15-440 Distributed Systems
Midterm

Name:
Andrew: ID

October 7, 2010

- Please write your name and Andrew ID above before starting this exam.
- This exam has 11 pages, including this title page. Please confirm that all pages are present.
- This exam has a total of 100 points.

Question	Points	Score
1	3	
2	3	
3	3	
4	4	
5	3	
6	3	
7	2	
8	3	
9	24	
10	28	
11	24	
Total:	100	

A Short Answers

1. (3 points) RPC tries to make remote procedure calls look the same as local procedure calls. But the illusion isn't perfect. Circle all of the options that correctly describe differences between a local function call and an RPC:

   A. RPC calls require an extra parameter to identify the server.

   B. RPC calls may have higher or variable latency.

   C. RPC calls are limited to call-by-value.

   D. RPC calls have different (more) failure modes.

   E. Local function calls require a `malloc` of a heap object to represent the return pointer.

2. (3 points) Why does the condition variable call to wait take a mutex as an argument?

3. (3 points) Recall that NFS uses a timeout-based mechanism to provide weak consistency, where AFS uses a callback mechanism to provide close-to-open consistency. Assume a file server has 100 clients. These clients have all opened the same file read-only. They have no other open files. In this scenario, would an NFS server or an AFS server experience lower request load? Why (briefly)?

4. (4 points) Using the 5 year-old single-disk computer in his office, Prof. Evil von Ahn repeatedly measured the read throughput of a 1 GB file on two systems: an unknown remote file server in the SCS machine room, and the local disk in Prof. Evil's computer itself. The throughput from the remote server was 12 times higher than Prof. Evil's local disk.

   Of the possible differences between the unknown remote server and Prof. Evil's computer and the trends in computing hardware, what factors *alone* could likely account for this performance difference? (Circle all correct answers.)

   A. The remote server disk(s) spin 12 times faster than the disk in Prof. Evil's computer.

   B. The remote server uses RAID 0 and has many disks.

   C. The network bandwidth is 12 times higher than the disk bus in Prof. Evil's computer.

   D. The remote server uses RAID 5 and has many disks.

   E. The remote server serves the file from a memory cache while Prof. Evil's computer repeatedly serves the file from disk.

   F. Prof. Evil runs Windows.

   G. The remote server is an AFS server.

   H. The remote server is a Sun NFS server.

5. (3 points) Name 3 reasons why you would use threads instead of processes for a problem requiring concurrency (doing multiple tasks in parallel).

6. (3 points) List and describe the three characteristics that a solution to a synchronization (mutual exclusion) problem must have.

7. (2 points) Describe two disadvantages of a ring-based algorithm (one that passes a token around a ring) for implementing a distributed mutex?

8. (3 points) For which of the following applications would UDP be preferable over TCP, ignoring annoying concerns such as firewalls that might block some protocols?

A. Streaming a live video over the internet	TCP / UDP
B. Instant messaging/email	TCP / UDP
C. Logging in to your bank website	TCP / UDP
D. Voice over IP	TCP / UDP
E. Large file transfers	TCP / UDP
F. Looking up a very small value from a directory service	TCP / UDP

B Going RAIDing

9. The growth of disk capacity has been outpacing the growth in speed, both for seek latency and transfer rate. A few years ago, you built a RAID array to reliably store your collection of course notes (you take very detailed notes – many hundreds of gigabytes of them). At the time, you built the system as a RAID5 (rotating parity) array using six drives. You picked a bunch of cheap 120GB drives off of newegg.com. The parameters of the drives are:

Capacity	120GB
Seek latency	7ms
Rotational delay	3ms
Transfer speed	60MB/sec
MTTF	1,000,000 hours

Assume where it matters that the disks are full—your data occupies *all* of the available capacity on the array—and that the parameters are given in SI units.

(a) (4 points) Using RAID5, and ignoring filesystem overhead, etc., what is the usable capacity of your RAID array?

(b) (3 points) How long will it take to write one byte of data to this RAID, assuming you write it to a randomly chosen location in the filesystem?

(c) (4 points) Pittsburgh goes to the superbowl again. You live near Atwood, and you worry that in the ensuing riots, your computer may be destroyed, so you decide to copy all of the data off of your computer to a remote server on the Internet. How quickly can you read all of the data off of the raid in huge sequential blocks to make a copy of it?

(d) (4 points) Disaster strikes! While you were copying your data, a power surge blew up one of the disks in the array. Your array is now "degraded." You have a spare disk, and you put it in the array in place of the dead one. How long will the rebuild process take? State any assumptions you make.

(e) (3 points) It's now 2009, and you have a lot more course notes. You decide to build a new RAID array using modern disks:

Capacity	1000GB
Seek latency	5ms
Rotational delay	2ms
Transfer speed	80MB/sec
MTTF:	1,000,000 hours

You've once again filled up your array of six disks. How long would it take to do a RAID rebuild using this new array?

(f) (4 points) Assume that disk failure probabilities are completely independent. What is the probability of your RAID array experiencing a second disk failure *during* the rebuild? (Hint: Use the MTTF, and state any assumptions you make.)

(g) (2 points) Do you expect that probability to be higher or lower in reality? Explain your answer briefly (1–2 sentences).

C The Rowing Cartographers

10. Adam, Becky, and Cartman decide to try out rowing crew. Unfortunately, there's only one boat left that can seat three people (it leaks a bit), and there are only four oars. A rower must have two oars in order to row, or else the boat will go in circles.

    Adam proposes that to arbitrate access to the oars, they should be placed in the center of the boat, and every rower will follow a simple protocol:

```
while (!at destination) {
  recover_strength();
  grab_one_oar();  /* may block */
  grab_one_oar();  /* may block */
  row();
  row();
  row_your_boat();
  drop_one_oar();  /* will not block */
  drop_one_oar();  /* will not block */
}
```

    (a) (5 points) Explain clearly why this plan will *not* lead to deadlock. Note that a one sentence answer probably isn't enough, but six sentences starts to look like too long an answer. Refer to the characteristics of deadlock and mutual exclusion.

(b) (5 points) After rowing down the river, an octopus jumps into the boat in a spare seat. The octopus wants to row too – but it has eight arms. Let's generalize the solution a bit to handle an arbitrary number of rowers, R, where each rower needs a particular number of oars in order to row.

Let's characterize the scenarios by:

- R - the number of rowers
- A - the total number of arms summed across all rowers

For example, the boat with Adam, Becky, Cartman, and Ozzy the Octopus, would have $A = 14$ arms and $R = 4$ rowers. The generalized rowers use this protocol:

```
int n_oars = my_number_of_arms();

while (!at destination) {
  recover_strength();
  for (int i = 0; i < n_oars; i++)
    grab_one_oar();  /* may block */

  row();  row();  row_your_boat();

  for (int i = 0; i < n_oars; i++)
    drop_one_oar();  /* will not block */
}
```

Phrased in terms of A and R, what is the smallest number of oars you need to put in the boat to ensure that deadlock cannot occur? Explain. Your answer must be organized and convincing.

(c) (10 points) Let's build this code. Provide code for three functions:

```
typedef struct boat { ... } *boat_p;
void boat_init(boat_p b, int n_rowers, int n_oars);
void grab_one_oar(boat_p b);
void drop_one_oar(boat_p b);
```

You may use only mutexes and/or condition variables, NOT semaphores or lower level atomic primitives. Assume that the environment is "error-free" (allocations always succeed, etc.). The mutex and cond var operations are:

```
mutex_init(mutex_t *mp);
mutex_lock(mutex_t *mp);
mutex_unlock(mutex_t *mp);

cond_init(cond_t *cp);
cond_wait(cond_t *cp);
cond_signal(cond_t *cp);
cond_broadcast(cond_t *cp);
```

Declare the struct boat here, and write the init function (on the next page). Assume that n_oars has been set appropriately for the needs of the rowers:

```
typedef struct boat {











} *boat_p;
```

```
void boat_init(boat_p b, int n_rowers, int n_oars) {







}
```

(d) (8 points) Write `grab_one_oar(boat_p)` and `drop_one_oar(boat_p)`:

D Shamport Clocks

11. Suppose each computer in a distributed system keeps an approximate real-time clock R_i in addition to a Lamport-like clock L_i. A computer's real-time clock R_i is an always-increasing integer (i.e., for any two reads r_1, r_2 of R_i such that $r_1 \to_i r_2$, we have $r_2 > r_1$), but the real-time clocks at different computers may drift relative to each other (i.e., $R_j - R_i$ is non-constant for $j \neq i$).

    Consider the following modification, *Shamport*, to Lamport's partial-ordering algorithm:

    Rule 1: Before each event at computer i, set $L_i = \min(L_i + 1, R_i)$.

    Rule 2: When sending a message m, apply Rule 1 and include the time L_i as part of the message (i.e. send (m, L_i) instead of just m).

    Rule 3: When receiving a message (m, t) at computer j, set $L_j = \max(L_j, t)$ and then apply Rule 1 before timestamping the message-arrival event.

    Let the Shamport global time of an event e at computer i be $S(e) = L_i(e)$.

    (a) (2 points) In the algorithm above, underline the difference between Shamport and Lamport's algorithm. (Underline as little as possible).

    (b) (8 points) Let e, e' be two events at computer i such that $e \to_i e'$. Prove that $S(e') > S(e)$.

(c) (6 points) Draw a time-series diagram in which $S(\text{send}(m)) > S(\text{receive}(m))$ for some message m. Be sure to compute the Shamport-time and label the values of the real-time clocks at all computers for any events in your diagram. Explain in 1-2 sentences why this unwanted behavior occurs.

(d) (8 points) Suppose we know that the one-way latency of any message between any two computers i, j is at least x clock-ticks by any local clock.

Also suppose we can synchronize R_i and R_j using an external time-source such that $|R_i - R_j| \leq y$, always, for some integer y. Is there a value of y small enough to guarantee that $S(\text{send}(m)) < S(\text{receive}(m))$ for any message m between i and j? If yes, compute the largest possible value of y to guarantee this property, and explain why your answer is correct. If no, explain why.