Distributed Systems

10. Exam 1 Review

Paul Krzyzanowski
Rutgers University
Fall 2014

1

Fall 2014 - Question 1

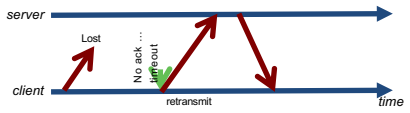
Is the statement below true or false? Explain your answer.
You should use a reliable transport protocol such as TCP for clock synchronization.

No.

Reliable, in-order transport comes at the price of incurring extra delays due to buffering and retransmission.

A lost packet in one direction will skew the latency of one of the messages dramatically.

An unreliable transport does not mean the message is usually lost!
It usually is NOT!



2

Fall 2014 - Question 2

Explain the benefit of multi-canonical marshaling.

Multiple data encoding formats are allowed.

The hope is that at least one will fit the architecture of one or both machines and avoid the overhead of data conversion.

3

Fall 2014 - Question 3

Why is using only reference-count based distributed garbage collection a problem?

It is not fault tolerant.

If a client dies or gets disconnected from the network, the server will not know to clean up the object.

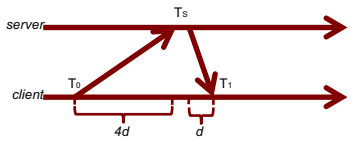
This is why lease-based collection wins.

Not: race condition when transferring an object reference – we know how to deal with that.

4

Fall 2014 - Question 4

Cristian's clock synchronization algorithm assumes symmetric network latency: the latency of sending a message is expected to be the same as that of receiving a response. This is not true in all networks. **How would you modify the formula for setting the time if the downlink is 4 times faster than the uplink**?



Assume processing time at server ≈ 0 relative to network delays

$$T_{new} = T_S + \left(\frac{d}{4d+d}\right)(T_1 - T_0) = T_S + \frac{T_1 - T_0}{5}$$

5

Fall 2014 - Question 4

Cristian's clock synchronization algorithm assumes symmetric network latency: the latency of sending a message is expected to be the same as that of receiving a response. This is not true in all networks. **How would you modify the formula for setting the time if the downlink is 4 times faster than the uplink**?

Multiplying a time *span* makes sense (e.g., *N times longer*).

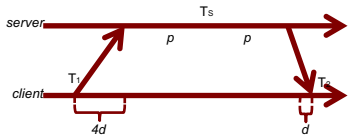
Multiplying an individual timestamp makes no sense!

$$T_{new} = T_S + \frac{1}{4}T_1 - T_0$$ **Wrong!**

6

Fall 2014 - Question 4 (continued)

Cristian's clock synchronization algorithm assumes symmetric network latency: the latency of sending a message is expected to be the same as that of receiving a response. This is not true in all networks. **How would you modify the formula for setting the time if the downlink is 4 times faster than the uplink?**



If, however, network delay is ≈ 0 relative to processing time at server

$$T_{new} = T_S + \lim_{d \to 0}\left(\frac{d + p}{4d + d + 2p}\right)(T_1 - T_0) = T_s + \frac{T_1 - T_0}{2}$$

7

Fall 2014 - Question 5

You have a set of servers that manage shopping cart data. Clients may send updated carts to any server and servers propagate these updates to other members of the group.

Unfortunately, the network sometimes gets partitioned temporarily so only a subset of servers get updates.

You need to reconcile the shopping cart for a user.

The vector timestamp is a set of values associated with processors P_0, P_1, … P_n. Items in the cart are a subset of {A, B, … Z}.

Causal updates should overwrite previous contents. Concurrent updates merge with previous contents. **What is the final set of items in the user's shopping cart** when reconciling the following timestamps?

- Clock = { P_0: 2 } Value = {A, C}
- Clock = { P_0: 1, P_1: 2 } Value = { A, B, Q }
- Clock = { P_0:1, P_1: 2, P_2: 1 } Value = {A, E, H }
- Clock = { P_0: 1, P_1: 2, P_2: 2 } Value = { A, K }

8

Fall 2014 - Question 5 (continued)

Causal updates should overwrite previous contents. Concurrent updates merge with previous contents. **What is the final set of items in the user's shopping cart** when reconciling the following timestamps?

- Clock = { P_0: 2, P_1: 0, P_2: 0 } Value = {A, C} — *concurrent*
- Clock = { P_0: 1, P_1: 2, P_2: 0 } Value = { A, B, Q } — *causal - overwrite*
- Clock = { P_0:1, P_1: 2, P_2: 1 } Value = {A, E, H } — *causal - overwrite*
- Clock = { P_0: 1, P_1: 2, P_2: 2 } Value = { A, K }

Note: if a vector clock value is missing a specific process, that means that it has not been seen and has a value of 0

We see with two concurrent events:

Clock = { P_0: 2, P_1: 0, P_2: 0 } Value = {A, C}
Clock = { P_0: 1, P_1: 2, P_2: 2 } Value = { A, K }

Shopping cart = {A, C} ∪ {A, K} = **{ A, C, K}**

9

Fall 2014 - Question 6

Compared to home snooping, source snooping:
(a) Uses less bandwidth and has a lower latency.
(b) Uses more bandwidth and has a higher latency.
(c) Uses more bandwidth but has a lower latency.
(d) Uses less bandwidth but has a higher latency.

- Source snooping requires contacting all processors
- Latency is lower: the processor with the cached data is contacted directly rather than via the home agent

10

Fall 2014 - Question 7

The most efficient use of a network is:
(a) Time division multiplexing (TDM).
(b) Frequency division multiplexing (FDM).
(c) A token passing protocol.
(d) Random access.

- TDM & FDM: bandwidth reserved even if not used
- Token passing: need to wait for a token

11

Fall 2014 - Question 8

Which of the following was *NOT* a design goal of the Internet?
(a) Provide reliable communication.
(b) Support the interconnection of different physical networks.
(c) Use routers to store and forward packets.
(d) Have decentralized control of the network.

- IP was designed on top of unreliable packet switched networks
- Any reliability would have to be provided via software at the endpoints

12

Fall 2014 - Question 9

A port number in IP protocols is used in the:
(a) Data Link layer.
(b) Network layer.
(c) Transport layer.
(d) Session layer.

- Data link: MAC address – below the IP layer
- Network layer – IP
- Transport layer – TCP and UDP – application-to-application communication

13

Fall 2014 - Question 10

The TCP layer does not deal with:
(a) Reliable data transfer.
(b) In-order data transfer.
(c) Flow control.
(d) Secure data delivery.

14

Fall 2014 - Question 11

Sockets are said to be compatible with files because:
(a) They can be named with textual names.
(b) You can use open and close system calls.
(c) You can use read and write system calls.
(d) All of the above.

15

Fall 2014 - Question 12

In RPC, the client stub:
(a) Implements the remote function on the client but uses remote access for all data in that function.
(b) Is an automatically-generated placeholder function where the programmer fills in the logic.
(c) Is a wrapper that loads the server function.
(d) Presents the functional interface of the remote function and sends a request to the server.

16

Fall 2014 - Question 13

In RPC, marshaling refers to:
(a) Serializing the data of an object for transmission.
(b) Sending a message to invoke a remote function.
(c) Looking up the address of a server.
(d) Packaging the code for a function so it can be sent and executed remotely.

17

Fall 2014 - Question 14

An interface definition language is used to:
(a) Enable the generation of stub functions.
(b) Implement the server-side remote functions.
(c) Access remote objects.
(d) Define interfaces for all local and remote classes in a program.

18

Fall 2014 - Question 15

A linear compensation function:

(a) Makes a clock run faster or slower by a constant amount.
(b) Converts a variable-frequency clock to a fixed-frequency clock.
(c) Sets the system time forward or back by a fixed amount to set the time to the desired value.
(d) Generates periodic timer interrupts to allow an operating system to keep track of time.

19

Fall 2014 - Question 16

A client requests a timestamp from the server at **3:52:20.200**. The server response is received at **3:52:20.600**. The response contains a server timestamp of **3:52:20.400**. Using Cristian's algorithm, the client should set its time to:

(a) 3:52:20.200
(b) 3:52:20.400
(c) 3:52:20.600
(d) 3:52:20.800

Overall delay = 3:52:20.600 – 3:52:20.200 = 400 ms

Cristian's algorithm assumes the timestamp was generated ½ of the delay ago = 200 ms ago

Set time to [received timestamp] + 200 ms =
= 3:52:20.400 + 0.200 = **3:52:20.600**

20

Fall 2014 - Question 17

A client synchronizes from a server. It sends a request at 3:52:20.200. The time between a request and a response is **52 ms**. The best-case time is **40 ms**. What is the error of the synchronization?

(a) ±6 ms
(b) ±12 ms
(c) ±20 ms
(d) ±106 ms

The error is the unaccounted time in our sync.

Best case time = 40 ms

Measured time = 52 ms

Unaccounted time = 52 – 40 = 12 ms.

Since the timestamp is set to the middle, the error is ± (12/2) = ± 6 ms

21

Fall 2014 - Question 18

If event A has a Lamport timestamp of **2** and event B has a Lamport timestamp of **3**, you are certain that:

(a) A happened before B if the events took place on the same process.
(b) A happened before B regardless of the processes on which A and B took place.
(c) A and B are concurrent.
(d) B happened before A, regardless of the processes on which A and B took place.

22

Fall 2014 - Question 19

Which event is concurrent with the vector timestamp { 5, 2, 1, 4 }?

(a) { 6, 3, 2, 5 }
(b) { 5, 2, 1, 5 }
(c) { 4, 2, 0, 4 }
(d) { 5, 1, 2, 4 }

23

Fall 2014 - Question 20

Which of these multicasts can be implemented with a group-wide sequence number server?

(a) Global time ordered multicasts
(b) Total ordered multicasts
(c) Partial ordered multicasts
(d) Unordered multicasts

24

Fall 2014 - Question 21

A hold-back queue:

(a) Allows a sender to queue messages if the receiver is too congested to receive them.
(b) Holds messages at the sender until they are acknowledged in case the sender needs to retransmit them.
(c) Enables the receiver to receive messages at a constant rate.
(d) Allows a receiver to switch the order of received messages before giving them to the application.

25

Fall 2014 - Question 22

A *precedence vector* in a message is used by a receiver to:

(a) Ensure that all messages are received in the same order by all group members.
(b) Create timestamps of when each message has been received.
(c) Identify whether all previous causally-related messages have been received.
(d) Create a log of received messages in the order they were received.

26

Fall 2014 - Question 23

IGMP, the Internet Group Management Protocol, is used to:

(a) Build a global list of hosts that are subscribed to a given multicast group.
(b) Tell a router that a connected host is interested in receiving a multicast group.
(c) Enable system administrators to manage multicast subscriptions.
(d) Allow a multicast sender to discover and contact a multicast receiver.

27

Fall 2014 - Question 24

Which mutual exclusion algorithm requires, on average, the fewest messages in a group of 100 machines?

(a) Centralized.
(b) Token-ring.
(c) Lamport's.
(d) Ricart & Agrawala.

28

Fall 2014 - Question 25

The Ricart & Agrawala mutual exclusion algorithm:

(a) Does not depend on time stamps in messages while Lamport's does.
(b) Cannot handle the case where two or more processes request the same resource at the same time.
(c) Does not require a process to send messages to the entire group while Lamport's does.
(d) Requires fewer messages than Lamport's algorithm.

29

Fall 2014 - Question 26

The Chang & Roberts ring election algorithm optimizes the ring election algorithm by:

(a) Contacting only higher-numbered processes.
(b) Multicasting the election request instead of passing messages in a ring.
(c) Choosing a leader as messages are sent instead of sending a growing list of candidates.
(d) Having a node always discard an election message that came from a process with a smaller ID.

With the ring algorithm, an *election* message circulates around all live processes.

When it reaches the originator, the process picks the leader (either the highest or lowest process ID)

Chang & Roberts optimizes in two ways:

(1) Keep at most one process ID in the *election* message. No need to put process IDs that have no chance of being the leader.

(2) Try to avoid concurrent elections. If a process gets an election message with a smaller ID and the process already is participating in an election, discard the message.

30

Fall 2014 - Question 27

A practical way to handle network partitioning is to:

(a) Have each process start an election.
(b) Elect a leader only if a majority of nodes can be reached.
(c) Run a normal election and multicast the result to the subset of processes that can be reached.
(d) Divide and conquer: elect multiple coordinators.

31

The end

32