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

Due in class on Wednesday 17 February 1999

1. (Exercise 1.9 from *Distributed Operating Systems*) What are the main differences between a network operating system and a distributed operating system?

In a network OS, the users know they are interacting with multiple machines; in a distributed OS, they are presented with the illusion of a single system. In a network OS, there are a number of computers running loosely-coupled but independent operating systems that communicate as necessary to perform a task; in a distributed OS the computers run a single system-wide OS. A network OS provides little fault-tolerance since each user is tied to a specific machine; a distributed OS provides considerable fault tolerance since one computer is as good as another in many situations.

2. (Exercise 2.3 from *Distributed Operating Systems*) The CSMA/CD scheme for medium-access control allows random access to the shared medium, detects collisions, and takes necessary steps to retransmit if a collision is detected. Is it possible to devise a random-access scheme for medium-access control in which collisions never occur (collisions are totally avoided)? If no, explain why. If yes, describe a possible scheme of this type along with its advantages and disadvantages.

One possibility would be for a site to listen for the presence of a signal for at least the collision interval (the time required for a signal to propagate from one end of the shared medium to the other) before transmitting. The motivation for this method is that with CSMA/CD, a collision can occur slightly after the first of the colliding sites begins transmission. However, if the site that started transmitting second had to wait for at least one collision interval of a "quiet" transmission medium before transmitting, it would never have started transmitting and thus would never have caused the collision. Unfortunately, although this method reduces the chances of a collision, it doesn't guarantee that there will be no collisions.

Another possibility would be for each site to be connected to a central coordinator, which arbitrates access to the shared medium and avoids collisions, but this method has the low fault-tolerance that comes with a central coordinator. Yet more methods exist, but are beyond the scope of this course, although they may be discussed in CS 4/55201 Computer Interconnection Networks. (Note that methods such as token-ring avoid collision, but aren't really random-access methods since access to the medium is very structured.)

3. (Exercise 3.21 from *Distributed Operating Systems*) What is the main purpose of using an acknowledgment message in an IPC protocol? Are acknowledgment messages always needed for reliable communication? Give reasons for your answer.

The main purpose is to provide reliability by ensuring that the intended recipient has actually received the message, and that it hasn't gotten lost due to a communication failure or a server crash.

A separate "acknowledgment" message is not needed if the communication protocol requires the recipient to send a follow-up message with additional information; in this situation that

follow-up message can also serve as the acknowledgment. As examples, see the three-message and four-message reliable IPC protocols in the text.

If the sender uses a timeout mechanism, and will retransmit the message if it does not receive a acknowledgment, then reliable communication can be achieved even when the acknowledgment fails. However, this method assumes that there is generally an acknowledgment, so it doesn't really answer this question.

4. (Exercise 4.6 from *Distributed Operating Systems*) Achieving complete transparency of an RPC mechanism that allows the caller and callee processes to be on different computers is nearly impossible due to the involvement of the network in message communication between the two processes. Suppose an RPC mechanism is to be designed in which the caller and callee processes are always on the same computer. Is it possible to achieve complete transparency of this RPC mechanism? Give reasons for your answer.

For complete transparency, the RPC mechanism should provide both syntactic and semantic transparency. Syntactic transparency can be provided by the RPC system, but semantic transparency is more problematic. For "normal" RPCs, semantic transparency is not possible since the lack of shared address space makes it difficult to pass pointers to large data structures; however this might be possible in "same-computer" RPCs if sharing data between processes is supported by the OS. (Note that two processes do not generally share the same address space.) Similarly, for "normal" RPCs, the program must be prepared to handle errors that do not occur with regular procedure calls, such as processor crashes and communication failures; however, those problems would not be a problem with "same-computer" RPCs. Finally, "normal" RPCs take much longer than RPCs, and it might be necessary to compensate for this delay, but this problem would not be so pronounced with "same-computer" RPCs. So, in summary, it might be possible to achieve complete transparency on a single computer.

5. (Exercise 5.24 from *Distributed Operating Systems*) Are DSM systems suitable for both LAN and WAN environments? Give reasons for your answer.

DSM can be implemented in either a LAN or WAN environment, but it is more suitable for a LAN environment. The WAN's lower transmission speed and higher error rate can result in much lower performance as pages are moved around in the DSM, unless the system is very carefully tuned, supports a weaker consistency model, etc. Furthermore, the machines in a WAN are more likely to be supplied by different vendors, run different operating systems, etc., and supporting a more heterogeneous environment is always more difficult.