

**CS 574**  
**Final Exam**  
**December 8, 2008**  
**Solutions**

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 Banner ID number, but not your name, on each sheet of paper you turn in

Also, please note the following:

- show your work whenever appropriate. There can be no partial credit unless you show how you derived your answers
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 3:00 to finish the exam. The questions are equally weighted.

1. If I boot my laptop into Windows, I need to either use the laptop's keyboard, mouse, and display to interact with it or else use a separate package like VNC to access it remotely. If I boot it into Linux and run Windows inside a virtual machine environment, I'm able to interact with Windows remotely from a workstation running my usual (X Window System) desktop environment with no extra packages. How does the virtual machine environment's implementation make this possible?

*The virtual machine environment provides a virtualized keyboard, mouse, and frame buffer. Since the standard windowing environment under Linux is X, these are implemented as an X window (and event handlers). As X provides interaction over the network, so does the VM environment.*

*In general, student answers to this question never addressed the question at all. There were quite a few descriptions of virtualization, and many detours into Xen's paravirtualization. I wound up giving five points for guesses that a virtualized NIC might have something to do with it. A few answers headed off into descriptions of page tables...*

2. Consider the following fragment of C code:

```
char a[1024], b[1024], c[1024];
for (i = 0; i < 1024; i++)
    a[i] = b[i] + c[i];
```

Suppose your computer has an 8K, direct-mapped physically-addressed cache with an eight-byte block size, and a 1K page size.

- (a) How can an unlucky page placement result in a cache hit rate in the arrays of essentially 0?

*Notice that each array fits exactly on one page. If the pages are placed so they are 8K apart, corresponding entries of the arrays map into the same cache block. So, when we read  $b[i]$  we get a cache miss. Now we read  $c[i]$ , get another cache miss, and evict  $b[i]$ . When we write  $a[i]$  we get another cache miss, and evict  $c[i]$ . When we go on to  $i+1$ , we repeat. A few people came up with answers that had a page fault on every access. Well, it meets the requirements so I'll take it...*

- (b) How can a good page placement improve this, and what is the best possible cache hit rate in the arrays? *I'll accept an answer that correctly sets up the equation for the approximate hit rate; you don't need to do the arithmetic*  
*Any page placement that doesn't put the arrays at multiples of 8K from each other will work. In this case, the arrays don't conflict with each other in cache. Some students waved their hands and talked about page coloring here; I'll give five points for that.*  
*There are eight array elements on each cache block. When we enter a new cache block, we have three cache misses (one each for  $a[i]$ ,  $b[i]$ , and  $c[i]$ ). For the next seven array indexes (21 memory accesses), we get cache hits. So, the hit rate is  $21/24=7/8=.875$ . Note: a hit rate is a number between 0 and 1 inclusive (alternatively, a percentage). Answers that are outside that range can't possibly be correct.*

*We'll break the question up into ten points for (a), ten points for the layout in (b), and 5 points for the hit rate calculation in (b).*

3. An extremely common error in HW 2 was, on timeout, to send a special packet with a key of RESEND to request a resend from the server (with the server responding to this key by resending the previous response). What would be the result if the original request had been dropped on the way to the server? *This question assumes RESEND was spelled correctly in both the client and server code. The fact that it wasn't in most cases is a separate bug, which I'm not asking about*  
*It would end up resending the response to whatever the previous request (if any) was.*  
*Some students are answering with the assumption that the server is correct (so the client sends a RESEND, but the server isn't written to interpret this as a request to resend). In this case, of course, it simply treats it as a request with a key of RESEND. I'll accept this. Another common way of answering the question seems to be to assume the RESEND is what gets lost. No, that's not a reasonable way to read it.*
4. In both the DASH and LimitLESS protocols, if a node flushes a read-only memory block from its cache, no message is sent to the block's home node to inform it of the fact.

- (a) It turns out that this does not affect the correctness of the memory consistency protocol. Why not?  
*Since the block was in the read-only state, there is a correct copy in the home node's memory (and, of course, the copies in any other nodes are correct as well). So they can use their copies, and other nodes can request copies, and there's no error. The only question is what happens if somebody requests it for read-write – we'll handle that in part (b).*
- (b) Of course, this implies that it is possible, under LimitLESS, for a memory block's home node to send an INV message to a node that does not in fact have a copy of the memory block. How should the node respond to the INV message?  
*Of course, this can only happen if a node wants the block in the read-write state, and INV messages are sent to all the nodes that the home node thinks have copies. The node can just reply with an ACKC, acknowledging that it no longer has a copy (it didn't have one before getting the INV either, but that isn't really what the home node is concerned about!). Note that you can't use REPM or UPDATE – those also send the data, and of course we don't have that since we flushed that cache entry. I'm treating this as 15 points for arguing correctness, and 10 for the ACKC.*  
*Notice that anyone who carefully read my solution to HW3 would get this part of the question!*

Banner ID: \_\_\_\_\_

5. The major difficulty in migrating a process from one host to another is moving its virtual memory. How would using a distributed shared memory system simplify this problem?  
*Having a consistent address space across all processors (the main point of DSM, after all) makes the problem go away completely. You don't have to explicitly send the data at all; it would just migrate as it's accessed, looking a lot like the schemes we saw that make use of the virtual memory to transparently move the data. Several students are seeing this in terms of flushing the memory to a backing store, and then paging in from that backing store. Hmmm, yes, that's a reasonable way to read the question. Not what I had in mind, but that's not your fault.*  
 Would it actually reduce the time required to transfer the memory?  
*No. The data still has to be sent. It's actually likely to make things worse, since any access to data that hasn't migrated yet will cause a flurry of messages to get a cache line transferred, while a protocol designed to transfer the data in bulk is likely to be much more efficient. But... in the special case I mentioned before, of combining a DSM system with flushing to backing store, it's not nearly so bad.*
6. You wish to send the message 1111010, appending a CRC calculated using a CRC polynomial of  $X^3 + X^2 + 1$ . Calculate the CRC for the message. *Note: assume the modern convention of having the left-most bit of the message as the most significant bit – backwards from the paper*  
*In binary, the polynomial is 1101. We append three 0s to the message and divide:*

```

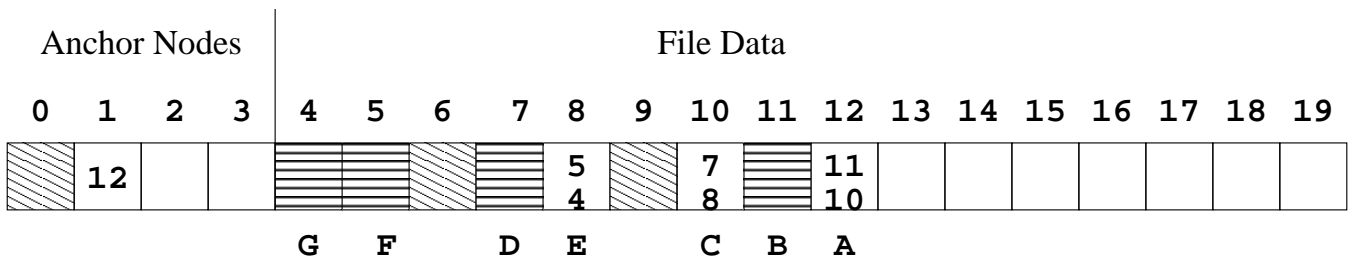
1101) 1111010000
      1101
      0010010000
        1101
        01000000
          1101
          101000
            1101
            11100
              1101
              110

```

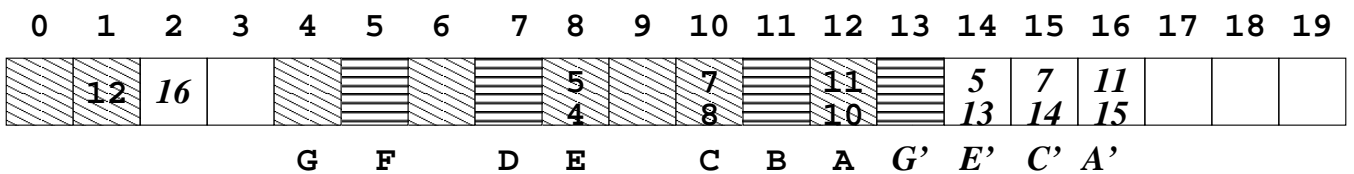
Points	Error
-5	didn't append 0s
-5	CRC polynomial backwards
-10	Stop early (several steps)
-5	Take an extra step

7. Consider the file transformation shown in the figure in Section 3.2 of the Engel and Mertens LogFS paper.

Suppose the file is laid out as shown in the following figure. Modify the figure to show the effects of applying the transformation. You may answer the question on this sheet, and turn it in. The problem does **not** require you to perform garbage collection



- Free block
- Obsolete block
- Anchor block containing pointer
- Data block containing two pointers
- Data block containing leaf node



Points	Error
-5	For "reasonably close" answers, 5 points per missed node
5	Far, far off but in terms of physical layout
0	Answer in terms of abstract graph

8.