University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Spring 2006 Anthony D. Joseph

Midterm #2 Exam *Solutions*

April 26, 2006
CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA Name:	
Discussion Section Time:	

General Information:
This is a **closed book and notes** examination. You have 90 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

Problem	Possible	Score
1	19	
2	36	
3	23	
4	22	
Total	**100**	

1. (19 points total) Short answer questions:
   a. (11 points) Translation Look-Aside Buffers:
      i) (6 points) On a TLB miss in Nachos, what data structures should you check to find a previously referenced page? *List the data structures in the order that they should be checked.*

         First, you should check the Inverted Page Table, then the swap file, and finally the application COFF file. Each correct answer was worth two points. We subtracted one point for each incorrect answer or wrong order.

      ii) (3 points) Give a two to three sentence description of "precise exceptions".
         When a precise exception occurs, the Program Counter cleanly divides the fully executed and non-executed instructions – there are no partially executed instructions.

      iii) (2 points) Briefly (1-2 sentences) say why precise exceptions are nice to have.
         With precise exceptions, there are no side effects or instructions for the exception handler to undo or rollback, making the OS simpler.

b. (4 points) Networking:

i) (2 points) The current version of IP uses 32-bit addresses. List one reason that we cannot use all 2^{32} addresses.
Two reasons: the hierarchical name allocation yields internal fragmentation of network address space, and some of the address are reserved for private networks or other purposes (e.g., x.y.z.0, 127.0.0.1, or localhost).

ii) (2 points) List one reason why it would still be possible to connect more than 2^{32} computers to the Internet.
Two reasons: Network Address Translation allows multiple computers to be connected using a single IP address, and IP addresses can be time-shared using the Dynamic Host Configuration Protocol.

c. (4 points) Caching:

i) (2 points) What are the two types of locality exploited by caching?
Spatial locality and temporal locality.

ii) (2 points) Which kind of locality makes large cache lines a good idea, and why?
Spatial locality makes large cache lines a good idea because it relies on memory references being to regions near each other.

2. (36 points) File Systems and Disks.

a. (8 points) Consider a file system with 2048 byte blocks and 32-bit disk and file block pointers. Each file has 12 direct pointers, a singly-indirect pointer, a doubly-indirect pointer, and a triply-indirect pointer.
Express your answers in symbolic form for partial credit.

i) (4 points) How large of a disk can this file system support?
2^{32} blocks x 2^{11} bytes/block = 2^{43} = 8 Terabytes.

ii) (4 points) What is the maximum file size?
There are 512 pointers per block, so: 2048 x (12 + 512 + 512^2 + 512^3) = 2^{38} + 2^{29} + 2^{20} + 24k = 256G + 513 M + 24 K. We gave subtracted 1 point each if you did not include the 2K block size, the powers of 512 for indirect pointers, or the overall multiplication

b. (12 points) Disk requests come into the disk driver for cylinders: 10, 22, 20, 2, 40, 6, and 38, in that order. The disk head is currently positioned over cylinder 20. A seek takes 6 milliseconds per cylinder moved. What is the sequence of reads and total seek time using each of the following algorithms?

i) (4 points) First-come, first-served:
10, 22, 20, 2, 40, 6, 38
10 + 12 + 2 + 18 + 38 + 34 +32 = 146 cylinders = 876 milliseconds.
We subtracted 1 point for missing the first reference (to 20), and 1 point for not writing out the sequence of accesses.

ii) (4 points) Shortest Seek Time First:
20, 22, 10, 6, 2, 38, 40
0 + 2 + 12 + 4 + 4 + 36 + 2 = 60 cylinders = 360 milliseconds.

iii) (4 points) SCAN (initially moving upwards):
20, 22, 38, 40, 10, 6, 2
0 + 2 + 16 + 2 + 30 + 4 + 4 = 58 cylinders = 348 milliseconds.

c. (2 points) Briefly (2-3 sentences) state the difference between a hard link and a soft link.
Hard links point to the same inode, while soft links simply list a directory entry. Hard links use reference counting. Soft links do not and may have problems with dangling references if the referenced file is moved or deleted. Soft links can span file systems, while hard links are limited to the same file system.

d. (7 points) Read-ahead.
i) (3 points) Give a brief (2-3 sentences) description of read-ahead.
Prefetching subsequent blocks on the same cylinder when a request is received. We subtracted 1-2 points if your answer didn't clearly state that subsequent blocks were read.

ii) (2 points) Briefly (2-3 sentences) state why read-ahead is useful.
An application with spatial locality will read the prefetched data (1 pt), avoiding rotational delay and reducing the number of I/O operations (1 pt). We subtracted 1 point if you didn't explicitly state that there are fewer I/O operations.

iii) (2 points) Briefly (2-3 sentences) state what happens if you read ahead too much.
Other normal reads can be delayed by prefetching (1 pt), and prefetched data may not be used (1 pt). Also, prefetching replaces blocks in the buffer cache that may be being used, and some files are not sequential, so they won't benefit from prefetching.

e. (7 points) RAID

i) (3 points) Give a brief (2-3 sentences) description of RAID 5.
Redundant Array of Inexpensive Disks level 5 stripes (1 point) blocks of data across at least three drives (1 point) along with an interleaved XOR-based parity block for each stripe set (1 point).

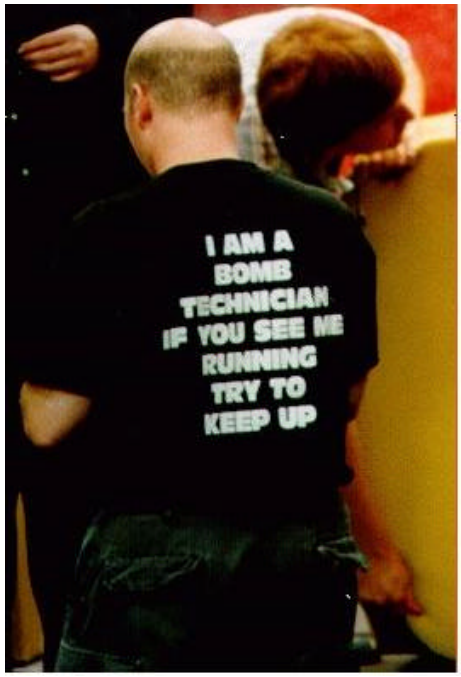
ii) (2 points) How many disk failures can RAID 5 tolerate without losing data?
One.

iii) (2 points) How would you reconstruct a failed disk?
Read blocks from the other drives (1 point) and use an XOR (1 point) of their blocks to reconstruct the data.

No Credit – **Problem X** (000000000000 points)

The importance of synchronization… or You're first – after me (This Only Happens in New York City).

Race Condition

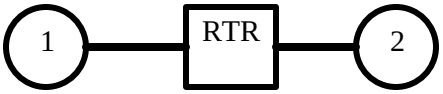


3. (23 points total) Reliable Peer-to-Peer (P2P) File Sharing.

Two computers running a P2P application using TCP/IP for connections are connected via a router (see below). To write data packets to the network, the P2P application issues a system call, and the OS copies the data first to a kernel buffer and then uses DMA to copy the data to the network controller board. Immediately after receiving the data, the controller generates one or more packets and sends them.

The packets travel one hop to router RTR, which will in turn forward the packets to their final destination. *Assume that no packets will be lost, and assume that the processing time at the router is negligible, and a router must wait for the entire packet to arrive before forwarding it.*

The receiving network controller uses DMA to copy the data it receives to memory, and when the last bit arrives and is transferred, it interrupts the CPU. The OS copies the data to the P2P application's buffer in user space. After the copy completes, the OS sends back a one-byte acknowledgement to the sending OS. *For simplicity, you may treat acknowledgements as if they are zero bytes long. Assume that a sender uses a send window-based acknowledgement protocol.*



System Parameters:

- Time to process an interrupt, T_{int} = 1.5 millisecond
- CPU cycle time, T_{CPU} = 10 nanoseconds
- Packet size, N_{pkt} = 1,000 bytes (ignore headers)
- Time to copy or DMA one byte, T_{copy} = 1 microsecond
- Latency of a link, T_{link} = 3 milliseconds
- Bandwidth of links, B = 8,000,000 bit/sec
- Send window N_{win} = 1 packet
- Retransmission timeout, T_{retran} = 400 milliseconds

a. (7 points) Compute T_{pkt}, the time to reliably send a single packet from a P2P application on computer 2 to a P2P application on computer 1 over a link with no losses.

$T_{pkt} = (4 \times N_{pkt} \times T_{copy}) + 2 \times T_{int} + 4 \times T_{link} + (8 \times N_{pkt}) / B$
Each packet is copied 4 times (user-level buffer to kernel buffer, kernel to network card, network card to kernel, and kernel to user-level), which takes 4 ms ($4 \times N_{pkt} \times T_{copy}$). There are 2 interrupts ($2 \times T_{int}$) for receiving the packet/acknowledgement. Finally, the two-way transmission time is 14 milliseconds (6 ms between RTR and the sender, 1 ms each to receive the packet at RTR and other host, and 6 ms between RTR and the other host) for a total of 21 milliseconds per 1,000 bytes. We subtracted 1 pt each for math errors and each mistake or extra term.

b. (3 points) What is the maximum or peak rate at which one process can reliably send data to another process?

T_{pkt} = 21 milliseconds per 1,000 bytes. The maximum rate is N/T_{pkt} = 47,619 byte/sec or 476,190 bit/sec.

c. (4 points) In terms of T_{pkt}, how long does it take to reliably send an .mp3 file from computer 2 to computer 1 if the file is 4,000,000 bytes in size (assuming the links have no losses)?

4 million bytes equals 4,000 packets. So, T = (4,000 x T_{pkt}). At 21 milliseconds per packet it will take 84 seconds.

d. (5 points) If we use a send window of 40 outstanding packets instead, what is the maximum rate at which one process can reliably send data to another process (assuming the links have no losses)?

If we can have outstanding acks, then the sending rate is gated by the copying overhead at the sender from user to kernel and then kernel to network card (the receiver's overhead is symmetric) = (2 x N_{pkt} x T_{copy}) or 2 milliseconds per 1,000 bytes sent. This yields a peak data rate of 500,000 bytes/sec or 4,000,000 bit/sec. Now we need to compare the bandwidth delay product using this rate to the send window, since it could be a potential bottleneck. Using T_{pkt} as the baseline, the bandwidth delay product is 21 milliseconds x 4,000,000 bit/sec or 84,000 bits. The sending window is 40 packets x 8 bits x 1,000 characters or 320,000 bits. So, the limiting factor is the copying overhead, and not the sending window. The correct answer is 4,000,000 bit/sec.

e. (4 points) If we use a send window of 40 outstanding packets instead, how long does it take to reliably send an .mp3 file from computer 2 to computer 1 if the file is 4,000,000 bytes in size (assuming the links have no losses)?

4 million bytes equals 4,000 packets, and at 2 milliseconds per packet it will take 8 seconds. We gave partial credit for answers that were consistent with part d, and subtract 1 pt each for minor errors or using T_{pkt} from part a. We did not give credit for answers that were the same as part c.

4. (22 points total) Read-only Page Sharing.

To more efficiently use the physical memory allocated to processes, modern operating systems share read-only pages of different running instances of the same application (e.g., multiple Emacs processes). Your task is to implement this functionality for Nachos – sharing of read-only COFF pages in Nachos memory. Since we don't have actual inodes in Nachos, it's hard to have a real unique file handle, so it is sufficient if your answer provides a file name.

a. (6 points) List the data structures that you would need to add to the kernel:
We generally looked for data structures that mapped executable name to physical memory page number (ppn). Example: open_execs: exec_name->Coff_Info object. Each Coff_Info has file handle/process name, table mapping COFF pages to ppn.
If the data structures were only partially sufficient, we subtracted 3 points.

b. (4 points) Name all the place where would you have to update these data structures:
When you page in/out COFF pages (3 pts) and when you load/unload an application (1 pt).

c. (12 points) Design the function `load_coff_page()` that takes an executable name and virtual page number, and returns a physical page number if that COFF page is already in memory, or -1 if it must be loaded from disk.

```
int load_coff_page( String exec_name, int vpn ) {
        coff_info = open_execs.get( exec_name );
        if (coff_info == NULL)
           return -1;
        else
           return coff_info[vpn];
}
```

We subtracted 6 pts if the answer missed a main point, 2 pts if the answer was inefficient, and 8 pts if the answer had major flaws.

This page intentionally left blank