Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2022

Exam 2

There are **14 questions** and **12 pages** in this exam booklet. Answer each question according to the instructions given. You have two hours to answer the questions.

- The questions are organized loosely 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.
- You are not required to explain your answers unless we have explicitly asked for an explanation. You may include an explanation with any answer for possible partial credit.
- Some students will be taking a make-up exam at a later date. **Do not** discuss this exam with anyone who has not already taken it.
- 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 exam, 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.

Name: SOLUTIONS

Kerberos:

1. [8 points]: Consider the following log of three transactions, T1, T2, and T3. Transactions in this system run **serially**, not concurrently.

Cache flush

TID	T1	T1	T1	T1	T2	T2	T2	T3
	UPDATE	UPDATE	UPDATE	COMMIT	UPDATE	UPDATE	COMMIT	UPDATE
OLD	A=0	B=0	A=2		B=1	A=4		B=3
NEW	A=2	B=1	A=4		B=3	A=6		B=5

- Writes go to cache
- Flush pushes to cell storage
- Reads from cache, or cell if not in cache

The system using this log also employs cell storage and a cache, exactly as described in Lecture 17. Assume that the cache is large enough to hold all necessary values, and that the cache was flushed after T1's commit, but before any of T2's actions. It has not been flushed since then. There have been no failures of any sort.

A. After T3's update, what are the values of the variables A and B in both the cell storage and in the cache? Write your answers in the table below.

The values in cell storage reflect the point @ which the flush happened. So we see the values from T1.

Variable	Value in cell storage	Value in cache
A	4	6
B	1	5

Since all writes go to the cache, it has the most recent values of A & B.

Assume that the system crashes after T3's update. The pseudocode to recover this system is as follows (this code is **not** the same as the final Lecture 17 recovery code):

The code below lacks a "redo" phase, which is going to cause problems.

```
recover(log):
   commits = {}
   for record r in log[len(log)-1] .. log[0]: // read log backwards
      if r.type == COMMIT:
         commits.add(r.tid)
      if r.type == UPDATE and r.tid not in commits: // undo
         cell_write(r.var, r.old_val)
```

B. After the system recovers from the crash, what are the values of the variables A and B in cell storage? Write your answers in the table below.

The recovery code rolls back uncommitted updates, so we see B's value change from 5 to 3. A's value is unchanged because the code doesn't redo committed updates, and the cache wasn't flushed after T2's commit.

Variable	Value in cell storage
A	4
B	3

C. Does this system, as described, provide atomicity? Select the **best** answer.

(a) Yes
(b) No
(c) There is not enough information to tell

In this case, since reads are from the cache, or from cell storage if the variables aren't in the cache, we'll see reads that reflect an internal state in T2.

Initials:

2. [11 points]: Phi is running a MapReduce computation on some machines in a datacenter. The datacenter is *incredibly* lightly-loaded during this process: Phi's single MapReduce computation is the only thing running. Phi is watching some of the network traffic as this computation runs.

Phi first observes the map phase of this computation. She knows that the basic process is as follows: the controller[1] assigns map tasks to workers; workers read input data; workers perform the map computation; workers write output data.

During the map phase, Phi observes the controller assign map tasks to ten of the machines in the datacenter. She then observes a network transfer of input data from Machine 5 to Machine 6. However, at **no point** during the map phase does she observe this type of data transfer to Machine 7 (i.e., no machine in the network sends input data to Machine 7 during the map phase).

A. **True** / **False** Machine 7 must not have been assigned a map task.

Machine 7 could be reading input data locally

B. After it finishes its map computation, Phi sees Machine 6 send a message to the controller. What might the contents of this message be? Select **all** that apply.

(a) A response to a ping from the controller — Section 3.3 under "Worker Failure"

(b) Output data from the map computation — Figure 1. Workers do not send output data to the controller

(c) Meta-information about output data from the map computation — Section 3.1, Step 4

(d) None of the above

GFS replicates input data, so (a) is true. (d) is also true; MapReduce reads data from the closest replica, which is better than if we had no replicas to choose from.

C. Phi knows that MapReduce utilizes GFS as one of its underlying file systems. In the context of MapReduce, what does GFS improve? Select **all** that apply.

(a) GFS makes it more likely that the **input data** will be accessible throughout the MapReduce job, compared to using MapReduce with a filesystem that doesn't replicate any data.

(b) GFS allows MapReduce to continue running the computation even if the controller fails.

(c) GFS makes the map and reduce computations atomic. These computations aren't atomic.

(d) GFS can decrease the amount of time it takes for a machine to read the input data, compared to using MapReduce with a filesystem that doesn't replicate any data.

(e) None of the above

GFS doesn't replicate the controller

Since Phi has gotten familiar with GFS through her use of MapReduce, she starts using GFS on its own as a file system to back a web service she's running. She is using GFS exactly as described in the paper, with a default replication factor of three for every file.

D. Phi wants to improve the responsiveness of her system to client requests by increasing the replication factor of some files. Which type of files should she increase the replication factor for? Select the **best** answer.

(a) Very small files

(b) Very large files

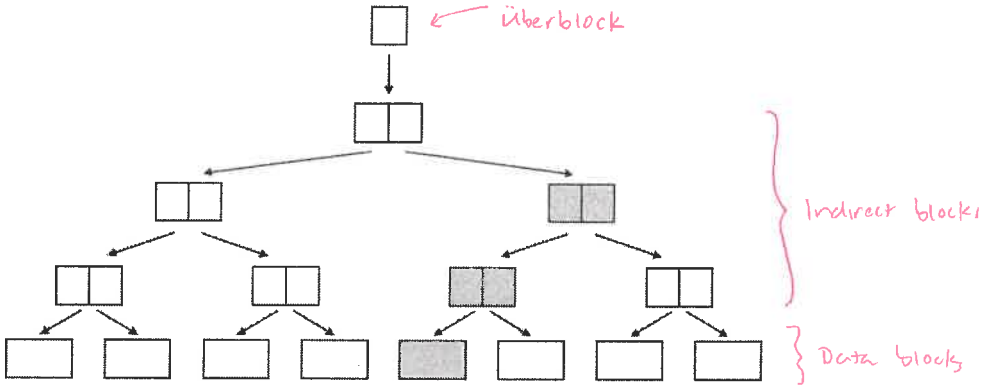
(c) Very popular files

This will let her spread requests out over more machines. (This is similar to the "hot spot" issue in Section 2.5 of the GFS paper.)

[1] Recall that we use the term "controller" where the paper uses "master".

Initials:

3. [8 points]: The image below illustrates a tree of indirect and data blocks that are part of ZFS, much like Figures 6-8 in the ZFS paper.



A. How many **data** blocks are pictured in the above image?

8 – The number of leaves in the tree.

B. Steve is observing a write in this filesystem. So far, three blocks shaded in gray have been copied and modified, but **no other changes have been made**. Thinking of this write as a transaction, has it reached its commit point yet? Select the **best** answer.

(a) Yes
(b) No
(c) There is not enough information to tell

The commit point is rewriting the überblock in place (Section 3.3 of the paper)

C. What is this copy-on-write technique most similar to? Select the **best** answer.

(a) GFS' record append
(b) Shadow copies
(c) Logging
(d) Two-phase commit

Shadow copies = copy over data + new modifications, then do atomic rename. A similar approach to copying/modifying data + indirect blocks and then rewriting the überblock.

Initials:

4. [9 points]: Consider the following four transactions. The syntax used here is the same as we used in Lecture 18, with line numbers in front of each `read()` or `write()`.

T1	T2	T3	T4
`T1.1 write(a)`	`T2.1 read(a)`	`T3.1 read(a)`	`T4.1 read(b)`
`T1.2 read(a)`	`T2.2 write(b)`		`T4.2 read(b)`
			`T4.3 write(a)`

Below are three possible schedules for these four transactions.

Interleaves T1 & T3

Schedule 1	Schedule 2	Schedule 3
`T1.1 write(a)`	`T1.1 write(a)`	`T1.1 write(a)`
`T1.2 read(a)`	`T3.1 read(a)`	`T1.2 read(a)`
`T4.1 read(b)`	`T1.2 read(a)`	`T3.1 read(a)`
`T4.2 read(b)`	`T2.1 read(a)`	`T2.1 read(a)`
`T3.1 read(a)`	`T2.2 write(b)`	`T4.1 read(b)`
`T4.3 write(a)`	`T4.1 read(b)`	`T4.2 read(b)`
`T2.1 read(a)`	`T4.2 read(b)`	`T4.3 write(a)`
`T2.2 write(b)`	`T4.3 write(a)`	`T2.2 write(b)`

Interleaves T3 x T4

The interleaved transactions are what we need to pay attention to for 2PL.

A. Which of the above schedules are conflict-serializable? Select **all** that apply.

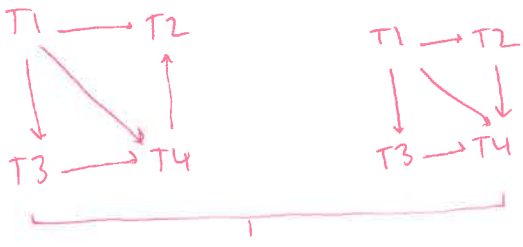
(a) Schedule 1
(b) Schedule 2
(c) Schedule 3
(d) None of the schedules are conflict-serializable

B. Which of the above schedules could be produced by the **basic two-phase locking protocol that uses exactly one lock per object** (no separate reader/writer locks)? Select **all** that apply.

(a) Schedule 1
(b) Schedule 2
(c) Schedule 3
(d) None of the schedules could be produced by two-phase locking

2PL is not guaranteed to produce all conflict-serializable schedules.

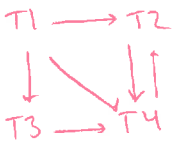
Schedule 1



Acyclic ⇒ conflict-serializable

Schedule 2

Schedule 3



Cyclic ⇒ NOT conflict-serializable ⇒ NOT 2PL

Schedule 1: T4 acquires b's lock, reads b twice, switch to T3, which acquires a's lock, reads, releases. T4 acquires a's lock, writes, releases both. ∴ 2PL

Schedule 2: cannot be produced by 2PL. T1 needs to hold a's lock through both operations; T3 can't interrupt & read.

Initials:

5. [8 points]: Consider a system running two-phase commit (2PC) for a particular transaction t. There is a single coordinator, C, which **never** fails. Three servers—S_1, S_2, and S_3—are involved in the transaction.

A. True / False Once S_1 has written a PREPARE record to its log for t, S_1 is guaranteed to commit t. C can safely abort at this point still. S_1 must be ready to commit, but is not guaranteed to

B. Suzanna observes this system at some point in time, and sees that S_1 has written both a PREPARE and COMMIT record to its log for t, while S_2 has only written a PREPARE record to its log for t. Select the **best** statement to describe this observation.

This is completely normal. It's what we'd see if everything was going fine, but S_2 just hadn't received the "commit" message from C yet.

(a) This observation is not possible in a correctly-working 2PC system.

(b) This observation is possible, but only if Suzanna is observing the system while S_2 is recovering from a failure.

(c) This observation is possible even if no part of the system has failed.

Assume, regardless of your answer to the previous question, that Suzanna is now dealing with a correctly-working 2PC system, and that transction t is complete and committed on all three servers.

Suzanna wants to optimize the performance of her system for the next transaction, t'. She decides to allow C to send S_1 a COMMIT message for transaction t' as soon as S_1 has responded to the PREPARE message for transaction t', regardless of whether S_2 or S_3 have responded to their PREPARE messages yet.

C. Assume that C does not fail, and that no messages from C to any other machines are lost. However, if a server fails, it will not receive any messages sent to it while it's down. With Suzanna's modification, and no others, is her system still correct?

(a) Yes

(b) No

Absolutely not. We cannot commit until all servers have prepared.

6. [4 points]: Consider a replicated state machine with a single coordinator C, a single view server V, and three replicas: S_1, S_2, and S_3. Currently, S_1 is the primary, and S_2 and S_3 are both backups; in terms of the view table, the current view—View 1—is `1: S1, S2, S3`.

This replicated state machine is attempting to achieve single-copy consistency.

A. True / False S_1 is allowed to complete updates and respond to C as long as it has received an ACK from at *least* one of S_2 or S_3.

B. True / False If S_3 fails, the view server does not need to update the view table to a new view (`2: S1, S2`) because there is still one backup available.

For A: This is a recipe for disaster. We could end up in a state where S_2 gets half the updates & S_3 gets the other half. If S_1 fails, neither backup is correct.

For B: False for @ least two reasons

1. S_1 needs ACKs from all backups, so the new server needs to make it "official" that S_3 is no longer up & running.

Initials:

2. If S_3 comes back online & S_1 & S_2 crash immediately (before transferring any data to S_3), S_3 could be promoted to primary in View 3, even though it's not up to date.

0 0
1 2
2 4
3 6
4 8
5 10 — 15 sec before Jay's read
6 12
7 14
8 16
9 18
10 20
11 22
12 24
13 26
14 28
15 30
16 32
17 34
18 36
19 38
20 40

7. [8 points]: A distributed system stores the value of a variable `x` on multiple replicas. At time 0, a process sends the write `x=0` to this system through some primary replica P. For the next twenty seconds, this same process sends a write to the system (again, through P) that increases the value of `x` by two every second. I.e., at time 1 the process sends `x=2`; at time 2, `x=4`; etc., all the way up to `x=40` at time 20.

No servers in this system fail, and no messages are lost. No other processes write to the variable `x`. You can assume that the first write (`x=0`) has made it to all replicas, i.e., there are no replicas that are storing an undefined value for `x`.

At time 10.1, Jay reads the value of `x` from this system on Replica A; the system returns `x=20`. At time 20.1, Jay reads the value of `x` again, this time from Replica B (A, B, and P are all different replicas).

The value that Jay reads at time 20.1 is affected by the type of consistency that the system provides. For each of the consistency models below, state the **smallest possible value** that Jay could read for `x`.

A. Eventual consistency: 0

Anything goes!

B. Consistent prefix: 0

The fact that Jay already read x=20 does not matter here.

C. Bounded staleness, with a staleness bound of 15 seconds

10

D. Monotonic reads 20

Note: Not 22. No older than his previous read, but not necessarily newer.

Initials:

8. [8 points]: Kriti owns four systems for storing usernames and passwords. Each of these systems stores salted hashes; they differ in how the salt is generated, and the type of hash function they use.

- System A uses "slow" hash functions and random six-digit salts (where the digits are taken from the numbers 0-9).
- System B uses "fast" hash functions and random six-digit salts.
- System C uses "slow" hash functions and a **fixed** six-digit salt. In this system, every user has the same salt.
- System D uses "fast" hash functions and a fixed six-digit salt.

Katrina gains access to each system and wants to determine which users are using one of the top ten most popular passwords. Suppose that it takes Katrina one second to compute a single slow hash, and one millisecond ($\frac{1}{1000}$ of a second) to calculate a single fast hash.

The longer it takes Katrina to determine which users are using the top ten most popular passwords, the more secure the system is.

A. Write down the systems in order of **least secure** to **most secure**. Pay attention to the time differences for a slow vs. a fast hash.

System A: 10 passwords $\times 10^6$ salts $= 10^7$ slow hashes $\rightarrow 10^7$ seconds

System B: 10 passwords $\times 10^6$ salts $= 10^7$ fast hashes $\rightarrow 10^7 \times 10^{-3} = 10^4$ seconds

System C: 10 passwords $\times$ 1 salt $= 10$ slow hashes $\rightarrow$ 10 seconds

System D: 10 passwords $\times$ 1 salt $= 10$ fast hashes $\rightarrow 10 \times 10^{-3} = 1/100$ seconds

Answer: D, C, B, A

B. Even though Kriti is doing her best to keep her systems secure, the name of the specific hash function that she's using is leaked to the public. Does this leak **substantially** affect the security of her system? Select the **best** answer.

(a) Yes
(b) No

There are only a few known hash functions; we assume an attacker knows them.

C. In Part A, did you write down the systems in order from least secure to most secure? We will not give partial credit if you accidentally write them in reverse order (i.e., most secure to least secure). (This part of this question is worth zero points)

Initials:

9. [6 points]: Consider the code below. This code is nearly identical to one of the examples from Lecture 22; the only change is the addition of two new variables, `x` and `y`

```
void win() {
   printf("code flow successfully changed\n");
}

int main(int argc, char **argv) {
   volatile int (*fp)();
   char x; // This line and the next are new
   char y;
   char buffer[64];
   fp = 0;
   gets(buffer);
   if(fp) {
       printf("calling function pointer, jumping to 0x%08x\n", fp);
       fp();
   }
}
```

argc
argv
BP
IP
fp 4B
x 1B
y 1B
buffer 64B

Amir runs this code on a machine that has no protection against stack-smashing. On his machine, integers are four bytes long and characters are one byte long. Any pointers—e.g., the base pointer, the instruction pointer—are also four bytes long. When a function is called, the following happens:

1. Any arguments to the function are pushed onto the stack
2. The base pointer (BP) is pushed onto the stack
3. The instruction pointer (IP) is pushed onto the stack
4. Local variables to the function are pushed onto the stack

Since Amir is overwriting fp, not IP, he doesn't really have to worry about this part

The list above describes **everything** that happens on the stack. There aren't, e.g., any other pointers pushed onto the stack between BP and IP. Amir's goal is to overwrite the variable `fp` and force the function `win()` to be called, by inputting a string that is longer than 64 bytes into this program.

A. How long should Amir's string be, in **bytes**?

64 + 1 + 1 + 4 = 70. Have to overwrite buffer, x, y, and fp.

These last 4 bytes are where the address of win needs to go

B. Where should the address of `win()` appear in the string? Select the **best** answer.

(a) As the first four bytes of the string
(b) As the final four bytes of the string
(c) Anywhere; it doesn't matter, so long as the address is there
(d) The address of `win()` doesn't need to appear in the string to get the code to jump to this function; Amir can use a random string, and it will work

Initials:

10. [9 points]: Alice and Bob are trying to communicate via a secure channel. They have correctly and securely shared two unique symmetric keys (one for each direction of communication). Alice and Bob also each have their own public/secret key pairs, and have shared the public keys correctly and securely. Only Alice and Bob know their respective secret keys.

Alice runs the secure channel protocol correctly, by doing the following:

1. Calculating $c = encrypt(k_a, m_a|seq_a)$, where m_a is the original message, k_a is the symmetric key that Alice uses to Bob, and seq_a is the appropriate sequence number.
2. Calculating $h = MAC(k_a, c)$.
3. Calculating $sig = sign(secret\_key_a, m_a, seq_a)$, where $secret\_key_a$ is Alice's secret key.
4. Sending $c|h|sig$ to Bob.

Bob, however, is tired of all of work he has to do to receive the message.

We still have confidentiality & security. They're open to an attack where Eve can masquerade as Alice.

A. Upon receiving a transmission, Bob checks that the received MAC is correct, but does **not** verify the signature. What attack(s) does this open Alice and Bob to? Select **all** that apply.

(a) A passive on-path attacker—one who observes messages but does not tamper with them—who does **not** know k_a could read the contents of Alice's message to Bob.

(b) An active on-path attacker could tamper with Alice's message and go undiscovered by Bob.

(c) None of the above.

B. Bob, begrudgingly, decides to check that the signature verifies. However, he doesn't process the sequence numbers correctly; rather than making sure he has received the next-expected sequence number and discarding duplicate messages, he simply ignores the sequence number and processes every message. What attack(s) does this open Alice and Bob to? Select **all** that apply.

(a) An active on-path attacker could successfully launch a replay attack by sending Bob additional copies of messages from Alice.

(b) An active on-path attacker could successfully launch a reflection attack at Alice by sending Alice copies of messages she sent to Bob. Messages A→B aren't valid messages B→A because of the different keys.

(c) None of the above

C. Bob gives in and decides to process sequence numbers correctly. However, he convinces Alice to use a standard hash function rather than a MAC. I.e., in Step 2, Alice calculates $h = hash(c)$. Bob still checks that the hash of the ciphertext matches what Alice sent (just as he would check that a received MAC was correct). What attack(s) does this open Alice and Bob to? Select **all** that apply.

(a) A passive on-path attacker could pre-compute many hashes and determine the value of m_a with a higher probability than otherwise

(b) An active on-path attacker could tamper with Alice's message and go undiscovered by Bob

(c) None of the above

Hashes don't provide integrity

Tempting! But Alice sends hash(c), not hash(m_a). Pre-computing those hashes would help an attacker learn c, but they already know c!

Initials:

11. [4 points]: Which of the following problems with DNS does DNSSEC attempt to solve? Select **all** that apply.

DNSSEC does not provide confidentiality

(a) Adversaries can observe the contents of messages between a DNS client and a nameserver

(b) Adversaries can utilize the DNS infrastructure to mount DDoS attacks DNSSEC makes this worse!

(c) Adversaries can masquerade as valid nameservers

(d) Adversaries can tamper with the contents of messages between a DNS client and a nameserver

(e) None of the above

12. [3 points]: Recall the DNS amplification attack from Lecture 25. Most botnets rely on this attack, or similar amplification attacks, to generate the bulk of their traffic. The Mirai botnet, however, does not. Why not? Select the **best** answer.

(a) Mirai bots were not programmed to execute amplification attacks.

(b) Mirai bots were mostly resource-constrained devices, which are more appropriate for executing other styles of DDoS attacks (e.g., TCP-state attacks).

(c) The majority of DNS nameservers blocked traffic from Mirai bots, rendering a DNS amplification attack ineffective.

13. [6 points]: Sabrina is experimenting with the toy example in Listing 1 of the Meltdown paper. She is using a system that has 4096-byte pages, just like in the paper. However, in her experiments, she uses the number 2048 where the example uses 4096; Sabrina's code is shown below:

```
1 raise_exception();
2 // the line below is never reached
3 access(probe_array[data * 2048])
```

This will not spread unique values onto different pages; instead, we get two values per page

Sabrina runs this changed example on her system. She observes that the access time for Page 63 is around 200 milliseconds; for every other page, the access time is around 400 milliseconds. Assume that pages are numbered starting at 0.

Given Sabrina's observations, what is the value of `data`? If there is more than one possible value of `data`, list them all.

126, 127

page 0: 0, 1
page 1: 2, 3
page 2: 4, 5
⋮
page 63: 126, 127
⋮
page k: $2k$, $2k+1$

Initials:

14. [6 points]: The TAs are experimenting with different approaches to Tor. In each section below, write **a single sentence** describing the problem with the TA's proposed scheme. **Do not write more than one sentence.**

A. Felipe proposes using Tor, but instead of using three proxies, he uses one hundred proxies between the source and destination.

Latency - it will take a long time to send from A to S

B. Amelia proposes using Tor, but with a single proxy. Traffic travels from the source, through this proxy, to the destination.

The proxy knows A & S are communicating

C. Jessica proposes using Tor with three proxies, but does not use onion routing. The source encrypts the data using the destination's public key, and proxies along the way only update packet headers. There are no "layers" of encryption to remove.

Attacker can observe same data from $A \rightarrow P_1$ & $P_3 \rightarrow S$

In the second half of 6.033, what was your primary means of consuming the lecture material? This question is worth zero points; we're interested in understanding what resources (recordings, notes, etc.) to offer students in Spring 2023.

(a) I watched the majority of lectures live and in the room.

(b) I watched the majority of lectures online, but at the regular time (MW2).

(c) I watched the majority of lectures online before the next day's recitation

(d) I watched the majority of lectures online, but not before the next day's recitation.

(e) I didn't watch the majority of lectures, but used the slides and notes.

(f) None of the above.

Initials: