Name: _____

CS 6/73201 **Exam #2** **Advanced OS**

Wednesday 14 March 2001

1. **Remote procedure calls typically support call-by-value parameter passing, but not call-by-reference.**

   a. **Why is call-by-reference not supported? (4 points)**

      Call-by-reference passes a pointer to a particular memory location, but since in a distributed system the remote procedure may be on a separate machine (with a separate memory). If so, that pointer would not be valid in the remote procedure call.

   b. **What mechanism is used instead of call-by-reference to provide similar functionality, and how does it work? (6 points)**

      Call-by-copy / restore is used instead. This mechanism passes the data to the remote procedure call, where the passed copy is modified; then the modified data is returned to the caller, where it replaces the original data.

2. **A common application of distributed systems is to provide access to a server of some kind.**

   a. **Briefly describe how a server can be structured using the dispatcher-worker model and threads. (10 points)**

      The server is built as a dispatcher thread and a number of worker threads. Requests to the server are received by the dispatcher, who assigns a worker to process each request. The number of worker threads may be fixed in advance, or new workers may be allocated as necessary and terminated after processing a client request.

   b. **Using threads in a server like this, which would work better — user-level threads or kernel-level threads — and why? (10 points)**

      Assuming the worker threads have to block, either because of I/O or page faults, it would probably be best to use kernel-level threads, so that if one thread blocks it will not cause the entire server to block. Also, with the kernel knowing about the threads, the server as a whole may receive more than one slice of CPU time. If the worker threads didn't have to block (which seems unlikely), then user-level threads might be better as they are faster, cheaper to create, etc.

3. **A distributed shared memory system may be structured with non-replicated, migrating pages.**

   a. **Briefly explain how the hardware and operating system work together to cause a remote page to migrate to the current machine. (10 points)**

Name: _____

When the system tries to access a page on a remote machine, a page fault occurs (this happens in hardware). The part of the OS & distributed shared memory system responsible for processing the page fault then sends the appropriate messages to the remote machine, causing the page to be sent to the machine that just accessed it. The page then moves into memory, and control returns to the process that caused the page fault.

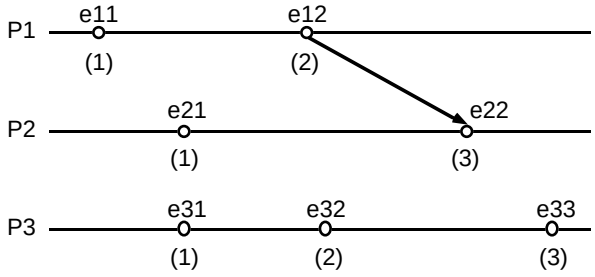
b. What additional complications would be introduced if the distributed shared memory system were to support replicated pages instead? (5 points)

The system would have to keep the various replicated copies of each page consistent, either by snooping or some other mechanism to keep the copies up to date, or by invalidating all the other copies when one copy changes.

4. In Christian's algorithm for synchronizing physical clocks, nodes send a request to the time server. How do they account for the network transmission delay? (10 points)

When a node sends a request to the server, it measures the round-trip transmission delay Dtrans required to receive the response from the server. Then it sets its local time to the time received from the server, Tserver, plus 1/2 of Dtrans, assuming that half the transmission delay can be attributed to receiving the response from the server.

5. Using Lamport's logical clocks, if a $\rightarrow$ b, then C(a) < C(b). However, if C(a) < C(b), we can not say that a $\rightarrow$ b. Draw and explain an example to illustrate this last statement. (15 points)



In the example above, taken from Lecture 12, C(e11) < C(e33), but it is not true that e11 $\rightarrow$ e33 (there is no path between those two events).

6. Using Garcia-Molina's bully algorithm to elect a coordinator, what happens if two processes independently discover that the coordinator is not responding? (10 points)

Both of them will then independently start an election by sending election messages to their higher-ups, and of those two, the higher-priority one will send an answer message to the lower-priority one, causing it to wait for a coordinator message. Regardless of how many elections are in progress, the process with the highest priority, whether it is one of these two processes or not, will eventually win.

Name: _____

7. Suppose a particular process A in a distributed system needs exclusive access to a particular shared resource on occasion, but not very often.

a. If the system uses the central coordinator algorithm for mutual exclusion, and process A is not the coordinator, what responsibility does A have to the system as a whole? (5 points)

If A has no interest in the shared resource at the moment, it does not participate in the central coordinator algorithm at all.

*Note that I did **not** ask you to tell me how A gets in the critical section, but instead asked about A's responsibility to the system as a whole if it does not need access to the shared resource very often. What I **asked** was much easier to answer than what some people **thought** I asked!*

b. If the system uses the Ricart and Agrawala algorithm for mutual exclusion, what responsibility does A have to the system as a whole? (5 points)

Even if A has no interest in the shared resource at the moment, it must send a reply message to every process that sends it a request message.

c. If the system uses the Suzuki and Kasimi broadcast algorithm for mutual exclusion, what responsibility does A have to the system as a whole? (5 points)

Even if A has no interest in the shared resource at the moment, it must update its vector RN with the SN sent to it by every process that sends it a request message.

d. If the system uses the Raymond tree algorithm for mutual exclusion, what responsibility does A have to the system as a whole? (5 points)

Even if A has no interest in the shared resource at the moment, if it is in the path of a request message or the token, it must perform the appropriate actions. If it receives a request message, it must put the requestor's id on its request_q, and if the queue was empty before the addition it must forward the request toward the holder of the token. If it receives the token, it must delete the first entry in its queue, send the token toward the deleted entry, and update its holder variable to point in that direction; if the queue is not empty, it must also send the request message toward the holder of the token.