Homework 3 Review
You have the following timestamps:

- Client request sent: 9:10:12.100  \( T_1 \)
- Server receives request: 9:10:14.250  \( T_2 \)
- Server sends response: 9:10:14.300  \( T_3 \)
- Client receives response: 9:10:12.300  \( T_4 \)

Time is expressed as hours:minutes:seconds.decimal_seconds

In the case of a client synchronizing with the server, \( A \) refers to the client and \( B \) refers to the server in the NTP RFC. Using NTP, what is the new time (add the offset, \( \theta \), to the client receives response time)?

\[
\theta = T(B) - T(A) = \frac{1}{2} \times \left( (T_2-T_1) + (T_3-T_4) \right)
\]

\[
= \frac{1}{2} \times \left( 9:10:14.250 - 9:10:12.100 + (9:10:14.300 - 9:10:12.300) \right)
\]

\[
= \frac{1}{2} \times [0:2.150 + 0:2.00]
\]

\[
= \frac{1}{2} \times 0:4.150 = 0:2.075
\]

\[
\text{time} = 9:10:12.300 + 0:2.075 = 9:10:14.375
\]
Two events are concurrent if neither can causally affect the other.
The table above shows ten events (a, b, ..., j) taking place among three processes. Assign Lamport timestamps to each event. The event clock on each process is initialized to zero at the beginning and incremented prior to timestamping each event. For instance, the clock on P0 starts at 0 and event a gets assigned a Lamport timestamp of 1.

Increment per-process counter prior to each event.

If time stamp in a received message > counter, set the counter to [message time stamp] + 1

If f was an isolated event on P1, it would get a Lamport timestamp = 1
Since the event is the receipt of a message sent from P0 with timestamp = 2, f has to be set to max(2+1, 1) = 3
The table above shows ten events (a, b, ..., j) taking place among three processes. Assign Lamport timestamps to each event. The event clock on each process is initialized to zero at the beginning and incremented prior to timestamping each event. For instance, the clock on P0 starts at 0 and event a gets assigned a Lamport timestamp of 1.

Increment per-process counter prior to each event.

If time stamp in a received message > counter, set the counter to [message time stamp] + 1

<table>
<thead>
<tr>
<th></th>
<th>a. 1</th>
<th>b. 2</th>
<th>c. 3</th>
<th>d. 5</th>
<th>e. 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>f. 3</td>
<td>g. 4</td>
<td>h. 1</td>
<td>i. 4</td>
<td>j. 5</td>
<td></td>
</tr>
</tbody>
</table>
Using the same set of events as in the previous question, assign vector timestamps to each event. The event clock vector at each process is initialized to all zeros at the beginning and a process increments its position in the vector prior to timestamping each event. Process positions in the vector are \( (P_0, P_1, P_2) \).

\[
(1, 0, 0) \quad (2, 0, 0) \quad (3, 0, 0)
\]

Increment per-process counter in the vector prior to each event.

For received messages with received vector \( r \), new vector =

\[
v[\text{process}_\text{id}]++; \\
\text{for } (i = 0; i < \#\text{elements}; ++i) \\
\quad v[i] = max(v[i], r[i])
\]

Vector is \( (P_0, P_1, P_2) \)
Event \( f \) would have been \( (0,1,0) \) if it was isolated. Since it’s the receipt of a message, We set the vector to \( (\max(0,2), \max(1,0), \max(0,0)) = (2, 1, 0) \)
Using the same set of events as in the previous question, assign vector timestamps to each event. The event clock vector at each process is initialized to all zeros at the beginning and a process increments its position in the vector prior to timestamping each event. Process positions in the vector are \( (P_0, P_1, P_2) \).

Increment per-process counter in the vector prior to each event.

For received messages with received vector \( r \), new vector =

\[
\text{v[process_id]}++; \\
\text{for} \ (i = 0; \ i < \ \#\text{elements}; \ ++i) \\
\text{v[ i ]} = \max(\text{v[ i ]}, \ r[ i ]) 
\]
Based on the vector timestamps, which events are causally dependent on event c (that is, which events follow c and are causally related)?

For two events to be causally dependent on each other, every element of one vector have to be $\geq$ the corresponding element of the other vector:

```c
for (i=0; i<$\#$elements; ++i)
    if ( a[i] < b[i] ) smaller = 1
    if (a[i] > b[i]) larger = 1
if ((smaller == 1) && (larger == 1))
    concurrent
else
    causal
```

We need to find events that are $>$ event c.

<table>
<thead>
<tr>
<th></th>
<th>a. (1,0,0)</th>
<th>b. (2,0)</th>
<th>c. (3,0)</th>
<th>d. (4,2)</th>
<th>e. (5,2,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f.</td>
<td>(2,1,0)</td>
<td>(2,2,0)</td>
<td>(0,0,1)</td>
<td>(3,0,2)</td>
<td>(3,0,3)</td>
</tr>
</tbody>
</table>

---
Question 5

Based on the vector timestamps, which events are causally dependent on event c (that is, which events follow c and are causally related)?

We need to find events that are > event \( c \).
\( c = (3, 0, 0) \)

a. \((1, 0, 0) < (3, 0, 0)\) – causal but \( a < c \)
b. \((2, 0, 0) < (3, 0, 0)\) – causal but \( b < c \)
d. \((4, 2, 0) > (3, 0, 0)\) – causal and \( d > c \)
e. \((5, 2, 3) > (3, 0, 0)\) – causal and \( e > c \)
f. \((2, 1, 0) \not< (3, 0, 0)\) and \((2, 1, 0) \not> (3, 0, 0)\) – concurrent

g. \((2, 2, 0) \not< (3, 0, 0)\) and \((2, 2, 0) \not> (3, 0, 0)\) – concurrent

h. \((0, 0, 1) \not< (3, 0, 0)\) and \((0, 0, 1) \not> (3, 0, 0)\) – concurrent

i. \((3, 0, 2) > (3, 0, 0)\) – causal and \( i > c \)

j. \((3, 0, 3) > (3, 0, 0)\) – causal and \( i > c \)

The set of events that causally follow \( c \) are:
\( d, e, i, j \)

<table>
<thead>
<tr>
<th></th>
<th>a. (1,0,0)</th>
<th>b. (2,0,0)</th>
<th>c. (3,0,0)</th>
<th>d. (4,2,0)</th>
<th>e. (5,2,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f.</td>
<td>(2,1,0)</td>
<td>(2,2,0)</td>
<td>(0,0,1)</td>
<td>(3,0,2)</td>
<td>(3,0,3)</td>
</tr>
</tbody>
</table>
Selected Questions From Past Exams
Why is processor affinity important in a NUMA (non-uniform memory access) multiprocessor system?

An operating system will try to allocate memory from the region that is local to the running process. If a process gets rescheduled to another processor, memory operations will be slower. Moreover, any cached contents will be lost.
With traditional marshaling, there is a chance that both sides will need to convert data between a native format and a network standard representation.

With multi-canonical marshaling, it is likely that at least one side will not have to bother to do that since its native format will be the same as one of the allowable network representations.
Assign vector timestamps to the six remaining events

Which events are concurrent with event e?

- g, h, c, i

Example:
- i is concurrent with e because (2, 0, 3) is not greater than or less than (2, 2, 0)
- f is not concurrent because (2, 3, 3) > (2, 2, 0)
  That means (2≥2), (3≥2), (3≥0).
a) From an application’s perspective, what are the differences between virtual circuit service over a packet-switched network and real virtual circuit networking?

**Constant bandwidth and constant latency (no jitter).**

*Virtual circuit service at the transport layer (such as TCP) tries to simulate the behavior of a virtual circuit network. A VC network provides guaranteed bandwidth, which software cannot recreate.*

b) What are the similarities?

**Reliable, in-order message delivery.**

*Software can retransmit lost packets and buffer & resequence received packets.*
The National Institute of Standards and Technology (NIST), an agency of the U.S. Department of Commerce, is responsible for keeping track of the official time in the U.S. Why not just sync your time from a stratum 1 server directly, such as time.nist.gov, and get the time from the most accurate source instead of syncing with a higher-stratum server?

You might experience high latency and/or high jitter when communicating with a stratum 1 server but achieve a lower total error when synchronizing with a higher-stratum server because you can communicate with it more reliably.
You are synchronizing with a server using Cristian's algorithm. You send a message at 1:10.100 (according to your clock). The server receives the message at 1:15.000 (according to its clock), processes the request, and sends a response containing 1:15.005, which you receive at 1:10.150. To what value do you set your clock?

The time at which the server receives the message (1:15.000) is irrelevant.

Note that 1:15.005 is not the time the response was sent; it is the server clock timestamp. We can just plug it into Cristian’s algorithm and set the time to the server time plus \( \frac{1}{2} \) of the round-trip time.

\[
\text{New time} = T_S + \frac{1}{2}(1:10.150 - 1:0.100) \\
= 1:15.005 + \frac{1}{2}(0.050) \\
= 1:15.005 + 0.025 = 1:15.030
\]
If the error of the clock at the server was ±0.015 sec and if the best-case round-trip time to the server is 30 msec. (0.030 sec), what is the time error of the synchronized clock at the client (express as ±E)?

The error is NOT the best case round-trip time; that would give us an error of 0 due to the network. We need to add the error of the clock and the uncertainty of the network round-trip time.

Error to server = ± 1/2(round trip delay – best case round trip delay)
= ± 1/2(0.050 – 0.030)
= ± 1/2(0.020)
= ± 0.010

Clock error = (error to server) + (error of clock at server)
= ± (0.010 + 0.015)
= ± 0.025 sec
Compare reference count based distributed garbage collection with lease-based garbage collection.

What advantages of the latter made it become the dominant technique for distributed garbage collection?

**Reference counting** requires clients to send increment and decrement operations to a server so it can maintain a reference count and free the object when the count goes to zero.

Because of possible software errors, software crashes, and client crashes, the server may end up with inaccurate counts and never free an object.

**Lease-based garbage collection** requires a client to periodically renew its leases on any objects that it uses. If the client dies, then it will fail to renew its leases and the server will free the object. It’s more fault tolerant. With reference counting, you still had to take care of the case of a client abnormally terminating (or improper counting).
Metcalfe's law isn't really a law but basically states that:

(a) A network gets more useful with more people on it.

(b) Processors get twice as powerful every two years.

(c) Transistors get twice as small every two years.

(d) Network bandwidth doubles every two years.
A multiprocessor system is best characterized as one where the processors:

(a) Connect to a data network.

(b) Communicate over a shared bus.

(c) **Shared system memory.**

(d) Share the same cache memory
Fall 2012: Question 8

An SMP (symmetric multiprocessor) system is a multiprocessor system where:

(a) The processors engage in the same computation for fault tolerance.

(b) All processors have equal access to memory.

(c) An even number (2, 4, …) of processors allows workloads to be partitioned.

(d) Two or more computers are connected to a shared network.
A snoopy cache reduces traffic on the system bus:

(a) For read operations.

(b) For write operations.

(c) For both read and write operations.

(d) None of the above; it has no effect on bus traffic.
A single system image refers to:

(a) multiple computers that look and behave like one system.

(b) a single operating system that controls a network of computers.

(c) a cloned set of computers where multiple machines run the same software.

(d) an instance of a single computer connected to a network.
A snoopy cache is designed for this architecture:

(a) Bus-based.

(b) Crossbar switch.

(c) NUMA.

(d) All of the above
TCP/IP occupies which layer of the OSI protocol stack?

(a) Data Link (2)

(b) Network (3)

(c) Transport (4)

(d) Presentation (6)

TCP and UDP both operate at the transport layer. They allow application-to-application communication and use port numbers to identify communication endpoints (sockets).
UDP/IP occupies which layer of the OSI protocol stack?

(a) Data Link (2)

(b) Network (3)

(c) Transport (4)

(d) Presentation (6)

TCP and UDP both operate at the transport layer. They allow application-to-application communication and use port numbers to identify communication endpoints (sockets).
There are three systems on your network: A, B, C. Prior to synchronization, A's clock reads 2:30, B's clock reads 2:36, and C's clock reads 2:42. B is the master. After synchronization via the Berkeley algorithm, what is the time on A's clock (ignore synchronization or network latencies)?

(a) 2:30
(b) 2:33
(c) 2:36
(d) 2:39

Berkeley: there is no master clock
compute the average of 2:30, 2:36, 2:42
= (2:30+2:36+2:42)/3 = 2:36
Based on their vector timestamps, which event causally precedes $(4, 2, 8, 5)$?

(a) $(3, 1, 7, 7)$

No: element 1: $3 < 4$ but element 4: $7 > 5$

(b) $(5, 1, 6, 2)$

No: element 1: $5 > 4$ but element 4: $2 < 5$

(c) $(4, 2, 8, 4)$

(d) $(4, 3, 8, 5)$

No: each element of (d) ≥ each element of $(4, 2, 8, 5)$. This vector causally follows $(4, 2, 8, 5)$.

We’re looking for a vector that is neither ≥ or ≤ $(4, 2, 8, 5)$ when doing an element-by-element comparison.
Fall 2012: Question 21

The implementation of which message ordering technique is not a feasible one to implement?

(a) Global time ordering

(b) Total ordering

(c) Sync ordering

(d) Unordered

Global time ordering requires globally exactly synchronized time stamps and the assumption that every single message will be generated at a different point in time. This is not feasible or realistic.
The following events took place on three systems at the stated physical (time of day) times:

System 1: a (3:00), b (3:01)

System 2: c (2:55), d (3:04)

System 3: e (3:03)

Event b is the sending of a message from system 1 to 2. Event d is the receipt of that message. Using Lamport's happened-before relation, which events happened before event d?

(a) a
(b) a, b
(c) a, b, c
(d) a, b, c, e
IP multicast provides which level of reliability?

(a) Atomic
(b) Reliable
(c) Unreliable
(d) User-selectable

(a) Ensures the message gets to all the clients, even if the sender dies partway through.
(b) Tries and retries, if necessary, but eventually gives up on delivery.
(d) Makes no sense.
5. ONC (Sun) RPC provides the ability to:
   (a) Use XML as a transport.
   (b) Start up the server process on demand.
   (c) Perform distributed garbage collection.
   (d) Have multiple versions of a function at the server.

6. A multi-canonical marshaling format
   (a) Provides greater efficiency because both sides usually won’t have to convert data.
   (b) Is a more compact way of representing data over a network.
   (c) Encodes data concurrently into both binary as well as text formats.
   (d) Allows one message to be sent to multiple servers.

7. For RPC, a DCE cell directory server allows:
   (a) A client to find out on what server an interface is available.
   (b) A client to find the port number of a service on a specific machine.
   (c) A server to send callbacks to clients.
   (d) An object to be distributed among multiple servers.
8. Java's Serializable interface:
   (a) Allows an object's data to be converted to a sequence of bytes.
   (b) Creates a remote reference for an object.
   (c) Enforces concurrency control to ensure that concurrent accesses to an object are serialized.
   (d) Creates client and server stubs for an object.

9. Compared with SOAP, REST:
   (a) Is based on remote method calls.
   (b) Identifies resources in the URL of an HTTP command.
   (c) Uses XML for creating a message within the HTTP message.
   (d) Is not tied to a single language.
13. With DCE and Microsoft RPC, the Unique Universal Identifier (UUID) is used to uniquely identify:
   
   (a) A client.
   
   (b) **An interface to a set of procedures.**
   
   (c) A communication session.
   
   (d) A server machine.

29. While NFS was originally designed to be stateless, state was first added to support:

   (a) **File locking.**
   
   (b) Coherent client-side caching.
   
   (c) RPC-based remote file access.
   
   (d) File replication.
The End