Stanford University
Computer Science Department
CS 240 Quiz 2
Winter 2002

This is an open-book exam. You have 50 minutes to answer ten out of twelve questions. Write all of your answers directly on the paper. Make your answers as concise as possible. Sentence fragments ok.

NOTE: We will take off points if a correct answer also includes incorrect or irrelevant information. (I.e., don't put in everything you know in hopes of saying the correct buzzword.)

Question	Score
1 - 3	
4 - 6	
7 - 9	
10 - 12	
total	

Stanford University Honor Code

In accordance with both the letter and the spirit of the Honor Code, I did not cheat on this exam nor will I assist someone else cheating.

Name and Stanford ID:

Signature:

Answer **10 of the following 12 questions** (i.e., skip two) and, in a sentence or two, say *why* your answer holds. (5 points each).

1. Livelock: in Figure 6-3 why does "Polling (no quota)" work badly?

   Process packets as long as they come in; the point of polling was for fairness.

   To get full credit you needed to mention how polling with no quota specifically causes livelock (not just, "packets getting dropped at output queue").

2. The livelock paper has a hack to handle the problem of dropping packets on the "screend" queue. What limit does this solution have for multiple applications getting network data?

   Suspends receive processing; if the packet was not for screend you are not happy.

   Two points were given for correctly describing the hack. The rest of the points were dependent on the description of the hack's behavior in a multi-process system.

3. If we use Ethernet point-to-point network, where there is a single sender and a single receiver on each cable, which mechanisms can we eliminate?

   For full credit you needed to specify at least three of the following points. Two points were deducted for each one less than three. One point was deducted for each incorrect one.

   Don't need: listen before transmit, listen for collision, retransmission, minimum or maximum packet size, addresses, exponential backoff, collision consensus enforcement.

4. What Ethernet design decision(s) seem to implicitly rely on the end-to-end argument?

   Ethernet is best effort rather than perfect, which is the classic end-to-end network contract.

5. Give two examples where Google takes a New Jersey design approach.

   Ignoring offsets above 4K, not considering huge font switches.

6. When cleaning in LFS how do you tell if an inode is live? If a data block is live?

   Inode: live if the inode map points to it. Data block: live if it is contained in a file.

7. A CS240 classmate claims that XFS write performance will always lag behind LFS, because XFS writes metadata to a log and also to another data structure on disk, whereas LFS only writes data to a log. Explain one fault in your classmate's argument.

   The cost of writes in LFS should include the cost of the the segment cleaner, which can end up being quite expensive, for example when the disk is near full. Also, XFS uses asynchronous logging of metadata only, so it can be very cheap since the metadata updates can be grouped together and written to the on-disk log quickly. The answer "XFS has asynchronous logging" without sufficient explanation of why this helps performance got 3 points. Also accepted were answers saying how XFS is designed to exploit parallelism, such as by allocating blocks in parallel or through using multiple allocation group threads. Answers that only mentioned that LFS also logs metadata were worth 0, since this doesn't invalidate the argument given.

8. Explain two reasons why delayed allocation improves performance for XFS.

   1) there is more information about the size of a file, so it's easier to allocate an appropriately sized extent for the file. 2) temporary files may be deleted before being written to disk, saving writes. 3) files that are randomly written can still be allocated contiguously if the writes happen to contiguous regions of the file.

9. Give two NFS operations that are not stateless. How could you build an NFS implementation to handle this? (Be concrete: don't just give a couple of buzzwords.)

   create, delete, etc. Also, file permissions may change onb open files but they should remain available. Also, according to unix semantics, files opened and subsequently deleted should remain available for reading. Clients can tag each request with a unique integer; server can track the last integers used and not do the operation if they match. Also accepted were fixes that mentioned a replay cache. If no fix was mentioned, the answer got 3 points.

10. In what way does synchronously writing an NFS operation to disk before replying to the sender violate the end-to-end argument?

    Three possible reasons: (1) not all clients need this functionality, so should be able to turn it off; (2) on a local system could still crash and lose data after file system operation returns to application, so having this extra guarentee is pointless in some *situations. (3) when the NFS server replies to the client it does so without ensuring that the data was written to disk successfully.*

Five points where given if you mentioned one of the three reasons. Three points where given if you had an answer that resembled one of the solutions but weren't specific enough.

11. Both EFS and LFS have a cleaner. How do they differ?

    EFS's cleaner deletes keep landmark files; does not compact dead space. LFS reads in data, frees, compacts and writes out. Much more expensive.

    4 points where different if you mentioned the compaction difference. 1 point was given for mentioning that the LFS cleaner has more overhead (or runs slower).

    Partial credit: 2 points: Understood how only one of the cleaners worked 2 points: Only mentioned the heuristic difference between the cleaners

12. You ditch cs240 to hack on your multi-file networking assignment. Since your work is so valuable, you use the EFS file system and mark all files as "keep landmark." What problem would multiple files create for the naive keep-landmark policy? How does EFS allow you to counter this?

    You will get inconsistencies if you keep different versions of a related group of files. EFS allows you to explicitly relate files.

    3 points where given for talking about the inconsitencies 2 points where given for giving the EFS solution.