Name: _____

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

Wednesday 24 March 1999

1. Briefly describe the Nachos Post Office. (10 points)

Archna Kalra's "Salsa — An Operating Systems Tutorial" has a nice description, which would make an appropriate answer:

"The PostOffice class is defined in the the files post.h and post.cc in the network directory. This class defines a post-office or a collection of mailboxes. It is a synchronized object that provides two main operations: Send - send a message to a remote machine, and Receive - wait until a message is in the mailbox, then remove and return it. Incoming messages are put by the PostOffice into the appropriate mailbox, waking up any threads waiting on Receive."

Her description also goes on to say the following, but this much detail was not necessary:

"The file post.h also defines three other classes - Mail, MailHeader and MailBox. Mail defines the format of an incoming/outgoing message. The constructor function of this class initializes a single mail message, by concatenating the PacketHeader (explained below) and the MailHeader. The MailHeader is part of the message header. It is prepended to the message by the PostOffice before the message is sent to the Network and contains the the destination mailbox address, the source mailbox address and the number of bytes in the message data. Thus, the packet that is delivered to the remote machine contains the actual message, the MailHeader and the PacketHeader (the Mail object). Finally, the MailBox class defines a single mailbox for messages. Incoming messages are put by the PostOffice into the appropriate mailbox, and these messages can then be retrieved by threads on that machine. A mailbox is implemented simply as a list of messages using the SynchList object (synchlist.h(cc))."

2. Two alternatives to synchronization are semaphores, and locks and condition variables.

a. Is it possible to "lock" a critical section of code in two separate processes using a semaphore, and then use another semaphore inside that critical section to signal between the processes (i.e., one process does a semaphore wait and the other does a semaphore signal)? Explain. (7 points)

No. Suppose you lock a critical section containing a "signal" in one process, and a critical section containing a "wait" in another process. Then once the "wait" blocks, since it doesn't exit the critical section the other process can never get inside its critical section to do the corresponding "signal", so the "wait" can never proceed.

b. Is it possible to do this using locks and condition variables? Explain. (8 points)

Yes, condition variables are designed to perform a wait and signal inside a locked region. Unlike a semaphore "wait", a condition variable "wait" releases the lock before going to sleep, so that another process can grab the lock, get inside the locked region, and use a

"signal" to wake it up. Then when the waiting process wakes up, it reacquires the lock and continues in the critical section.

3. Briefly explain (i) the hardware behind a physical clock, and (ii) how it is used to determine time-of-day. (15 points)

A clock is an electronic device that counts oscillations in a crystal at a particular frequency. This count is typically divided, and stored in a counter register. The clock can then be programmed to generate interrupts at regular intervals.

The clock value can be scaled to produce time of day, given a known time for some particular counter value (i.e, the clock must be externally synchronized at some point).

4. With logical and vector clocks, if $a \rightarrow b$, then $C(a) < C(b)$. Is it also true that if $C(a) < C(b)$, then $a \rightarrow b$? Explain. (10 points)

No, it is not true for logical clocks — see diagram on slide 10 of Lecture 13.

Yes, it is true for logical clocks, since they pass additional information around — that is their primary advantage over logical clocks!

5. For each of the following mutual exclusion algorithms, clearly explain either (i) how it guarantees "happened before" ordering of the requests, or (ii) why it does not make this guarantee. Note that I am <u>not</u> asking for a summary of the algorithm, but instead an answer to this specific question! (28 points)

a. Lamport's algorithm

Guarantees "happened before" by satisfying requests according to their logical / vector clock timestamps.

b. Suzuki and Kasimi's algorithm

It could guarantee "happened before" if logical / vector clock timestamps are sent with the requests, and those timestamps are used to sort requests into the token queue Q. Otherwise, it does not guarantee "happened before" order.

c. Le Lann's token ring algorithm

Does not guarantee "happened before" because a process that just gets "missed" by a token passing by it could request the critical section, but another process further along the token's path that makes a later request could get the token before this missed process.

d. Raymond's tree algorithm

Does not guarantee "happened before" for reasons similar to those in part c.

Name: _____

6. For <u>each of the three types</u> of messages sent by Garcia-Molina's bully algorithm, does the algorithm work correctly if an individual message gets lost? Explain your answer. (15 points)

Lost election message — probably OK, since someone else will probably eventually notice that the coordinator isn't responding, and they will start an election

Lost answer message — not OK, if an answer from a live higher priority process gets lost, the process will declare itself the coordinator and send out coordinator messages. However, the higher priority thread will also be holding an election, and either it or someone higher will eventually win, so that winner will later send out coordinator messages, which will…

Lost coordinator message — OK but not great, since if an active process misses getting a coordinator message, it will try to contact the old coordinator, which is presumably not responding, so it will start a new election. However, if the lost coordinator message was going to an inactive process, then the loss doesn't matter.

7. What is the big (i) advantage and (ii) disadvantage of self-stabilizing algorithms? (7 points)

Advantage — if the system enters an illegitimate state state, it will eventually reach a legitimate state. Other advantages include: (1) does not need to be initialized, (2) recovers from transient failures, and (3) adapts to changes in system topology.

Disadvantage — while it's recovering, the system does not guarantee correct execution.