Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2011

Quiz 2

There are 11 questions and 8 pages in this quiz booklet. Answer each question according to the instructions given. You have **50 minutes** to answer the questions.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

For true/false questions, you will receive 0 points for no answer, and negative points for an incorrect answer. Do not guess; if you are unsure about your answer, consult your notes. The total score for each numbered question (1 through 11) will be at least 0 (i.e., those scores cannot go negative).

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

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 DON'T USE YOUR LAPTOP FOR COMMUNICATION WITH OTHERS.

CIRCLE your recitation section number:

10:00	1. Lampson/Pesterev	
11:00	2. Lampson/Pesterev	3. Ports/Mutiso
12:00		4. Ports/Mutiso
1:00	5. Katabi/Raza	6. Strauss/Narula
2:00	7. Katabi/Raza	8. Strauss/Narula

Do not write in the boxes below

1-2 (xx/20)	3-4 (xx/20)	5-6 (xx/20)	7-9 (xx/20)	10-11 (xx/20)	Total (xx/100)

Name:

I Reading Questions

1. [10 points]: Based on the paper "Do incentives build robustness in BitTorrent?", which of the following statements are correct?

(Circle True or False for each choice.)

A. True / False A reference BitTorrent client will attempt to split its outgoing bandwidth equally between every peer it can connect to.

B. True / False BitTorrent permits content providers to shift bandwidth costs from their own ISP to those paid for by their content consumers.

C. True / False The *torrent* file that a BitTorrent client downloads before joining a swarm contains the IP addresses of the *seed* nodes.

D. True / False The *BitTyrant* client uses unequal outgoing bandwidth allocations as one strategy to improve its download performance.

2. [10 points]: Based on the paper "A case for redundant arrays of inexpensive disks (RAID)" which of the following statements are correct?

(Circle True or False for each choice.)

A. True / False A RAID array can improve reliability when individual disks fail independently from each other, but is less useful in the face of highly correlated failures.

B. True / False The RAID paper predicted that disk seek times would improve at an annual rate similar to improvements seen in transistor density in microprocessors.

C. True / False If average disk capacities grow proportionally faster over time than sequential data transfer rates between disks and disk controllers, the MTTR (mean time to repair) of a RAID array will decrease.

D. True / False If solid-state disks entirely replace spinning magnetic disks, none of the approaches described in the RAID paper will be needed anymore, since SSDs do not need to seek between different portions of the disk.

II Border Gateway Protocol (BGP)

These questions are based on reading "An Introduction to Wide-Area Internet Routing".

3. [10 points]: Ben Bitdiddle was asked to debug a BGP problem. Help Ben to brush up his information about BGP. Which of the following statements are correct?

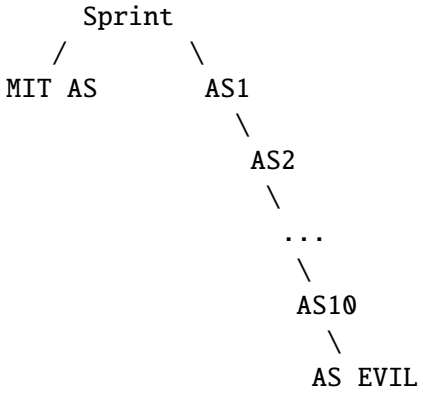
(Circle True or False for each choice.)

A. True / False To make money, an AS should announce the AS routes it learned from its customers to its peers.

B. True / False To make money, an AS should not announce the AS routes it learned from its peers to its provider.

C. True / False In BGP, an AS may learn many AS routes to an IP prefix.

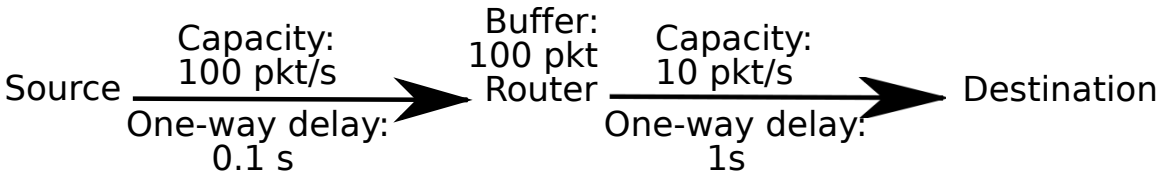
D. True / False In BGP, the customer-provider relationships imply that if an AS has announced a route for a particular prefix to a neighbor, and later it learned a new route to that prefix, the AS will announce the new route to the neighbor.



4. [10 points]: Consider the topology shown above. For any link, assume the higher node is a provider and the lower is a customer. MIT advertises prefix 18.* to Sprint. How can AS EVIL hijack most traffic destined to MIT from Sprint? (Describe briefly.)

Initials:

III Congestion control



Consider the above topology and assume the following:

- Link between Source and Router has a capacity of 100 packets/second and a propagation delay of 0.1 second in the forward direction.
- Link between Router and Destination has a capacity of 10 packets per-second and a propagation delay of 1 second in the forward direction. The transmission delays are 0.
- Acknowledgments from the destination to the source are delivered on a different path which has infinite capacity and zero delay, i.e., this means that acknowledgments will be delivered immediately from the destination to the source and will not suffer any drops.
- The Source and Destination can process packets very fast and the destination does not have any buffering limitations.
- The Router has a queue that can store a maximum of 100 packets.

5. [10 points]: Assume the Source uses a fixed window whose size is set to W. What is the maximum throughput in packet/sec that the source can deliver to its destination, if W=5, and then if W=20? (You can round your answer to the nearest integer.)

W = 5: _____ pkt/s

W = 20: _____ pkt/s

6. [10 points]: Assume that the source uses TCP. Focus on how TCP adapts its congestion window using AIMD as described in class. Ignore other details of TCP (e.g., slow start, fast retransmission and fast recovery). What is the maximum window size that the source's TCP will experience during AIMD?

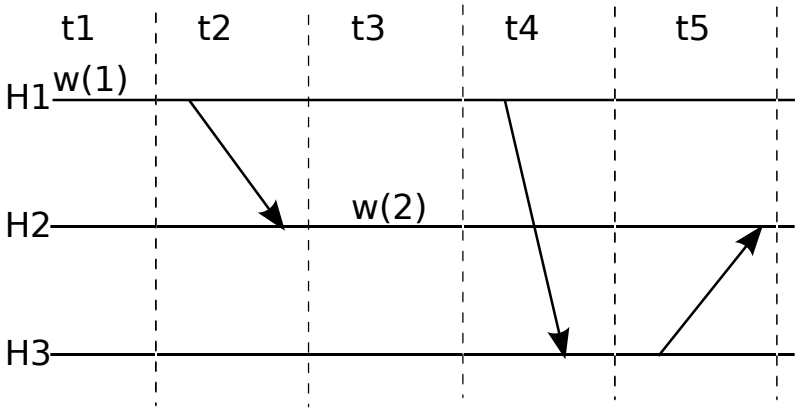
Maximum window size: _____ packets

Initials:

IV File reconciliation and time vectors

Ben designs a new tool for reconciling copies of a file on different computers. The scenario is a single user who has a number of computers, each of which may have a copy of a file f, and the user updates the copies of f independently from each other. For example, when the user is on an airplane, he can update only the copy on his laptop, and when the user gets to his office but forgets his laptop at home, he can update only the copy on his office machine. Ben's goal is that the different copies of f should behave as if there is a single copy of f.

To develop a design, he considers two update and reconciliation patterns. The first pattern for file f is as follows:



It shows 3 hosts: H1, H2, and H3. H1 writes (w) the value 1 to the file f, and then reconciles to H2 (i.e., changes on H1 are propagated to H2 and reconciled with changes on H2). Later H2 writes 2 to the file f and H1 reconciles to H3. Then, H3 reconciles to H2. To make it easy to talk about the different events, we have labeled them with real times t1 through t5, but you should assume that the clocks of the different computers are not synchronized and the computers have no agreement on this global time.

7. [5 points]: For the ideal outcome (i.e., f should behave as if there is a single copy), what value should f contain at H1, H2, and H3 after the above update and reconciliation pattern completes (i.e., at the end of t5)?

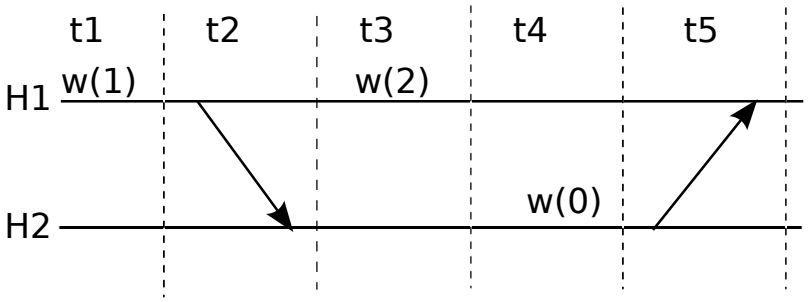
(Circle True or False for each choice.)

A. True / False H1: 1, H2: 2, H3: 2

B. True / False H1: 1, H2: 2, H3: 1

Initials:

After reading the Unison paper, Ben realizes that it isn't always possible to come to agreement between nodes what the value of f should be. Consider the following second pattern:



H1 writes 1 into f and reconciles to H2. Then, H1 and H2 modify f in different ways, and then H2 reconciles to H1 at t5. At t5, H1 and H2 have a *conflict* because there is no way to order the updates at H1 and H2 at t3 and t4 that is consistent with what could have happened with a single copy of f. A user needs to get involved to determine what the right value is.

Ben's goal is to design a tool that reconciles updates to a file f correctly when it is possible, but raises a conflict when it is not possible. His design is based on time vectors (introduced in the lecture on time, but you don't need to recall the lecture to be able to answer this question). A time vector is a vector of integers, where entry i counts the updates to f on computer i. For example, the vector (0, 0) on host 2 signals that it hasn't seen updates for f from H1, and that H2 itself hasn't done any updates either.

8. [10 points]: For the pattern above (the second pattern), what is the value of the time vector for f at each node at the end of t1 through t4? (complete the table)

time	H1	H2
t1	(1, 0)	(0, 0)
t2	—	
t3		—
t4	—	

9. [5 points]: How can Ben's design detect that pattern 2 is a conflict? Be specific, explain using the time vectors in the examples above.

Initials:

V Write-Ahead Logging

Ben hasn't solved enough problems in the last hour. Inspired by the lectures on atomicity and hands-on #5, he decides to build a database system for his bank. The database is a simple one: its cell storage is a table of account numbers and balances stored on disk, and it has only one operation: `credit(account, delta)` adjusts the balance of the specified account by the (possibly negative) amount `delta`.

Ben wants the database to support atomic transactions, so he adds the usual `begin` and `commit` operations, and stores a log on a separate disk. The operations are implemented as follows:

- `begin` appends a $\langle \mathsf{BEGIN},\ transactionid \rangle$ record to the log.
- `credit(account, delta)` updates the balance of the appropriate account in memory, but does *not* write the new balance to cell storage. It also appends to the log the record:
  $\langle \mathsf{UPDATE},\ transactionid,\ account,\ delta,\ oldbalance,\ newbalance \rangle$
- `commit` first appends to the log $\langle \mathsf{OUTCOME\ COMMITTED},\ transactionid \rangle$. It then writes out to cell storage any account balances modified by the transaction. When this is done, it appends to the log $\langle \mathsf{END},\ transactionid \rangle$

Ben knows that this protocol can support all-or-nothing atomicity across system crashes. He sketches a recovery procedure. After a crash, the system will scan the log and *redo* the effects of any transaction that logged an OUTCOME COMMITTED record but not an END record. But rather than implementing the recovery procedure, he puts the system into production immediately. After all, Worse Is Better, and he can always write that recovery procedure later...

Inevitably, Ben's system crashes before he gets around to writing that recovery procedure. The bank's customers are not amused. Panicked, Ben turns to you to help him recover the state of his database.

Initials:

The cell storage of the database at the time of the crash, and the last few entries of the log appear below. All earlier entries in the log correspond to transactions known to have completed in their entirety.

Cell Storage

AccountID	Balance
account1	$890
account2	$648
account3	$32
account4	$1500

Log

$\langle$BEGIN, $T_1\rangle$
$\langle$UPDATE, T_1, $account2$, 200, 448, 648$\rangle$
$\langle$OUTCOME COMMITTED, $T_1\rangle$
$\langle$END, $T_1\rangle$
$\langle$BEGIN, $T_2\rangle$
$\langle$BEGIN, $T_3\rangle$
$\langle$UPDATE, T_3, $account1$, 500, 390, 890$\rangle$
$\langle$UPDATE, T_3, $account4$, -500, 1500, 1000$\rangle$
$\langle$UPDATE, T_2, $account3$, 100, 32, 132$\rangle$
$\langle$OUTCOME COMMITTED, $T_3\rangle$

10. [10 points]: Fill in the values that will appear in the cell storage once recovery is completed using Ben's intended recovery scheme:

AccountID	Balance
account1	
account2	
account3	
account4	

11. [10 points]: Ben is worried that the write-ahead log will take up too much space. Given Ben's recovery procedure, which of the following options would reduce the size of the log, while still being able to persistently store data, and recover from crashes with all-or-nothing atomicity?

(Circle True or False for each choice.)

A. True / False eliminating the BEGIN record.

B. True / False removing both the $oldbalance$ and $delta$ fields from the UPDATE record

C. True / False removing both the $oldbalance$ and $newbalance$ fields from the UPDATE record

D. True / False periodically writing a CHECKPOINT record to the log which contains the ID of each uncommitted transaction, then discarding all log entries before the CHECKPOINT record.

End of Quiz I

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

Initials: