

Name:

uteid:

1

CS439H: Fall 2011 – Final Exam

Instructions

- Stop writing when “time” is announced at the end of the exam. I will leave the room as soon as I’ve given people a fair chance to bring me the exams. I will not accept exams once my foot crosses the threshold.
- This final is closed book and notes.
- If a question is unclear, write down the point you find ambiguous, make a reasonable interpretation, write down that interpretation, and proceed.
- For full credit, show your work and explain your reasoning and any important assumptions.

Write brief, precise, and legible answers. Rambling brain-dumps are unlikely to be effective. **Think before you start writing** so that you can crisply describe a simple approach rather than muddle your way through a complex description that “works around” each issue as you come to it. **Perhaps jot down an outline** to organize your thoughts. And remember, a **picture can be worth 1000 words.**

- There is no need to fill every nanoacre of paper with writing.
- **Write your name and uteid on every page of this exam.**

1. (25 points)

- (a) **True/False** Doubling the block size in a file system like FFS that uses multi-level indexing will double the maximum file size.

- (b) Write a one-sentence definition of the term *transaction*.

- (c) **True/False** A disk's *average seek time* usually paints too optimistic a picture of a disk's performance for most workloads.

- (d) **True/False** A disk's *mean time to failure* often paints too optimistic a picture of a disk's reliability when the disk is deployed.

- (e) Suppose I have 64-bit virtual addresses, 64-bit physical addresses, and 16KB pages. How many bits from the virtual address are used as the "offset" within a page when computing the physical address?
 - i. 10
 - ii. 12
 - iii. 14
 - iv. 16
 - v. 18

- (f) **True/False** The NFS distributed file system guarantees FIFO/PRAM consistency.

- (g) **True/False** The NFS distributed file system guarantees causal consistency.

- (h) **True/False** It is impossible for a distributed file system that uses callbacks and leases to implement FIFO/PRAM consistency.

- (i) **True/False** It is impossible for a distributed file system that uses callbacks and leases to implement sequential consistency.

Name:

uteid:

3

- (j) Suppose machine A encrypts a file using machine B's public key and sends the encrypted file to machine B. Which of the following guarantees does this strategy provide?
- i. Secrecy (no one but B could read this message)
 - ii. Authentication (no one but A could send this message)
 - iii. Both secrecy and authentication
 - iv. Neither secrecy nor authentication

2. (10 pts)

SOS (Sleazy Operating Systems, Inc.) has a problem. Their OS has a set of queues, each of which is protected by a lock. To do an enqueue or dequeue, a thread must hold the lock associated with the queue.

SOS needs to implement an atomic transfer routine that dequeues an item from one queue and enqueues it on another. The transfer must appear to occur atomically: there should be no interval of time during which an external thread can determine that an item has been removed from one queue but not yet placed on another. An SOS engineer (who didn't take CS 439H!) implemented the following:

```
void transfer (Queue *queue1, Queue *queue2) {
    Item thing; /* thing being transferred */
    queue1->lock.Acquire();
    queue2->lock.Acquire();

    thing = queue1->Dequeue();
    if (thing != NULL) {
        queue2->Enqueue(thing);
    }
    queue2->lock.Release();
    queue1->lock.Release();
}
```

Assume that `queue1` and `queue2` never refer to the same queue. Also assume that you have a function `Queue::Address()` which takes a queue and returns, as an unsigned integer, its address.

(a) What is the concurrency bug with this code?

(b) Implement a better version of the `transfer()` function.

3. (15pts) You have been hired by TxDOT to synchronize traffic over a narrow light-duty bridge on a public highway. Traffic may only cross the bridge in one direction at a time, and if there are ever more than 3 vehicles on the bridge at one time, it will collapse under their weight. In this system, each car is represented by one thread, which executes the procedure `OneVehicle()` when it arrives at the bridge:

```
OneVehicle(Direction direc, Bridge *bridge){
    bridge->Arrive(direc);
    printf("Crossing bridge.\n");
    bridge->Exit(direc);
}
```

In the code above, `direc` gives the direction in which the vehicle will cross the bridge. This is a lightly travelled rural bridge, so the solution does not need to guarantee fairness or freedom from starvation.

`Bridge::Arrive()` must not return until it safe for the car to cross the bridge in the given direction (it must guarantee that there will be no head-on collisions or bridge collapses). `Bridge::Exit()` is called to indicate that the caller has finished crossing the bridge; `Bridge::Exit()` should take steps to let additional cars cross the bridge. One of the engineers on your team proposes this solution:

```
class Bridge{
    enum Direction {EAST=0, WEST=1, ANY};
    Condition direction, number;
    Lock lock;
    int cars;
    Direction currentDir = ANY;
    int64 ticket[2] = {0,0}; // Assume these do not
    int64 turn[2] = {1,1}; // overflow/wrap around

    void Bridge::Arrive(Direction myDir) {
        lock.Acquire();
        int myTicket = ticket[myDir]++;
        while (currentDir != myDir && currentDir != ANY){
            direction.Wait(&lock);
        }
        currentDir = myDir;

        while (cars >= MAX_CARS || turn[myDir] < myTicket){
            number.Wait(&lock);
        }
        cars++;
        turn[myDir]++;
        lock.Release();
    }

    void Bridge::Exit(Direction myDir) {
        lock.Acquire();
        cars--;
        number.Broadcast(&lock);
        if (cars == 0) {
            currentDir = ANY;
            direction.Broadcast(&lock);
        }
        lock.Release();
    }
}
```


4. (15 points)

Suppose a memory cache server workload consists of network clients sending 64-byte requests containing a hash key to a server which reads a 1 KB chunk from a hash table in DRAM and sends 1 KB to the client.

Each network interface has a bandwidth of 1 Gbits/s (that's Gbits not GBytes!) and there is a 400 microsecond one-way network latency between a client and the server. The network interface is full-duplex: it can send and receive at the same time at full bandwidth. The CPU has an overhead of 100 microseconds to send or receive a network packet. Additionally, there is a CPU overhead of .01 microseconds per byte sent, and doing the hash table lookup takes 10 microseconds.

For the back of the envelope type of calculations being asked for here, we are usually happy to be within 10% for these numbers, so feel free to do appropriate rounding and to ignore negligible terms.

- How many requests per second can each network interface satisfy?
- How many requests per second can the server CPU satisfy (assuming the system has a sufficient number of network interfaces?)
- When the server and network loads are low, what is the latency from when a client begins to send the request until it receives and processes the last byte of the reply (ignore any queuing delays).
- Assuming that requests come from many different clients whose activities are uncorrelated and assuming that you must keep average response time below 2 ms, how many requests per second can a system with 4 network interfaces and 4 CPUs handle?

1 Gbit
= 10^9 bits
1 microsecond
= 10^{-6} seconds

1 ms
= 10^{-3} seconds

5. (15)

Consider a cluster file system that stores data in 10MB chunks on chunkservers, that identifies each chunk with a 8-byte chunkID, and that identifies each chunk server with a 2-byte server ID; to read a file, a node first asks a master which chunk server to go to, and then reads from that chunk server. Assuming that the chunk servers have disks with a 10ms avg seek + 10ms average time for 1/2 rotation and 100 MB/s of bandwidth for sequential access, and assuming that the single-processor master can process a lookup request in 5 microseconds of CPU processing time, what is the maximum read bandwidth this system can support (assuming an unlimited number of chunk servers and clients and assuming that the network is not a bottleneck.)

For this file system if 1TB disks cost \$100, each chunk server has 10 1TB disks attached, each chunk server costs \$1,000 plus the cost of its disks, the master server costs \$1,000 plus the cost of its memory, and DRAM costs \$20 per GB, what percentage of total system cost is the master server's DRAM for a 100 chunk-server system (considering only the components listed here and assuming that the master keeps in DRAM all of the information necessary to do its job)?

6. (20 points) A few years ago some people began to “sign” their email by including, at the bottom of an otherwise normal email message, the sender’s name and the date encrypted in the sender’s private key. The message itself is unencrypted, but the signature can be validated by using the `finger` command to retrieve the sender’s public key.

(`finger` is an old utility for getting basic information about a user. If I run the command

```
> finger userID@linux.utexas.edu
```

my local machine will send a message to a well known port on the machine `linux.utexas.edu` where the `finger` daemon will respond with the contents of the file `.finger` from the user `userID`’s home directory. So, users in this problem would just store their public key in their `.finger` file.)¹

Explain why this gives a completely false sense of security, by outlining 5 different ways that you could make it appear that the sender signed mail saying “Prof. Mike is a fink”. The definition of “different” is that each attack has a unique fix. For each of the five attacks you list, give a countermeasure that the sender/receiver could take to protect themselves against just that one attack, where the countermeasure would not help against any of the other attacks you list.²

You may assume that the sender and receiver are on different machines, that both are running on “diskless” workstations whose files are provided by NFS, and that you have the ability to spy on and/or alter packets on any network at the sender or receiver’s site. However, you do not have the power to break into either the sender or receiver’s machine — you can just view/change network packets.

Hint: You may find it helpful to draw a picture showing the flow of network messages in the system.

¹The UTCS department has configured `finger` to only allow such requests from other UGCS department machines. Ignore this limitation. For the purposes of this problem, assume any client can use `finger` to access the file `.finger` for any account.

²You don’t need to provide complete designs; just give the essence of the idea; a couple lines should suffice for each

Name:

uteid:

10

This page intentionally left almost entirely blank