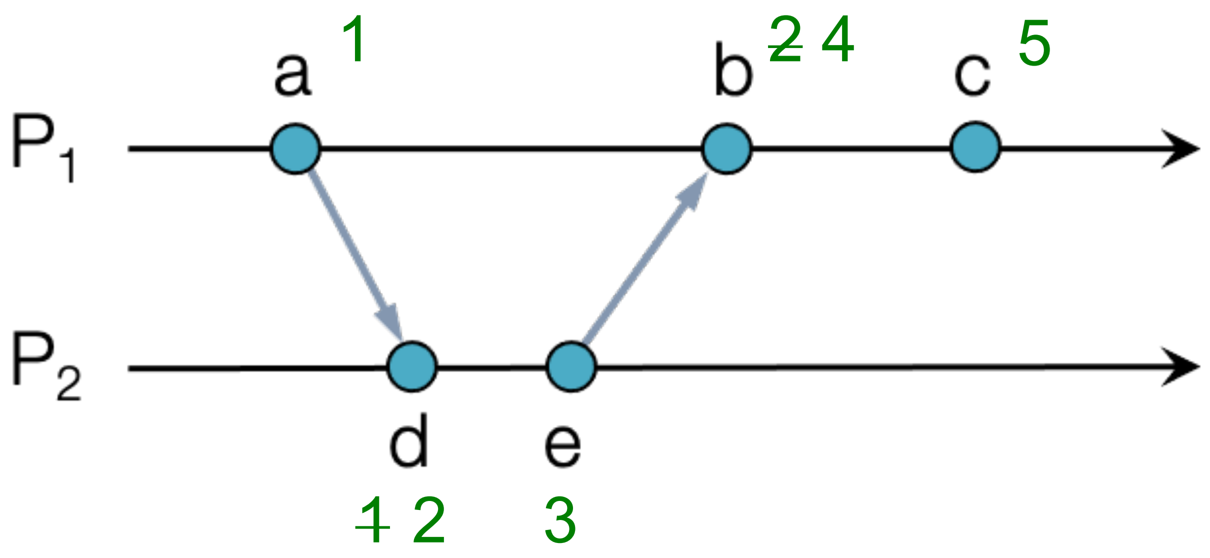
The diagram shows two processes creating events and communicating. Lamport clocks on both systems are initialized to 0 prior to any events (e.g., event a gets a timestamp of 1). What is the Lamport timestamp of event *c* ?

a) 3

b) 4

c) 5

d) 6



Question 18

Events are generated by processes that are uniquely named a, b, c, ... Which event causally precedes the event with a vector clock of (a:5, c:15, d:8)?

a) (a:5, b:1, c:12, d:7)
b) (c:8, d:8)
c) (a:4, c:14, d:9)
d) (a:6, c:16, d:15, e:0)

For *A* to precede *B*, each element of *A* ≤ *B* and *A* ≠ *B*

If we have <process_id, #> tuples, instead of positional indices then we compare matching processes

– Missing process IDs means we haven't seen messages from that process – its # = 0

(a) (a:5, b:1, c:12, d:7)	b:1 > b:0, c:12 < c:15: they are concurrent
(b) (c:8, d:8)	a:0 < a:5, c:8 ≤ c:15, d:8 ≤ d:8
(c) (a:4, c:14, d:9)	a:4 < a:5, c:14 < c:15 but d:9 > d:8 concurrent
(d) (a:6, c:16, d:15, e:0)	a:6 > a:5, c:16 > c:15, d:15 > d:8, e:0 = e:0 – this event follows *(a:5, c:15, d:8)*

Group communication

Question 19

This technique has the danger of creating *feedback implosion* in a multicast:

a) Pipelining.
b) Negative acknowledgements.
c) Flooding.
d) Message acknowledgement.

Feedback implosion: send one message out; receive lots of responses

(d) Acknowledgements from each receiver of a multicast can create this situation

(a) Pipelining

- send successive requests without waiting for acknowledgements

(b) Negative acknowledgements

- Sent when missing messages are detected – ***minor implosion possible here***

(c) Flooding

- Every node (router) relays a copy of the message to every connected node

Question 20

A process P_0 has a precedence vector of **(8,2,4)** with the sequence (P_0, P_1, P_2). Which message from P_2 can be delivered immediately to preserve causal ordering?

a) (8, 2, 6)
b) (9, 3, 5)
c) (4, 1, 3)
d) (1, 1, 5)

Check

1. Do we have the next sequential message from P2?
   - Our last message had a sequence # of 4 – we need 5
2. Is every other sequence # in the received message ≤ ours?
   - Ensures that the message is not based on any other messages we have not seen
   - We need $P_0 \leq 8$, $P_1 \leq 2$

(b) and (d) satisfy condition 1

– But in (b): $P_0 = 9 > 8$ and $P_1 = 3 > 2$

– In (d): $P_0 = 1 \leq 8$ and $P_1 = 1 \leq 2$

Question 21

The *Internet Group Management Protocol* (IGMP):

a) Enables a multicast coordinator to track the members of a specific IP multicast group.
b) Links multicast senders with multicast receivers.
c) Enables a multicast sender to get permission to transmit to a multicast group.
d) Allows a computer on a LAN to inform its connected router that it wants to join a multicast group.

(a) There is no multicast coordinator and nobody tracks the set of members of a multicast group

(b) No – a protocol independent multicast (PIM) deals with getting messages from senders to receivers

(c) Senders do not need permission to send messages

(d) IGMP is a protocol for receivers to inform their router that they're interested in a multicast group

– The router will have to use PIM to talk to other routers (if necessary)

Question 22

Dense mode multicast differs from sparse mode multicast because dense mode multicast:

a) Messages reach all group members while sparse mode multicasts reach only a subset.
b) Tries to forward a multicast stream to every router on the Internet.
c) Is designed for high-bandwidth multicast streams, such as video.
d) Requires defining a rendezvous point prior to the multicast.

Dense mode multicast uses flooding to try to reach every router

Routers at edges then send *prune* messages if they have nobody interested in the stream

– Other routers, in turn, can send *prune* messages to their connected routers

Question 23

A *rendezvous point* is:

a) The system that contains a list of all multicast subscribers for a specific multicast address.
b) The path that a multicast stream takes from the sender to a specific destination.
c) A router to which all multicast streams for a specific multicast address are sent.
d) Any edge router that connects to the local area network and receives multicast streams.

- A rendezvous point is used by Sparse Mode multicast (PIM-SM) for specific addresses
  - Predefined router
  - Senders send multicast messages to the rendezvous point
  - Receivers send *join* messages toward the rendezvous point to join a group
    - Routers along the way will route the *join* requests if they have not already done so
- Messages will flow from the sender → rendezvous point → receivers

Question 24

The Isis implementation of virtual synchrony does *not*:

a) Have a group member to take over finishing a multicast when the sender died.
b) Use a persistent log to resume a multicast if a sender restarts.
c) Redefine the group if a member dies.
d) Use a hold-back queue.

(a) If a sender dies, a surviving group member will ensure everyone in the group gets the message

(c) If a member dies, a view change takes place, defining a new group without that member

- If the member come back, it will have to re-join the group

(d) Hold-back queues are used to hold messages that are not stable

- If we are not sure that everyone received the message, no member can process it

(b) There is no requirement for a persistent log. Any system that restarts will have to first sync its state from another group member.

Question 25

A message is considered *stable* in virtual synchrony when:

a) It is confirmed that every group member received the message.
b) Sending the message does not cross view change boundaries.
c) It is confirmed that every group member delivered the message to its application.
d) A sender has created the message and is ready to multicast it to the group.

- A sender reliably multicasts a message to each group member
  - Each group member sends an acknowledgement
  - At this point, the message is considered *unstable* at each member
- After the sender gets all acknowledgements, it can let group members know the message is *stable*
  - Now each group member can deliver it to the application

The end