

CS 574

Final Exam

December 13, 2002

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your social security number, but not your name, on each sheet of paper you turn in
- show your work whenever appropriate. There can be no partial credit unless I see how answers were arrived
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 12:30 to finish the exam. The questions are equally weighted.

1. A criticism of microkernel based operating systems has been that they are inefficient in comparison to conventional monolithic operating systems. In the context of a distributed system, would you expect a microkernel operating system to be more or less efficient than a system build as an application layer atop a network of computers running conventional operating systems?

If the distributed system were microkernel-based, messages from process to process would go to the local kernel, to the local network driver, to the remote network driver, to the remote kernel, to the remote process.

If it were an application layer, messages would go from the local process, to the local kernel, to the local distributed system server, to the local kernel (which would include the network driver), to the remote kernel, to the remote distributed system server to the remote kernel, to the remote process. So you'd expect the microkernel approach to be more efficient.

An alternate interpretation for "application layer" which occurred to me later (and, once it did occur to me strikes me as more reasonable than the interpretation on which I based the answer above) would be an implementation as a library, not requiring servers (more like MPI, for instance). In this case, a message from process to process would go from the local process, to the local kernel, to the remote kernel, to the remote process. In this case, the monolithic approach might be more efficient after all.

2. Would it be possible to build a CORBA server on top of RPC? Would it be possible to build it on top of MPI? If both are possible, which would be more reasonable?

You can certainly build an ORB as an RPC server - sending messages to the ORB is easily handled by defining RPC calls. As for MPI, it depends on your definitions of "possible" and "CORBA." Generally CORBA processes are started independently and don't have knowledge of each other; MPI assumes cooperating tasks which are all aware of each other (and in the normal case starts them all at once). If you regard the MPI version as possible, it's certainly not as reasonable.

3. Consider the following memory reads and writes:

P1	W(x)1	W(y)1			
P2	W(x)2	W(y)2			
P3		R(x)1	R(y)2	R(x)2	R(y)1
P4		R(x)2	R(y)1	R(x)1	R(y)2

- (a) Name a memory consistency model for which this diagram *is not* valid. Why does it violate the model you picked?

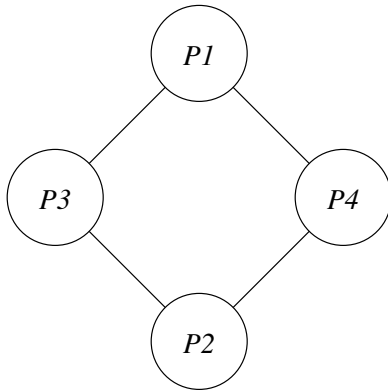
It's not valid for any model that requires all processors to see all writes in the same order. Of the models we discussed, the diagram is not valid for strict and sequential consistency.

- (b) Name a memory consistency model for which this *is* valid.

It's valid for any model in which the order of writes by different processors can be seen in different orders by different processors. Basically, that's everything from Processor Consistency on down. A few students noted that it's not valid for LRC consistency since two processors can't have write access to the variables at the same time; that's not the sense in which I meant "invalid," but I'll take it.

- (c) Draw a picture showing a network topology which could, in combination with the consistency model you picked in 3b, result in the given diagram.

Basically, a topology is required in which it's reasonable for P3 and P4 to get their writes from P1 and P2 in arbitrary order (P3 gets P1's write of X first and P2's write of Y first, with P4 getting the opposite), so they should be equidistant. Here's a topology that meets the requirement:



For that matter, a bus would meet the requirement as well.

4. One of the differences between ReiserFS and a conventional Unix-like file system such as ExtFS is that ReiserFS lets several “tails” (the incomplete last block of a file) be combined in a single block. Would it be possible to modify Ext2FS to do this, as well? If not, why not? If so, briefly outline an approach.

Yes, it's possible. You'd need to add some i-node fields specifying what data block contained the file's tail, and the tail's offset in the data block. There's also be a bit of extra bookkeeping; in addition to having free vs. busy data blocks, you'd also need to keep track of the ones being used for tails.

5. If you write code assuming Mesa monitor notify/wait semantics but your system uses Hoare signal/wait semantics, it will still work; if you assume Hoare semantics but the system uses Mesa semantics it won't. Why?

Hoare monitors require a waiting process be restarted immediately after signal; Mesa monitors don't. So a program assuming Hoare monitors can write the code as

```
if (bool) cond.wait;
```

while Mesa monitors have to be written as

```
while (bool) cond.wait;
```

since the bool can change value between the notify and the resumption.

In the latter case, if the system actually uses Hoare monitors no harm is done since just rechecking a condition won't hurt anything (assuming no side effects).

6. Suppose NMSU had a mobile computing infrastructure, in which all of campus is covered by overlapping wireless zones. Any given user has at least one zone on campus whose server is trusted, but does not typically “know” all of the zones on campus.

Suggest a way that a user, walking across campus and entering a new zone, can determine that the new zone is a legitimate part of the NMSU infrastructure and not a “pirate” running a wireless access point distributing incorrect information.

In answering this question, you can assume (1) arbitrary encryption is available, and (2) the legitimate NMSU wireless access points can cooperate in assisting the user in authenticating a new zone.

The basic requirement here is that when the user crosses from one zone to the next, the old zone can authenticate the new zone to the user, but that the user can't make use of the information to build a pirate access point. An obvious approach would be to use a public-key cryptosystem; when the user is changing zones the old zone can send the user the public key for the new zone. If the new zone sends messages encrypted using its private key, the user can use the public key to decrypt. The user can't set up a pirate, since they don't have the zone's private key.