Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2016

Quiz 2

There are **21 questions** and **?? 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: SOLUTIONS

I Reliability

1. [6 points]: RAID-033 is a new addition to the family of RAID schemes. A RAID-033 array consists of N data disks ($D_1, D_2, \ldots, D_N$) and an additional parity disk. Block i on the parity disk is equal to the `xor` of all blocks of disk D_i.

You are considering whether to use RAID-033, or to stick with the time-tested RAID-4. For each of the scenarios below, specify whether RAID-4 or RAID-033 is more resilient to the failure described. In all cases, you can assume that no additional failures occur.

A. Total failure of the parity disk.

(a) RAID-4 is more resilient to this failure.
(b) RAID-033 is more resilient to this failure.
(c) The two schemes are equally resilient to this failure.

(c). Both RAID-4 and RAID-033 can recover from this failure and reconstruct the parity from the other data blocks.

B. Concurrent failure of multiple blocks on the same disk.

(a) RAID-4 is more resilient to this failure.
(b) RAID-033 is more resilient to this failure.
(c) The two schemes are equally resilient to this failure.

(a). RAID-4 can reconstruct this data from the other disks + parity, but RAID-033 cannot.

C. Concurrent failure of block i on multiple (different) disks.

(a) RAID-4 is more resilient to this failure.
(b) RAID-033 is more resilient to this failure.
(c) The two schemes are equally resilient to this failure.

(b). Same reasoning as the previous part, but for RAID-033 instead of RAID-4.

2. [2 points]: True / False GFS does not allow for concurrent writes to the same file (i.e., writes from multiple distinct users at the same time).

False. GFS allows this. (End of Section 1 in the paper.)

3. [4 points]: Inspired by the first two questions on this exam, you decide to combine the ideas of GFS and RAID into a single system, D-RAID ("D" for distributed).

In D-RAID, files are broken into three chunks—F_1, F_2, F_3—and a parity block $P = F_1 \oplus F_2 \oplus F_3$ is calculated. F_1, F_2, F_3, and P are stored on four separate machines.

Circle True or False for each of the following. In all below, by "GFS" we mean the default configuration of GFS, with a replication factor of three.

A. True / False In general, D-RAID requires less storage space per file than GFS.

B. True / False Both GFS and D-RAID can recover from any single-disk failure.

C. True / False Both GFS and D-RAID can recover from any two-disk failure.

D. True / False In general, D-RAID has better performance for concurrent reads to the same file than GFS.

Initials:

A. True (D-RAID requires 1.33x space per file, GFS requires 3x space per file)
B. True (RAID and GFS can handle single-disk failure)
C. False (RAID cannot handle two disks failing)
D. False (D-RAID only has one copy of each piece of data, so all reads for a single block will go to one machine. GFS can load balance reads.)

II Atomicity

4. [4 points]: Consider the following **portion** of a write-ahead log. In this log, `UPDATE` records are of the form `UPDATE <var>=<old_value>; <var>=<new_value>`.

Log Entry	Transaction ID	Record
	...	
229	100	UPDATE A=0; A=10
230	100	UPDATE B=0; B=10
231	100	COMMIT
232	101	BEGIN
233	101	UPDATE A=10; A=25
234	101	UPDATE B=10; B=20
235	102	BEGIN
236	102	UPDATE C=0; C=30
237	102	UPDATE D=0; D=40
238	102	COMMIT
239	101	UPDATE D=40; D=55
240	101	UPDATE B=20; B=60
241	103	BEGIN
242	103	UPDATE C=30; C=70
	/** CRASH **/	

The system crashed after log entry `242`; no further entries were written.

A. After recovery, what are the values of `A`, `B`, `C`, and `D`? If it is impossible to determine the value for any of these variables, write "None" in the corresponding space.

Value of `A`: 10	Value of `C`: 30
Value of `B`: 10	Value of `D`: 40

Transactions 100 and 102 are the only ones that commit.

B. True / False The log portion shown is consistent with a transaction processing system that is using two-phase locking.

True. Transaction 100 releases its locks at commit. Transaction 101 holds locks for A and B while 102 holds locks that protect C and D, but 102 releases those on commit. 101 can then grab the lock for B, and 103 can later grab the lock for C.

Initials:

5. [6 points]: Consider a write-ahead logging system that uses non-volatile cell storage. Writes go to the log and then to cell storage. A write is complete once it has been written to the log and to cell storage. Reads are done directly from cell storage.

You decide to modify the system to improve the performance of **writes**. Which of the following three modifications will result in a new logging system with improved write performance **that still guarantees atomicity**?

(Circle True if the modification satisfies this property, and False otherwise.)

A. True / False

- Send writes to the log and cell storage in parallel instead of sequentially. A write is complete when both writes have returned.

B. True / False

- Add a volatile cache. Assume that the cache is large enough to hold all data.
- Send writes to the cache after writing to the log the log. Delay updating cell storage until the cache is flushed (in which case all cache updates will be written to cell storage). A write is complete once it has been written to the cache and to the log.
- Reads go directly to the cache.

C. True / False

- Add a volatile cache. Assume that the cache is large enough to hold all data.
- Send writes to the cache. Delay updating the log and cell storage until the cache is flushed (in which case all cache updates will be written to the log and to cell storage). A write is complete once it has been written to the cache.
- Reads go directly to the cache.

Only Part B is True. Part B descibes how a WAL with a cache and cell storage works. Parts A and C might result in faster rights, but they don't provide atomicity.

6. [4 points]: Answer the following True/False questions about LFS.

A. True / False LFS's garbage collection mechanism will occasionally copy data from its original location to elsewhere on the disk in order to minimize fragmentation.

B. True / False Given two segments, one with k dead blocks and one with ℓ dead blocks ($k > \ell$), LFS's garbage collection mechanism will clean the one with k dead blocks first.

True, False. LFS's garbage collection cleans cold segments before hot segments; cold blocks are not necessarily those with fewer dead blocks.

Initials:

7. [8 points]: A coordinator C is running a two-phase commit protocol with two servers, X and Y. During this process, you observe X's and Y's logs, specifically the entries for transaction ID 78. These logs use the same format as those in Question 4:

- The three entries in each row are the log entry number, the transaction ID, and the record.
- `UPDATE` records are of the form `UPDATE <var>=<old_value>; <var>=<new_value>`.

For each of the following log snippets, specify whether it is consistent with a correct two-phase commit protocol. In each case, we have shown you all of the log entries that have been written thus far pertaining to transaction 78; it is possible that more entries will be written in the future.

A. Consistent / Not Consistent

```
            X's log                                Y's log

141   78   UPDATE A=10; A=20              159   78   UPDATE B=70; B=80
142   78   PREPARE                        160   78   PREPARE
```

B. Consistent / Not Consistent

```
            X's log                                Y's log

141   78   UPDATE A=10; A=20              159   78   UPDATE B=70; B=80
142   78   PREPARE
143   78   COMMIT
```

C. Consistent / Not Consistent

```
            X's log                                Y's log

141   78   UPDATE A=10; A=20              159   78   UPDATE B=70; B=80
142   78   PREPARE                        160   78   PREPARE
143   78   COMMIT                         161   78   COMMIT
```

D. Consistent / Not Consistent

```
            X's log                                Y's log

141   78   PREPARE                        159   78   PREPARE
142   78   UPDATE A=10; A=20              160   78   UPDATE B=70; B=80
143   78   COMMIT                         161   78   COMMIT
```

A and C are consistent. A reflects a 2PC that hasn't reached the commit point yet, and C reflects a complete 2PC.

In B, we would not see `PREPARE` and `COMMIT` on X's log without also seeing at least `PREPARE` on Y's log; `COMMIT`s don't happen until all servers are `PREPARE`d.

In D, we would not see `PREPARE` before `UPDATE` for a transaction; all UPDATES complete before we `PREPARE`.

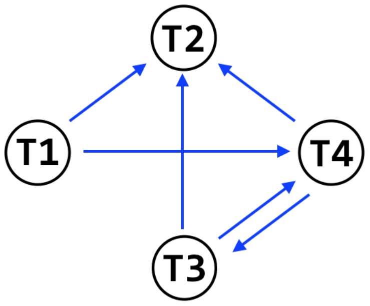
Initials:

III Isolation

8. [8 points]: A transaction processing system runs the steps of transactions `T1`, `T2`, `T3`, and `T4` in the order shown (time goes downwards; step i runs before step $i + 1$). All transactions commit after step 7 below.

T1	T2	T3	T4
1. `write(x)`			
	2. `read(x)`		
			3. `read(y)`
		4. `write(y)`	
			5. `read(y)`
			6. `read(x)`
	7. `write(y)`		

A. Draw the conflict graph corresponding to this schedule of steps.



B. In order to make this schedule conflict serializable, we would need to do which of the following? Circle the **best** answer.

(a) Nothing; this schedule is already conflict serializable.

(b) Swap the order of **exactly one** pair of operations (i.e., run B before A instead of A before B).

(c) Swap the order of **more than one** pair of operations.

(d) It is impossible to alter this schedule such that it is conflict serializable.

C. In order to make this schedule final-state serializable, we would need to do which of the following? Circle the **best** answer.

(a) Nothing; this schedule is already final-state serializable.

(b) Swap the order of **exactly one** pair of operations (i.e., run B before A instead of A before B).

(c) Swap the order of **more than one** pair of operations.

(d) It is impossible to alter this schedule such that it is final-state serializable.

The answer to Part B is (b): swapping the order of step 4 and either step 3 and 5 will remove one of the edges between T3 and T4, which breaks the cycle in the conflict graph. The answer to Part C is (a); this schedule is equivalent to a schedule that runs T1 first and T2 last (and T3 and T4 in between in either order).

Initials:

9. [4 points]: Consider a workload of $N + 1$ transactions, $T_0, T_1, \ldots, T_N$, all of which access a shared variable x.

- T_0 writes to x (but does not read its value)
- $T_1, T_2, \ldots, T_N$ read x (but do not write to it)

Ben is using two-phase locking to provide isolation, and is considering using reader-/writer-locks ("R/W locks") to improve performance.

Circle True or False for each of the following.

A. True / False R/W locks could improve performance because they will allow $T_1, T_2, \ldots, T_N$ to execute in parallel.

B. True / False R/W locks could improve performance because they will allow *all* transactions ($T_0, T_1, \ldots, T_N$) to execute in parallel.

C. True / False R/W locks could improve performance, but only if the read locks can be safely released early (i.e., before the commit point of the transaction).

D. True / False R/W locks will not improve performance because all accesses are to the same variable.

Only the first is true.

B is false because R/W locks don't allow for writes to execute while any other locks are held.

C is false because read locks can improve performance regardless of whether they're released early (they let multiple reads execute in parallel).

D is false since A is true.

IV Availability

10. [6 points]: Consider the following traces of requests and responses made by a single client process in PNUTS (responses to read requests contain the result along with a version number; responses to write requests return a version number on success). For each, indicate whether a client could observe this behavior on PNUTS. Assume that there are no failures (either servers or network). Assume that there can be other concurrent clients.[1]

A. Could Observe / Could Not Observe

```
Read-latest(‘‘foo’’) → (‘‘bar’’, 2.8)
Read-any(‘‘foo’’) → (‘‘bar’’, 2.6)
```

True. First goes to master in another data center. Second goes to a stale copy in local data center.

B. Could Observe / Could Not Observe

```
Read-latest(‘‘foo’’) → (‘‘bar’’, 2.8)
Read-any(‘‘foo’’) → (‘‘bar’’, 2.6)
Read-critical(‘‘foo’’, 3.6) → (‘‘bar’’, 2.7)
```

False. The read-critical for 3.6 would not return a version of 2.7

Note: If 3.6 had been 2.6 instead—which it would've been had we caught this typo—the answer would be true: First goes to master in another data center. Second goes to local data center, sees stale data. Third goes to local data center again, which by now received another update bumping it to version 2.7.

C. Could Observe / Could Not Observe

```
Read-any(‘‘foo’’) → (‘‘bar’’, 2.6)
Write(‘‘foo’’, ‘‘baz’’) → 2.8
Read-any(‘‘foo’’) → (‘‘bar’’, 2.7)
```

True. The first read went to the local data center. The write went to a master in another data center. The last read went to the local data center again, which by now received another update that was in flight at some point since the first read

[1]Thanks to Frans Kaashoek and Nickolai Zeldovich for this problem.

Initials:

11. [4 points]: Consider a replicated state machine (RSM) where each view has a single primary server P and *two* backup servers, B_1 and B_2.

A. In order to correctly provide single-copy consistency, which of the following must be true? Circle all that apply.

(a) There must also be two view servers.

(b) Only P will communicate with the view server(s).

(c) P must receive acknowledgments from both B_1 and B_2 before acknowledging an update to the coordinator.

(d) P must always communicate with B_1 before B_2 (or vice versa).

(e) None of the above.

Only (c) is true. RSMs have a single view server, and all backups must communicate with it (so the view server can detect liveness). There is no need for P to communicate with the backups in a particular order, so long as it gets ACKs from both before preceding.

B. Compared to an RSM where each view has only a single backup, what does the two-backup RSM improve? Circle the **best** answer.

(a) Atomicity

(b) Availability

(c) Consistency

(d) Isolation

(b) Availability. More backups means we're less likely to have a scenario when the primary and all backups have failed.

Initials:

12. [4 points]: Consider five machines using RAFT. The current log of each machine is given below. Log entries are specified as `<term number>.<update ID>`; we do not give the specific contents of the updates.

Leader's log:	1.1	1.2	1.3	2.1	2.2	3.1	3.2	3.3
F1's log:	1.1	1.2	1.3	2.1				
F2's log:	1.1	1.2	1.3	2.1	2.2			
F3's log:	1.1	1.2	1.3	2.1	2.2	3.1		
F4's log:	1.1	1.2	1.3	2.1	2.2	3.1	3.2	

A. Which of the log entries are *not* guaranteed to be committed? If they are all guaranteed to be committed, write "None".

3.2, 3.3. These entries have not been replicated on the majority of logs.

B. Which of the followers (`F1` through `F4`) could potentially lead the next election term, assuming that multiple other nodes may fail? If none of the followers could possibly lead the next term, write "None".

F3, F4. These nodes hold all committed values.

Initials:

V Authentication

13. [6 points]: Ben is working on a scheme to store salted passwords. For each user, he generates a salt s and sends that to the user. On his server, he stores three things:

- The user's ID (e.g., their username).
- $H(s)$, where H is a cryptographically-secure hash function.
- The result of $f(p)$, where f is a currently-unspecified function of the user's password p.

These are the only three things Ben stores for each user. He does not, for instance, store s.

Ben's authentication scheme is the following:

1. The client sends their username and password $p_{inputted}$ to Ben's server.
2. Ben calculates $f(p_{inputted})$ and compares it to the stored value of $f(p)$.
3. If the values are equal, the client is authenticated.

Consider a server-side adversary Eve who has the ability to read the data stored on the server. Eve has no other access to the server. Eve also **cannot** perform rainbow-table attacks, or any other brute-force attacks. Eve would like to be able to masquerade as a valid client (i.e., to be authenticated as a valid client despite not being one).

A. Which of the following values of f can Ben use on his server such that his authentication mechanism is resistant to this server-side adversary? Circle all that apply.

(a) $f(x) = x$

(b) $f(x) = H(x)$

(c) $f(x) = H(s \mid x)$

(d) $f(x) = H(H(s) \mid x)$

(b) and (d). (a) is out because that is equivalent to storing plaintext passwords. (c) is also out because the server can't compute it (server doesn't store s).

Having chosen a proper function f (one that is resistant to server-side attacks), Ben turns his attention to network adversaries who can observe and tamper with network packets. These adversaries do not have infinite computational resources, and they cannot launch any other types of attacks (e.g., they cannot DDoS Ben's server).

Instead of having clients send their password across the network in plaintext, he decides to have the clients send $H(p)$ instead.

B. True / False Because users now send the hash of their password, Ben's system is secure against these network adversaries.

We accepted both answers to this question. We intended for the answer to be False: network adversaries can't determine p, but they can launch replay attacks. However, we realized that the wording above this question was ambiguous, and did not make it clear that replay attacks were allowed.

Initials:

14. [8 points]: In Katrina's Amazing Web Suite (KAWS), users make requests to the server to purchase cars. There are only two models of cars, and thus only two valid requests:

- `BUY BATMOBILE`
- `BUY JOKERMOBILE`

Each request r is tied to a unique ID, seq_r, which is sent along with the request itself (see below).

Four months after its launch, KAWS is adding some security features. After authenticating users with their passwords, KAWS uses session cookies for quick authentication.

The KAWS developers have a few proposals for how to construct their session cookies. Below, | implies string concatenation, H is a cryptographically-secure hash function, and p refers to the user's password.

Proposal 1. $\{username,\ request,\ seq_{request},\ H(p \mid request \mid seq_{request})\}$

Proposal 2. $\{username,\ request,\ seq_{request},\ H(p \mid seq_{request})\}$

Proposal 3. $\{username,\ request,\ seq_{request},\ H(p \mid request)\}$

The KAWS developers are not concerned with an adversary that has access to the server, and so store passwords in plaintext on the server.[2] They are, however, very concerned about a network adversary between the client and server. This adversary does not know the user's password p, nor do they have infinite computational power. They are able to **observe**, **tamper** with, and **replicate** packets sent between the client and server.

Suppose a user U sends a request to buy a Batmobile (i.e., `BUY BATMOBILE`) as part of a cookie. The unique ID for this request is 186950.

A. Which of the proposed cookies prevent a network adversary from buying multiple Batmobiles on behalf of U? Circle all that apply.

(a) Proposal 1
(b) Proposal 2
(c) Proposal 3
(d) None of the proposals

(a) and (b). We need $seq_{request}$ to be included in the cookie to prevent this attack.

B. Which of the proposed cookies prevent a network adversary from tampering with U's packet, causing U to buy a Jokermobile instead of a Batmobile without the KAWS server noticing that something is amiss? Circle all that apply.

(a) Proposal 1
(b) Proposal 2
(c) Proposal 3
(d) None of the proposals

(a) and (c). We need $request$ to be included in the cookie to prevent this attack.

[2] This is a terrible idea, of course, but we truly are not concerned with their password-storage method for this problem.

Initials:

15. [2 points]: Which of the following **best** describes the job of a certificate authority? (Circle only one answer.)

(a) To generate and disseminate key pairs to users.

(b) To store name-to-key mappings.

(c) To authenticate name-to-key mappings.

(c). CAs do not generate key pairs (that would be *extremely* insecure), nor do they have to store them (though they can).

VI Secure Channels and Network Security

16. [6 points]: Alice is getting ready to communicate a message m to Bob using a shared symmetric key k. Both Alice and Bob know k; no one else does. Alice and Bob have many operations at their disposal:

- `MAC(k, m)`, which outputs the message authentication code of m using key k.
- `ENC(k, m)`, which outputs the encryption of m using key k.
- `DEC(k, c)`, which outputs the decryption of c using key k.

You can assume that the network is reliable: no packets will be dropped or corrupted by the network.

A. Consider a network adversary Eve who can **intercept** Alice's transmission and observe the contents. Eve does not have infinite computational power, nor can she tamper with packets; she can only passively observe.

Alice transmits `MAC(k, m)` to Bob (and nothing else).

(a) **True / False** Upon intercepting Alice's transmission, Eve cannot determine m.

(b) **True / False** Upon intercepting Alice's transmission, Eve canotn determine k.

Both are True. Note that at this point, Bob also cannot determine m.

B. After sending `MAC(k, m)` to Bob, Alice then sends `ENC(k, m)`. Unfortunately, Eve has now gained the ability to tamper with packets.

(a) **True / False** Upon intercepting Alice's transmission (of `ENC(k, m)`), Eve can tamper with it in a way that Bob will not be able to detect.

False. Bob has the mac and the encryption of m; he can check for integrity.

C. True / False If Eve had not gained the ability to tamper with packets, Alice could have communicated m to Bob securely by sending only `ENC(k, m)` (assume Eve is the only adversary in the network).

True. If Eve cannot tamper, we need only confidentiality, which is provided by encryption.

Initials:

17. [2 points]: True / False Without chains of trust, DNSSEC would be open to man-in-the-middle attacks.

True

18. [2 points]: True / False The reason Torpig uses domain flux is so that potential victims have a harder time blocking the bots that are sending traffic to their machines.

False. Domain flux makes the C&C server harder to take down.

19. [6 points]: Eve has rented a botnet, but has found it to have one flaw: the bots cannot spoof their IP addresses. This means that any traffic sent from bot B will have, as its source address, B's IP address.

Alice has her computer set up to drop all packets that have a source IP address from Eve's botnet. She's concerned about two attacks on her network:

- An **HTTP-flooding attack**, where Eve's bots send large, valid requests to her machine, preventing other non-malicious requests from getting through.
- A **DNS-reflection attack**, where Eve's bots send requests to DNS nameservers who then respond to Alice's machine.

A. True / False Alice is protected against the HTTP-flooding attack.

False. Alice's computer is blocking her packets; the network in front will still receive them. Queues will fill up, valid requests will be dropped.

B. True / False Alice is protected against the DNS-reflection attack.

True, because Eve's bot cannot *mount* such an attack anymore. DNS reflection requires spoofing IP addresses.

Initials:

VII Anonymity

20. [6 points]: Alice is using Tor with three proxies—P_1, P_2, and P_3—and is concerned about an adversary who has access to the proxy servers, and thus may be able to learn the state stored on each proxy.

To combat this adversary, Alice decides to use more proxies in her Tor circuit: instead of just P_1, P_2, and P_3, she sends traffic through $P_1, P_2, \ldots, P_N$, $N > 3$ (and then the traffic goes to its ultimate destination, S).

Answer True or False for each of the following questions about Alice's new technique.

A. True / False The latency of Alice's packets will increase (compared to her original three-proxy technique).

True. Packets travel more hops through the network.

B. True / False The size of Alice's packets will be roughly N times as large as her original data packet.

False. The encrypted value of a message is roughly the same size as that message, and additional headers do not add this much data.

C. True / False Eve will now have to work harder (access more state on more servers) to determine that Alice is communicating with S.

True. Eve needs to get state from the entire circuit, which is now longer.

21. [2 points]: Consider a Sybil attack, where a user creates multiple online identities in order to undermine a system. What aspect of Bitcoin is used to mitigate Sybil attacks? Circle the **best** answer.

(a) Cryptographic signatures

(b) Peer-to-peer networking

(c) Proofs of work

(d) None of the above

(c). The more accounts one creates, the more work they'll have to do.

Initials: