Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.824 Spring 2006

Midterm Exam

Please write your name on this cover sheet and on any exam pages you detach from this sheet.

Some questions may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit.

There's a feedback form at the end of the exam which we'd like you to fill out if you have time.

You have 80 minutes to complete this exam.

THIS IS AN OPEN BOOK, OPEN NOTES EXAM.

Please do not write in the boxes below.

1 (xx/15)	2 (xx/15)	3 (xx/40)	4 (xx/30)	Total (xx/100)

Name:

I Lab 5

The server implementation of the NFS GETATTR RPC only needs to read blocks; it does not modify any blocks. Nevertheless in Lab 5 GETATTR needs to lock the relevant file handle.

1. [5 points]: Explain what would go wrong if GETATTR did not lock.

You'd like to make your Lab 5 file system continue to work even if one ccfs crashes (out of a set of ccfs's serving the same file system). You just want to ensure that the other ccfs's can successfully continue using the shared file system despite one ccfs crash. It's OK if some sub-set of the last few NFS RPCs served by the crashed ccfs ”disappear” (you do not have to enforce any kind of Echo-like prefix rules). It's clear that it will be a problem if a ccfs crashes while holding locks. You modify your lock server to use leases (as in the Echo file system) so that the lock server can take back a lease from a crashed ccfs after the lease expires. You also modify ccfs to re-acquire each lease it holds before the lease expires.

2. [10 points]: Are these modification sufficient? If yes, briefly argue why; if no, describe a specific situation that would cause trouble.

II Java RMI

The paper *A Distributed Object Model for the Java System* describes Java's RMI RPC system. The paper mentions that a remote object's server garbage-collects the object: when there are no local or remote references to an object, the server can discard the object and free its memory. Suppose that an RMI implementation implements garbage collection by keeping a single reference count in the server for each of its objects, holding the total number of references to the object, local and remote. Each client notifies an object's server whenever the client creates or destroys a reference to the object. For example, at the end of this client code the acct object's reference count in the server will be two (assuming no other clients have references to the object):

```
acct = bank.lookupAccount("12345");
acct1 = acct;
```

3. [5 points]: Explain a performance problem with the reference counting implementation outlined above, and explain how to fix it.

If an RMI client crashes while holding one or more reference to an object, it should nevertheless be possible for the object's server to eventually garbage-collect the object once no live clients have references to the object. It's OK if the server garbage-collects an object referred to by a client that is alive but unreachable due to network failure.

4. [10 points]: Suppose an object's server has noticed that a particular host has crashed. Explain how the server could eventually garbage-collect the object despite the possibility that the host had references to the object when it crashed. Explain any modifications you might need to make to the design outlined above.

III Porcupine

5. [10 points]: Consider a performance experiment like the one in Figure 6 of the Porcupine paper, but with all mail addressed to a single recipient. With this workload, roughly how many messages per second would a 30-node non-replicated Porcupine system be able to handle with policy D1? With D2? With D4? Explain your answers.

6. [10 points]: The R (random) line in Figure 6 is slightly higher than the D1 line. What Porcupine design feature(s) does D1 involve that R does not? What does the graph imply about the value of those feature(s)?

7. [10 points]: D2 has better performance in Figure 6 than D1. Why?

8. [10 points]: Section 4.3 of the Porcupine paper says that Porcupine replicates entire fragments: the fragment list is really a set of pairs of nodes, and the two nodes in a pair each hold an identical copy of the relevant fragment. A simpler design might instead put each message in two different fragments. Then a user's fragment list would be a set of nodes (rather than a set of node pairs), and when a new email message arrived, Porcupine would choose a fragment on each of two lightly loaded nodes and append the new message to both fragments. Why do you suppose Porcupine didn't take this simpler approach? What are the advantages of having explicitly replicated fragments?

IV Memory Consistency

The Speedy Compute Cluster Corporation (SCCC) is designing a parallel computer called the SCC. The SCC is intended to run existing parallel applications that were originally written for an expensive computer with multiple CPUs and a single memory system shared among those CPUs. The parallel applications have many threads (one per CPU) that communicate intermediate results through the shared memory.

The SCC will consist of a set of workstations connected by a LAN. Each workstation will have a complete copy of the shared memory in its local RAM. Each workstation's operating system will intercept the local application thread's loads and stores to memory, and execute a network protocol to keep the other workstations' copy of the shared memory up to date.

SCCC has hired you to help them understand how their memory protocol will affect their customers' programs. SCCC's main customer spends all its time running this application:

```
// these global variables are correctly initialized
// in all CPUs’ memories before any CPU starts
// running the program:
int done = 0;
int x = 1;

cpu0(){
  x = x + 1;
  done = 1;
}

cpu1(){
  while(done == 0){
    // do nothing...
  }
  x = x * 3;
  printf("%d\n", x);  // print x
}
```

Based on a student's comment that they overheard while attending 6.824, SCCC's engineers have arrived at the following design for their shared memory protocol. When an application reads from memory, the load proceeds immediately from the local RAM. When an application writes to memory, the operating system causes the application to pause, sends a WRITEMEM message containing the written value, the memory address, and the current real time to every workstation (including itself), waits for responses from all workstations (including itself), and then lets the application continue.

Every workstation is always capable of receiving and processing a WRITEMEM message (perhaps in an interrupt). Each workstation processes incoming WRITEMEM messages one at a time. Each workstations maintains a timestamp on each application memory location in its RAM, containing the time in the WRITEMEM message that last updated the memory location. A workstation only updates a memory location in response to a WRITEMEM if the time in the WRITEMEM is greater than the timestamp stored for the memory location. A workstation always replies to a WRITEMEM, even if it did not update. An application is never allowed to directly store into the local RAM; instead, each workstation's local memory is only updated by WRITEMEM messages. You can assume that no two WRITEMEMs occur at precisely the same time (i.e. no two WRITEMEMs will have exactly the same timestamp). Each memory location's timestamp is initialized to zero when the program starts; this is smaller than any node's clock.

9. [10 points]: Assume all the workstations have exactly synchronized clocks; at any given instant, they all yield the same timestamp. List the possible outputs (final values of x) of the application shown above.

10. [10 points]: Suppose that the workstation clocks do not agree on what time it is; some are ahead, others are behind. List the possible outputs (final values of x) of the application shown above.

One of SCCC's engineers suggests that programs will be more likely to run correctly with the following protocol addition. When a WRITEMEM arrives, a workstation should compare the timestamp in the WRITEMEM with the workstation's clock; if the WRITEMEM's timestamp is greater than the clock time, set the clock time to the WRITEMEM's timestamp.

11. [10 points]: Assuming this modification, list the possible outputs (final values of x) of the application shown above.

End of Exam

6.824 Questionnaire

We’d like to ask you for feedback to help us make the course better, both for the rest of this semester and in future years. Feel free to tear off this page and hand it in separately from the exam for anonymity.

If you could change one thing about 6.824 to make it better, what would it be?

What’s your favorite aspect of 6.824? That is, what *shouldn’t* we change?

What could we do to make the paper discussions more informative and useful?

Any other suggestions about the labs, lectures, TA, choice of papers, or projects?