Name: _____

CS 6/73201 **Exam #1** **Advanced OS**

Wednesday 16 February 2000

1. In trying to distinguish between parallel and distributed systems, I often used the terms "tightly coupled" and "loosely coupled".

a. What is the distinction between "tightly-coupled hardware" versus "loosely-coupled hardware"? (7 points)

Tightly coupled usually means that the processors share a common clock and memory, communicate frequently over some fast communication medium, and are in close physical proximity to each other (in the same box). Loosely coupled usually means independent systems communicating less frequently over a standard network.

b. What is the distinction between "tightly-coupled software" versus "loosely-coupled software"? (7 points)

Tightly coupled means the software (we are usually talking about the operating system here) provides the illusion of being a single OS. All the various "transparencies" are supported as much as possible. Loosely coupled means independent pieces of software that cooperate with each other in some fashion.

2. For each of the network topologies below, explain when (if ever) it might be suitable for use in a physical network.

a. fully connected (7 points)

Each message takes a single hop to reach its destination, so this topology would be good when speed and high throughput are important. Furthermore, it is good when fault-tolerance is important, as the loss of a single node or network link affects only that node or the nodes connected to that link. Unfortunately, it also requires a lot of wiring and thus is not very scalable, so it would only be suitable if those factors could be ignored, which realistically means a small number of processors in close physical proximity.

b. star (7 points)

Each message requires only two hops to reach its destination, so this topology would be good when moderately high speed and throughput are important. It is also fairly fault-tolerant, as the loss of a single node (other than the central node) or network link affects only that node or the nodes connected to that link, but if that central node is important, then this might not be the best network topology. In practice, this topology is often used for computers connected to a network hub or switch (in this situation, if the hub or switch fails, it is simply replaced by another one).

c. bus (7 points)

Since this topology requires only a single wire passing by each node, it is very inexpensive. However, since the communication medium must be shared, scalability isn't as good as most of the other topologies.

3. In message passing, explain the distinction between direct communication and indirect communication (using mailboxes). (10 points)

In direct communication, the remote process being contacted is explicitly named, which means there can be at most one link between two processes, and a process wanting to receive messages from two or more processes must explicitly receive from each process in turn.

In indirect communication, a receiver can have one or more mailboxes, and any remote processes can send to those mailboxes. Thus there can be more than one link between two processes (multiple mailboxes, perhaps for different types of data), and a receiver can simply check its mailbox to get a message from anyone who sent to that mailbox.

(Note that this distinction does not have anything to do with synchronous versus asynchronous communication, the size of the message buffer, etc. Those are all separate issues.)

4. In the Ethernet protocol, explain how contention is avoided between multiple processes that want to send data on the shared bus at the same time. (20 points)

A process wanting to send first listens to the bus, and defers attempting to transmit until the bus is not in use. Then it broadcasts its data on the bus, but as it does so, it listens to the bus. If another process also tries to broadcast at the same time, the data will become garbled. In this case, both processes will notice the collision, and will "jam" the bus to make sure the other notices the collision. Then each will wait a random amount of time and try again; if they collide again, this random amount of time will increase, lessening the chance of another collision.

5. In many situations, threads are much more suitable than processes. What are the advantages of threads? (17 points)

Threads are cheap, as they use very few resources (stack space, registers). Context switching among threads is very fast, not much slower than a procedure call. Threads are a good match for many problems, such as a server, where the system can be modeled with many "workers". Threads can take advantage of multiprocessor systems.

There are also many other advantages specific to user-level or kernel-level threads.

6. For each of the following architectures, briefly explain how remote memory is accessed. (Do not explain the <u>caching</u> mechanism to me, only the mechanism for <u>accessing</u> remote memory.)

a. UMA multiprocessors / SMPs (6 points)

A global shared memory is connected to all processors via a bus. All processors have direct hardware access to the shared memory through normal memory-access machine instructions.

b. NUMA multicomputers (6 points)

A separate memory is connected to each processor, or perhaps to a group of processors. Each processor sees these memories as one large memory space, and has direct hardware access to the entire memory. However, access to local memories is faster than access to remote memories, hence the name "non-uniform" memory access.

c. Distributed shared memory (6 points)

A separate memory is connected to each process, and there is no direct hardware access to remote memories. However, the operating systems provides the illusion of a single memory space, and uses the page fault mechanism and message passing to either bring a remote page (or a copy of it) to the local machine.