Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2016

Quiz I

There are **15 questions** and **13 pages** in this quiz booklet. Answer each question according to the instructions given. You have two hours to answer the questions.

- The questions are organized (roughly) by topic. They are not ordered by difficulty nor by the number of points they are worth.
- **If you find a question ambiguous, write down any assumptions you make.** Be neat and legible.
- Some students will be taking a make-up exam at a later date. **Do not** discuss this quiz with anyone who has not already taken it.
- You may use the back of the pages for scratch work, but **do not** write anything on the back that you want graded. We will not look at the backs of the pages.
- Write your name in the space below. Write your initials at the bottom of each page.

This is an open-book, open-notes, open-laptop quiz, but you may **NOT** use your laptop, or any other device, for communication with any other entity (person or machine).

Turn all network devices, including your phone, off.

CIRCLE your recitation section:

10:00	1. Matei/Cathie	7. Mark/CK	8. Michael/Pratheek	15. Karen/Jacqui
11:00	9. Matei/Cathie	10. Mark/CK	11. Michael/Pratheek	16. Karen/Jacqui
12:00	12. Asaf/Anubhav			
1:00	2. Mike/Steven	5. Peter/Sumit	13. Mark/Jodie	14. Asaf/Anubhav
2:00	3. Mark/Jodie	4. Peter/Sumit	6. Mike/Steven	

Name:

I Virtual Memory

1. [9 points]: Operating system X uses a standard page-table scheme, where the first N bits ($1 \leq N < 32$) of a 32-bit address are interpreted as the virtual page number, and the last $32 - N$ bits are interpreted as the offset into the page.

Operating system Y uses a hierarchical page-table scheme. The first M bits ($M \geq 1$) are interpreted as the outer page number, the next P bits ($P \geq 1$) are interpreted as the inner page number, and the last $32 - M - P$ bits are interpreted as the offset into the inner page table ($M + P < 32$).

Answer the following questions for **each** operating system.

A. What is the maximum number of virtual addresses available to any process?

In OS X:

In OS Y:

B. A process attempts to read a value in virtual address v. The process has read-access to this data (i.e., it has permission to read the data stored in v). What is the maximum number of page faults that could be thrown as a result of this access?

In OS X:

In OS Y:

C. Assume that each page-table entry (in any of the tables) is K bits. What is the maximum amount of memory that will be allocated **for page tables** after a single access?

In OS X:

In OS Y:

Initials:

2. [6 points]: Modern hardware has built-in support for virtualization (i.e., it has functionality that can support a VMM that manages multiple guest OSes). Which of the following is true about such hardware?

A. True / False There is a special VMM operating mode in addition to user and kernel operating modes.

B. True / False Such hardware supports both monolithic and microkernels.

C. True / False Guest OSes' page tables map directly from guest virtual addresses to host physical addresses.

II Threads and Processes

3. [9 points]: Ben Bitdiddle is attempting to implement bounded buffer `send` and `receive`. Below is the pseudocode of his current implementation:

```
// bb is a structure containing:
//    buf - the buffer itself
//    in - the number of messages that have been placed in buf
//    out - the number of messages that have been read from buf
//    lock - a lock which must be held to read from or write to buf

// N is a global variable specifying the capacity of the buffer
```

```
send(bb, message):
    acquire(bb.lock)

    while (bb.in - bb.out) == N:
        pass

    bb.buf[bb.in % N] = message
    bb.in += 1

    release(bb.lock)
    return
```

```
receive(bb):
    acquire(bb.lock)

    while (bb.in - bb.out) == 0:
        pass

    m = bb.buf[bb.out % N]
    bb.out += 1

    release(bb.lock)
    return m
```

A. Describe a scenario that could lead to deadlock in Ben's code, specifying the **precise sequence** of events that would lead to deadlock.

Write **no more than four sentences** (or bullet-points). If you write more than four sentences (or bullet-points), we reserve the right to only read the first four. We are interested in the clarity of your explanation as much as its content.

B. Ben claims that adding a single call to `yield()`—and no other code—in each of the two methods will fix his deadlock issue. Is he correct? Explain (again, in no more than four sentences).

C. **True / False** All correct multi-threaded code—including Ben's, once it is correct—must call either `yield()` or `wait()`; otherwise, one thread may completely halt the process of other threads.

4. [4 points]: Consider the following commands executed, in order, from a single UNIX shell, as described in "The UNIX Time-Sharing System" by Ritchie and Thompson. `$` is the command prompt.

```
$ command1
$ command2 &
$ command3
```

The shell is ready to accept the next command—`command4`—from the user. Answer the following true/false questions.

(a) **True / False** Since the shell is ready for `command4`, all previous commands (1–3) have completed.

(b) **True / False** At the start of its execution, `command1` has zero open file descriptors.

5. [6 points]: You run Eraser—as described in "Eraser: A Dynamic Data Race Detector for Multithreaded Programs" by Savage et al.—on a program that you wrote. It issues no warnings. Answer the following true/false questions about the program.

(a) **True / False** At the end of execution, the lockset of each variable was nonempty, but may have been empty at some point during execution.

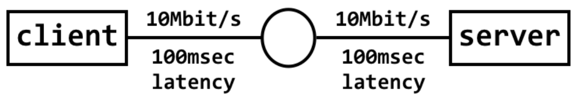
(b) **True / False** Because Eraser issued no warnings, it must be the case that no threads competed for access to a single variable in your program.

(c) **True / False** Because Eraser issued no warnings, deadlock will not occur in your program.

(d) **True / False** Because Eraser issued no warnings, data races will not occur in your program.

III Performance

6. [8 points]: Consider a client and a server connected via a single switch. Each link in the network has a capacity of 10 Mbit/sec, and a one-way latency of 100 msec. The processing time at the switch is negligible. The **maximum** packet size in the network is 1000 bytes.



The server hosts a table that contains <`key, value`> pairs. The client sends requests to the server that contain a `key`; the server responds with the corresponding `value`. Each request packet has a 100-byte header and 100 bytes of data (the key). Responses have the same format (100-byte header with 100 bytes of data).

You have three possible techniques for improving the system:

(a) **Parallelism:** Add an additional network link, with 10 Mbit/s throughput and 100 msec latency, between the client and the switch. Have the client send requests out on both links (in parallel) instead of a single link.

(b) **Caching:** Cache server responses at the switch. If the switch has a cached <`key, value`> pair, it will return the value to the client immediately instead of forwarding the request to the server. Assume that the time it takes for the switch to look up a key in its cache is negligible.

(c) **Batching:** Have the client batch requests before sending them over the network: each packet to the server would include multiple keys, instead of one key per packet. Assume that the server can parse and process multi-key packets.

A. Of the three techniques, which (if any) will most significantly improve latency?

(a) Parallelism
(b) Caching
(c) Batching
(d) None of the techniques would significantly improve latency

B. Of the three techniques, which (if any) will most significantly improve throughput (number of requests per second)? Assume that the client is generating enough requests to take advantage of any improvements in throughput.

(a) Parallelism
(b) Caching
(c) Batching
(d) None of the techniques would significantly improve throughput

Initials:

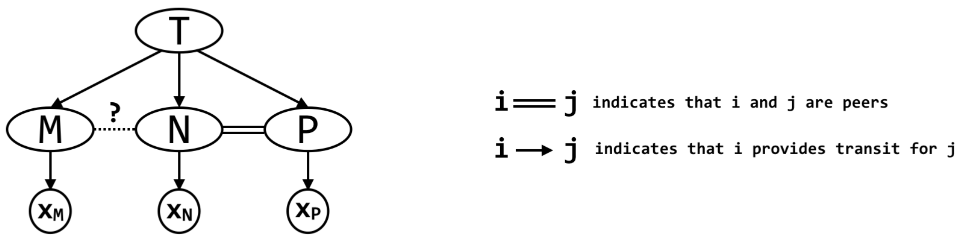
7. [6 points]: Answer the following true/false questions about MapReduce, as described in "MapReduce: Simplified Data Processing on Large Clusters" by Dean and Ghemawat.

A. True / False Intermediate key-value pairs from `map` tasks must be written to GFS to protect against worker failures.

B. True / False MapReduce takes data locality into account when scheduling `map` tasks, but not when scheduling `reduce` tasks.

C. True / False MapReduce starts backup tasks for all tasks in order to quickly deal with stragglers.

IV Routing, Addressing, and Naming



8. [9 points]: Consider the above AS topology. These ASes advertise routes according to the standard BGP rules discussed in class. Assume that there are no failures and no loss in the network and that all routes have converged.

All of the inter-AS relationships are known to you, except the relationship between M and N. In an attempt to infer that relationship, you observe some of the traffic flowing in the network.

For each of the following, indicate which relationships between M and N are consistent with the traffic that you observed. Consider each scenario independently.

A. Traffic between x_M and x_P traverses the following ASes: $x_M \to M \to T \to P \to x_P$.

(a) **Consistent / Not Consistent** M and N are in a peering relationship.
(b) **Consistent / Not Consistent** M provides transit for N.
(c) **Consistent / Not Consistent** N provides transit for M.

B. Traffic between x_M and x_P traverses the following ASes: $x_M \to M \to N \to P \to x_P$.

(a) **Consistent / Not Consistent** M and N are in a peering relationship.
(b) **Consistent / Not Consistent** M provides transit for N.
(c) **Consistent / Not Consistent** N provides transit for M.

C. Traffic between x_M and x_N traverses the following ASes: $x_M \to M \to N \to x_N$.

(a) **Consistent / Not Consistent** M and N are in a peering relationship.
(b) **Consistent / Not Consistent** M provides transit for N.
(c) **Consistent / Not Consistent** N provides transit for M.

Initials:

9. [4 points]: Answer the following true/false questions about RON, as described in "Resilient Overlay Networks" by Andersen et al.

A. True / False To improve scalability, each RON node only probes other RON nodes that are geographically close.

B. True / False Adding more nodes into a RON will cause more overhead but may expose more possible paths in the network.

10. [6 points]: Two DNS clients, A and B, make a DNS request to resolve `google.com` within milliseconds of each other. However, A and B receive different responses. For each of the following scenarios, circle "True" if that scenario could have caused the change.

(a) **True / False** The two requests were resolved by the same nameserver, but the mapping changed in between the two requests being processed.

(b) **True / False** `google.com` is using a CDN such as Akamai, which changes the value of the mapping depending on (among other things) the geographic location of the clients.

Initials:

V Transport

11. [9 points]: A TCP sender S sends a window of 9 packets—`101` through `109`—with .1 msec in between each packet (i.e., packet `101` goes out at time t, packet `102` at time $t + .1$, etc.). Packets `104`, `105`, and `106` are lost; all other packets arrive at the receiver successfully. The round-trip time between S and the receiver is 1 second, and S's TCP timeout is set to 2 seconds.

Assume that there is no additional loss on the network. I.e., no ACKs from the receiver are lost, and no retransmissions from the sender are lost.

A. Which packets will be retransmitted due to TCP's fast-retransmit mechanism?

(a) `104`
(b) `105`
(c) `106`
(d) None of the above.

B. Which packets will be retransmitted due to a TCP timeout?

(a) `104`
(b) `105`
(c) `106`
(d) None of the above.

C. Assume that S currently has a window size of 9. Let W_f be the size of S's window immediately after a retransmission due to fast-retransmit, and let W_s be the size of S's window immediately after a retransmission due to a timeout. Which of the following is true?

(a) $W_f > W_s$
(b) $W_f < W_s$
(c) $W_f = W_s$
(d) There is not enough information to decide.

Initials:

12. [6 points]: Consider the following four queue management schemes: DropTail, ECN, RED, and DCTCP. In each of the following questions, circle the name(s) of the queue management scheme(s) for which the statement is **true** (if it's not true for any of the schemes, circle "None").

(Note: we use "RED" to refer to the scheme that drops packets rather than marks packets. This is the convention that was used in lecture, but not the convention used in the DCTCP paper.)

A. Requires a measurement of the average queue size.

DropTail ECN RED DCTCP None

B. Can react to congestion before queues are full.

DropTail ECN RED DCTCP None

C. Requires **no** changes to a TCP sender's behavior.

DropTail ECN RED DCTCP None

Initials:

13. [8 points]: Two types of flows are sending data through a single switch S to a single destination:

1. Small, latency-sensitive flows, which send at most 20 packets per second total (i.e., all latency-sensitive flows combined send no more than 20 packets per second in aggregate).
2. Large, throughput-sensitive flows. There is no limit to the amount of data that the large flows may send.

Currently, both flows send their data through a single queue in S. This queue is the bottleneck in the network; capacity is not an issue after the packets leave S. Packets will also not be lost or corrupted after they leave S. There are no failures in the network (e.g., switches dying or links going down).

The processing time for each packet in the queue is 1 msec, and the one-way latency to the destination is 100 msec. This means that, if there are x packets already in the queue, it will take 101+x msec for a packet to arrive at its final destination. All packets in the network are the same size.

For each of the following queueing schemes, circle "True" if it allows you to guarantee that every **latency-sensitive** packet arrives at its destination within 150 msec. Assume that all of the proposed queues can hold up to 1000 packets.

(a) **True / False** Use a single queue. When a latency-sensitive packet arrives, if there is room in the queue, place the packet at the **head** of the queue, instead of the tail.

(b) **True / False** Use two queues: Q_1 for latency-sensitive traffic, and Q_2 for throughput-sensitive traffic. Service the queues in a round-robin fashion, sending a single packet from each queue at a time. If there are no packets in a queue at its service time, simply move to the next queue.

(c) **True / False** Use two queues: Q_1 for latency-sensitive traffic, and Q_2 for throughput-sensitive traffic. Service the queues in a round-robin fashion, but send **all** packets from Q_1 when servicing it, and only a single packet from Q_2 when servicing it. If there are no packets in a queue at its service time, simply move to the next queue.

VI Newer Internet Technologies

14. [4 points]: Akamai is considering a new hybrid technology. When a client makes a request for a piece of content, they are directed to an edge server as usual. This edge server then points them to a BitTorrent swarm (e.g., by providing the client with a `.torrent` file). The client downloads half of the content from the edge server, and simultaneously downloads the other half of the content from the BitTorrent swarm. Akamai believes that the selling point of this technology is that it improves client performance.

Write **one sentence** describing a scenario where this technology **does not** improve client performance.

15. [6 points]: Consider two clients, A and B, attempting to send data to an access point, X. A and B can hear X perfectly, and X can hear A and B perfectly. However, A and B cannot hear each other.

Which of the following will prevent A and B's **data** packets from colliding? Circle all that apply. Ignore collisions from control-traffic—such as RTS/CTS messages—when applicable.

(a) Use CSMA/CA as the MAC protocol, but turn off carrier sense: the mechanism by which A and B listen to the channel before sending.

(b) Use CSMA/CA with the RTS/CTS mechanism. A "CTS" message for a client clears that client for sending in the next timeslot. Clients will **not** send in the next timeslot if they overhear a CTS message meant for someone else.

(c) Use the "round-robin" MAC protocol—A sends every other timeslot, B sends on the opposite timeslots—with X dictating the timeslot length, start of the timeslots, etc. (This "round-robin" MAC protocol is also known as TDMA.)

(d) None of the above.

End of Quiz I

Please double check that you wrote your name on the front of the quiz, circled your recitation section, and initialed each page.

Initials: