15-440 Distributed Systems
Extra Credit Studying

Name:
Andrew: ID

Hand in at the start of the final

- Please write your name and Andrew ID above before starting this exam.
- This exam has 17 pages, including this title page. Please confirm that all pages are present.
- This exam has a total of 118 points.

Question	Points	Score
1	8	
2	3	
3	2	
4	3	
5	12	
6	10	
7	14	
8	21	
9	20	
10	15	
11	10	
Total:	118	

Short Answers

1. (8 points) Circle True or False as appropriate. If you don't know the answer, come back at the end and GUESS; a blank answer is the same as a wrong answer, so guessing can only help.

   True False Using nanosecond-granularity timestamps in a distributed system provides a total ordering of events.

   True False Mutual exclusion, Safety and Fairness are the basic requirements for a distributed mutual exclusion algorithm.

   True False The domain name system (DNS) provides cache-consistent replication to client resolvers for scalability and fast notification of changes.

   True False Storing data along with its hash and then comparing the retrieved data to the hash is an effective method of fault detection.

   True False The C in ACID stands for Concurrency.

   True False Consistent Hashing moves, on average, only $\frac{\text{total keys}}{\text{total buckets}}$ elements when adding or deleting a node.

   True False Because they run on virtual machines instead of physical machines, users of services such as Amazon EC2 must first recompile their programs into VM-specific bytecode.

2. (3 points) Describe three different keywords or features of Go that empower the programmer to more easily create and manage concurrent code (one brief sentence each).

3. (2 points) Given a MapReduce job that reads in an *extremely large* amount of data, and is perfectly parallelizable, if we run it on a HDFS cluster with an infinitely fast network and an infinite supply of mapper, reducer, and storage nodes, what is the most likely cluster/system bottleneck that limits the execution speed of the job?

4. (3 points) Briefly explain how synchonous RPC and asynchronous RPC are different. Also briefly, when would asynchronous be a huge advantage over the other?

Security Protocols

5. (16 points) Two agents A and B want to communicate securely. Fortunately, they both are already registered with a key generation server S. Each agent shares a secret key with this server S: A has K_{AS}, and B has K_{BS}. The intent is to get the server S to generate a shared session key K^* that is only known by A, B, and S.

   In this problem, we present a few protocols and ask you to break them. We use the notation $X \rightarrow Y : M_1, M_2, ...$ to means that X sends the message $M_1, M_2, ...$ to Y. We use the notation $K(M_1, M_2, \ldots, M_n)$ to mean an encrypted version of the sequence $M_1, M_2, \ldots, M_n$ using key K.

   Assume the following:

   - Initially, only A and S know K_{AS}, and only B and S know K_{BS}.
   - The true server S is not malicious, but you as an imposter could try to pose as S.
   - Imposters can intercept any traffic, replay old messages, or inject new ones.
   - The encryption is secure, and the encrypted form of the sequence does not reveal any information about the encrypted form of the individual elements. For example, knowing $K(M_1, M_2)$ does not reveal any information about $K(M_1)$ or $K(M_2)$.

   To indicate that attacker X has intercepted a message sent by A, please write $A - \times \rightarrow X$ (arrow with an x on it)

Consider the following protocol:
1.$A \rightarrow B : A$
2.$B \rightarrow S : A, B$
3.$S \rightarrow B : K_{AS}(K^*), K_{BS}(K^*)$
4.$B \rightarrow A : K_{AS}(K^*)$

(a) (2 points) Suppose A and B want to communicate with this protocol. Using the same notation, show how an imposter X could pretend to be A (i.e., convince B that it's sharing the key K^* with A, when X really has K^*).

(b) (2 points) Suppose A and B want to communicate with this protocol. Using the same notation, show how an imposter X could pretend to be B (i.e., convince A that it's sharing the key K^* with B, when X really has K^*).

(c) (1 point) What is this type of attack called?

To avoid this attack, we revise the protocol and have the server S validate that the messages are really for the intended recipients with the extra information in this new protocol.
$1.A \rightarrow B : A, K_{AS}(B)$
$2.B \rightarrow S : A, K_{AS}(B), B, K_{BS}(A)$
$3.S \rightarrow B : K_{AS}(K^*), K_{BS}(K^*)$
$4.B \rightarrow A : K_{AS}(K^*)$

(d) (3 points) Suppose A and B want to communicate with this protocol. Suppose an impostor X had previously communicated with A. Using the same notation, show how an imposter X could pretend to be B (i.e., convince A that it's sharing a key K (K need not be the same as K^*) with B, when X really has K).

(e) (1 point) What is this type of attack called?

To avoid this attack, we revise the protocol once again to include randomly generated numbers R_a and R_b.
$1.A \rightarrow B : A, K_{AS}(R_a, B)$
$2.B \rightarrow S : A, K_{AS}(R_a, B), B, K_{BS}(R_b, A)$
$3.S \rightarrow B : K_{AS}(R_a, K^*), K_{BS}(R_b, K^*)$
$4.B \rightarrow A : K_{AS}(R_a, K^*)$

(f) (1 point) What do we call these random numbers R_a and R_b when they're used this way?

(g) (2 points) Upon completion of the protocol, no one other than A, B, or S can know the value of K^*. True/False?

Distributed Replication

6. (a) (4 points) If a Paxos group of $2f + 1$ nodes is split into two sets due to network partition, is it possible for these two sets to achieve agreement on different values? Briefly explain your reasons in 1–2 sentences.

   (b) (6 points) In a Paxos instance of $2f+1$ nodes, a leader is sending "Accept" messages to propose a value v to all nodes. The three sub-parts of this problem ask you what happens if your TA kills the leader after it has sent a total of x "Accept" messages and received a total of x "Accept-ok" responses. For each, indicate whether the value v will be committed as the final value after a new leader is selected and the protocol eventually runs to completion. Please answer "yes", "no", or "maybe", and for each, explain your answer in 1–2 sentences.

   1. $x = 1$ (1 "Accept" message and 1 "Accept-OK" response)

   2. $x = f$ (f "Accept" messages and f "Accept-OK" responses)

   3. $x = f + 1$ ($f + 1$ "Accept" messages and $f + 1$ "Accept-OK" responses)

Peer to Peer

7. (a) (2 points) Name an advantage to using a centralized p2p network.

   (b) (2 points) Name one major disadvantage of query flooding.

   (c) (4 points) Supernode based p2p networks allow us to alleviate some of the issues with centralized and flooding approaches that make both less scalable. How many supernodes would be needed for each supernode to only require $O(\sqrt{(n)})$ storage? Using both the storage and number of nodes requirements, explain how the supernode network is superior to both centralized and flooding approaches.

   (d) (2 points) Why can't Chord or BitTorrent easily support natural language search?

(e) (4 points) Explain how a node joins in a list-based DHT (Chord without the finger table optimization).

1. Compute ID, establish connection to ID's successor node in the ring.
2.
3.
4. Done.

Consistent Hashing

8. For this problem, you decide to use hash-based naming to build a distributed storage system. In the back end of your system, individual 4KB data blocks are stored in a key-value storage system, where `key` = (`SHA1(block contents) mod 10000`). You have implemented the system using the Chord DHT. Note that the use of mod means that keys go from 0 to 9999. Backend nodes store the block in a hash table on their local disk.

   You start out the system with three nodes, whose IDs are:

A	1593
B	8732
C	4568

   (a) (2 points) Identify the node responsible for answering queries for...

Key 9153	
Key 1472	
Key 7654	

   (b) (2 points) You insert two different data blocks into the ring. The blocks contain different content. What are the odds that by inserting the second block, you have caused data corruption for the first block? How would this happen?

   (c) (2 points) What is the major *advantage* of using name-by-hash to store the data blocks?

(d) (4 points) Identify what you think are the two biggest *performance disadvantages* of this scheme when you run the command: `cat hugefile | wc` ?

(e) (2 points) *Without* using its finger table, how many hops *on average* does it take to find a file in a Chord system on N nodes?

Assume that you then built this basic DHT into a massive, global public utility, usable by anyone. The servers for it span the globe, distributed equally between three mega-datacenters in the US, Europe, and Asia. You now use the full SHA-1 hash key as the DHT key.

(f) (3 points) For speed, when your system retrieves a medium or small-sized file containing less ≤ 100 blocks, it fetches all of the blocks in parallel. What would you expect to be a typical lower bound on the amount of time it would take to download to a fast, well-connected machine at CMU a 100 block file? Express your answer in terms of appropriate constants, not a numerical answer.

(g) (3 points) Explain a simple replication strategy that you could use (and is commonly used in DHTs) to both increase fault tolerance and reduce the expected latency for retrieving small files. *read the next part of the question before answering this one.*

(h) (3 points) Explain briefly (1–2 sentences) why you would not need to implement a consistency protocol to ensure that these replicas all agree on the same value, in the context of your system.

Byzantine Fault Tolerance

9. (10 points) A professor in some other class wants the students in her class to collaborate on solving an exam.

   Some of the students in the class are faulty: They either do not understand the material, or will deliberately lie to other students to try to make the others fail.

   The professor asks the students to form a practical Byzantine fault tolerant system to solve the exam. The professor ("the client") sends the exam out to all of the students. The students have designated one student to be the organizer who then directs the other students as per the BFT protocol (Assume that this student does properly forward the request.)

   (a) (3 points) If all 82 students are participating, what is the maximum number of students who can be wrong or malicious?

   (b) (3 points) The professor/client is watching the exams being handed in (replies to the client). After how many students' matching committed replies will she have to observe before knowing that any other non-malicious student will hand in the same answers?

   (c) (4 points) Why couldn't a Paxos approach help here? (1–2 sentences).

MapReduce and GFS

10. (15 points) MapReduce, running on top of a distributed file system, such as GFS or HDFS, has become a widely used technique for harnessing the power of many processing nodes to perform a single computing task. However, due to the programming model and the implementation, it is not suitable for all large-scale computing applications. For each of the following tasks T, indicate whether the following sentence is true or false:

    "MapReduce on GFS would be an appropriate way to implement T."

    In addition, give a brief (1–3 sentence) justification of your answer. That is, if your answer is *true*, then explain how it would be implemented via mapping and reducing. If your answer is *false* explain why this is so.

    (a) A phone company database, allowing users to look up phone numbers.

    (b) A tool for use by hackers that will scan through a collection of documents and identify possible social security and credit card numbers.

    (c) A program that can identify duplicates among a large set of files

(d) A system to model the airflow over an airplane wing design using a finite-element simulation.

(e) A distributed password cracker, based on brute force search.

HDFS

11. (a) (1 point) GFS and HDFS, by default, triplicate each data block on three different nodes. How many node failures can it tolerate (at least)?

    (b) (4 points) Triplication results in 200% space capacity overhead. So your TA wants to reduce this overhead by using RAID and recalls that RAID is very efficient in saving space. Thus instead of storing block A and block B each on 3 different DataNodes by triplication, he modifies HDFS to store block A and B on 2 different data nodes respectively, and also store 2 new parity blocks calculated from A and B (e.g., $A \bigoplus B$ and $f(A, B)$—some function other than exclusive-or) on 2 more different DataNodes. What is the storage overhead now? How many node failures could this scheme ensure to tolerate?

(c) (3 points) Your TA has a MapReduce job that has no output. Can you explain briefly the reason why your TA may observe this job runs slower on input data stored by the above scheme?

(d) (2 points) If we decrease the default HDFS block size, how could this change affect the NameNode?