15-440 Distributed Systems
Final Exam SOLUTION

Name:
Andrew: ID

December 12, 2011

- Please write your name and Andrew ID above before starting this exam.
- This exam has 15 pages, including this title page. Please confirm that all pages are present.
- This exam has a total of 80 points.

Question	Points	Score
1	8	
2	3	
3	6	
4	12	
5	10	
6	12	
7	13	
8	6	
9	10	
Total:	80	

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 DNS is delegated hierarchically by having, e.g., the root nodes tell resolvers which servers to query for ".com", and so on, until the client's query can be answered.

Solution: True

True False Domain Name System (DNS) resolvers use Paxos and invalidation messages to maintain the consistency of cached records.

Solution: False

True False In Tor's "onion routing", a compromised node in the middle of the network can identify the client and destination web site of the communication.

Solution: False.

True False If the server a client communicates with does not support any form of encryption (e.g., https), the client should use Tor so that nobody can overhear its traffic.

Solution: False.

True False Some Content Delivery Networks (CDNs) use DNS responses to direct clients to the closest CDN cache.

Solution: True.

True False A major challenge for Tor is finding nodes to act as exit points from the Tor network, becauses these nodes may appear to be performing illegal activities on the Internet.

Solution: True.

True False The goal of paravirtualization (e.g., Xen) is to make the guest operating system completely unaware that it is running on a virtual machine.

Solution: False. Guest OS must be modified.

True False The Map step of MapReduce provides a way to store and retrieve items according to their keys.

2. (3 points) Cloud computing services like Amazon's EC2 assign users virtual machines (VMs) instead of allocating physical machines directly. Doing so provides at least three major benefits to Amazon. Explain what these three benefits are, giving a brief motivation for each one.

Solution:

1. Security: virtualization gives the cloud provider more control over what the client OS can do.
2. Cost: multiple clients can use the same physical machine instead of having to rent full machines.
3. Scheduling flexibility: virtualization allows the cloud provider to make better scheduling decisions by making it easier to migrate client operating systems from one physical machine to another.

Crash Recovery

3. (6 points) In the following, keep your answers brief and to the point.

   Suppose we wish to implement a transaction processing system that maintains ACID properties even in the presence of crashes. In event of a crash, any information stored on disk can be retrieved, but any data stored in memory will be lost.

   Briefly describe *one serious shortcoming* of each of the following implementations:

   (a) The database is updated on disk with each transaction

   Solution: This would yield unacceptable performance, given the slow speed of disk writes

   (b) The database is kept in memory and on disk, with the copy on disk updated every 50 transactions.

   Solution: This would violate durability when the system crashes before the disk copy has been updated.

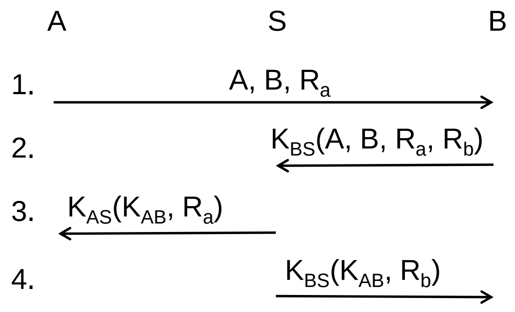
   (c) The database is kept in memory. A log file is maintained on disk recording every transaction.

   Solution: The log file would never stop growing. It would be inefficient in terms of size and recovery time.

Security Protocols

4. (12 points) The following figure illustrates a protocol between two agents A and B, and a server S. The intent is to get the server to generate a shared session key K_{AB}, and to also enable A and B to make sure they are communicating with each other.

   Each agent shares a private key with the server S: A has K_{AS}, and B has K_{BS}. Values R_a and R_b are randomly generated "nonces" (number-used-once). The notation $K(M_1, M_2, \ldots, M_n)$ means to generate a message containing an encrypted version of the sequence $M_1, M_2, \ldots, M_n$ using key K.


A
S
B
1.
A, B, R_a
2.
K_{BS}(A, B, R_a, R_b)
3.
K_{AS}(K_{AB}, R_a)
4.
K_{BS}(K_{AB}, R_b)

   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 there could be an imposter S′ trying to pose as S.
   - There could be an imposter A′ or B′ trying to pose as A or B.
   - 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 $K(M_2)$.
   - We will say that a message is *freshly generated* if it must have been created some time after the initial message in the protocol.

For each of the following statements, state whether it is true or false. Give a brief (one or two sentences should suffice) justification for your answer.

Solution: This protocol has a vulnerability that was not recognized at the time the exam was created. An imposter B$'$ could intercept message 1 and send the message $K_{B'S}(A, B', R_a, R_{b'})$ to S. S would generate a session key $K_{AB'}$ and send a message $K_{AS}(K_{AB'}, R_a)$ to A. Since message 3 does not identify the involved parties, A would not realize that it had created a session with B$'$ rather than B.

In the below answer key, we give both the intended answer and the correct answer, in terms of this vulnerability. We accepted either version but required the answers be consistent and properly explained.

Note: this vulnerability could be avoided by including the identities of A and B in messages 3 and 4.

(a) S can be certain that message 2 was freshly generated by B.

Solution: False. This could be a replay of an old message, since there is no guarantee that R_a is fresh.

(b) A can be certain that message 3 was freshly generated by S.

Solution: True. Only someone who knew K_{AS} could have generated $K_{AS}(R_a)$

(c) B can be certain that message 4 was freshly generated by S.

Solution: True. Only someone who knew K_{BS} could have generated $K_{BS}(R_b)$

(d) Upon completion of the protocol, A can be certain that it has established a session with B.

Solution: Intended: True. Since A knows that R_a is fresh, it can be certain that S must have received message 2 from B.

Corrected: False. As noted above, the session could be with an imposter B$'$.

(e) Upon completion of the protocol, B can be certain that it has established a session with A.

Solution: False. A has not established its identity with either B or S.

(f) Upon completion of the protocol, no one other than A, B, or S can know the value of K_{AB}.

Solution: Intended: True. The only risk is for an imposter A$'$ of A, but it could not read K_{AB} without knowing K_{AS}.

Corrected: False. The session key could be known to imposter B$'$.

Distributed Replication

5. (a) (2 points) Consider an implementation of the Paxos algorithm where a leader waits for less than a majority of acceptors to answer OK to a Prepare or an Accept message before it proceeds to executing the next steps. Explain how such an implementation can break the Paxos guarantees.

   Solution: A majority of acceptors may already have chosen a different value. The leader may get responses from acceptors from outside that majority and may proceed to propose and then commit a new value (which breaks the property that only one value can be chosen as a result of running a Paxos instance).

   (b) (2 points) Explain why Paxos cannot tolerate f failures with less than $2f+1$ nodes.

   Solution: Assume we try to tolerate f faults with only $2f$ nodes. If f nodes fail, a leader cannot hear back from a majority of acceptors ($f+1$ nodes), and a value will never be chosen.

   (c) (6 points) One common assumption in Paxos is that different leaders use different proposal numbers. A colleague shows you an implementation of Paxos where:

   1. Different leaders can use the same proposal numbers, and
   2. An acceptor rejects a Prepare or Accept message only if the proposal number in the message is strictly smaller than the largest proposal number that acceptor has seen (i.e., exactly as presented in class and in the course notes).

   Is this Paxos implementation correct? Find a counterexample or explain briefly why it is correct.

   Solution: It is not correct. Counterexample: consider three nodes A, B, and C. At the same time A and B want to be leaders and propose different values v_A and v_B, with the same proposal number. Because of lost messages, they only reach C (not each other). C will OK both their proposals, and will then accept both v_A and v_B. Both A and B will think they have a majority (themselves + C) and will commit two values.

Peer to Peer

6. (a) (2 points) In a centralized p2p network (such as the old Napster), how many indices must be searched if a client wants to locate a particular file?

   Solution: 1

   (b) (2 points) Name one major disadvantage of the centralized p2p system (from a distributed principles point of view)

   Solution: - Single point of failure - Server processes everything - Server must keep track of a potentially very large number of clients - more?

   (c) (2 points) Query flooding is an alternative design that solves some of the problems of centralized p2p and eliminates the central server. However, it changes the mechanics of peer interactions significantly. Explain (1 sentence each) how a newly-joining node publishes the files they wish to make available, in...

   A centralized p2p network:

   Solution: They send their list of files and metadata to the server.

   A query flooding p2p network:

   Solution: They don't do anything - queries come to them.

   (d) (2 points) One popular improvement upon query flooding is to move to a "supernode" flooding architecture. Using N as the number of nodes in the network and, S as the number of supernodes ($S << N$), explain the benefit of moving to this supernode architecture.

   Solution: Queries require now $O(S)$ messages instead of $O(N)$. In addition, if the nodes used as supernodes are more stable or have higher capacity, can further improve the performance or stability of the network.

(e) (2 points) What is a common mechanism used to limit the propagation of queries in a flooding network?

Solution: TTL - time to live - scoping. Also known as hop count limits, etc.

(f) (2 points) List one typical criterion for selecting a node to be promoted to a supernode. Explain in one sentence why such a choice would improve network stability.

Solution: How long a node has been part of the network (time), because how long a node has been around is a good predictor of how long it will be around.

Consistent Hashing

7. (a) (2 points) In a distributed hash table (DHT) using a *ring mapping*, explain how a key is mapped to a specific node?

   Solution: Keys and nodes map to points in a ring space. A node handles all keys mapped between it and its successor.

   (b) (2 points) In the most basic form of DHT systems, nodes simply track their predecessor and successor. Very briefly state what problem this leads to as the network grows larger.

   Solution: Performance problems as nodes forward a key resolution request to larger chains of neighbors.

   (c) (2 points) The Chord scheme overcomes this problem by having each node maintain a "finger table". Given starting node j and destination node k, where $k > j$, how much is the distance between the two nodes reduced with each hop?

   Solution: It decreases exponentially, or it is cut in half

   (d) (2 points) In a Chord structured DHT with N nodes, how many hops would a lookup operation require?

   Solution: $\log(N)$

   (e) (2 points) Assume a Chord network in which node q is the successor of node p. During operation, node p discovers that its successor link is no longer consistent because q has updated its predecessor link. What does this change imply must have happened in the network?

   Solution: A new node joined the network 'between' p and q.

(f) (3 points) An optimization to Chord involves storing several nodes for each entry in the finger table instead of just one. Explain an important benefit this optimization confers in a globally distributed DHT.

Solution: (1) This allows routing based on proximity, which would reduce slow routes that criss-cross the globe.

(2) It provides a fallback in case one of the nodes in the finger table is unreachable.

Byzantine Fault Tolerance

8. (6 points) In distributed systems where messages are asynchronous and failures can be Byzantine, we have to use at least $n = 3f + 1$ replicas in total to tolerate f faulty replicas. Show that this bound is tight, i.e., that $n \geq 3f + 1$ *must* hold in order for the system to work properly.

   To approach this problem, consider this scenario: a client sends the same command to each of n servers and then waits for the servers to execute the command and send back the result. Ideally, the client should receive n matching results, but remember that f servers may be malicious. Additionally, messages are asynchronous, so they can be delayed for an indefinite amount of time. Show that $n \geq 3f + 1$ must hold for the client to always be able to identify the correct result.

   Solution: From the point of view of a client of the system, f out of the total of n nodes may be faulty and not responding, so the client must be able to function with just $n - f$ responses. But the messages are asynchronous, so the f unreceived messages may in fact have been from slow non-faulty nodes, which means that f out of the $n - f$ responses may be wrong. Even so, the messages that are correct must outnumber those that are not for the client to identify which is which: $n - 2f > f$, and therefore $n > 3f$.

MapReduce and GFS

9. (10 points) MapReduce has proved an extremely popular framework for distributed computation on large clusters, because it masks many of the painful parts of ganging together thousands of nodes to accomplish a task.

   (a) In general high-performance computing (HPC), programmed using message passing, what is the technique used to provide fault tolerance? (Very short answer)

   Solution: Checkpoint/restore

   (b) Identify two important limitations that MapReduce places upon the Map functions so that the framework can hide failures from the programmer.

   Solution: They must be side-effect free, deterministic, and idempotent.

   (c) What advantage does this give MapReduce over MPI for failure handling and recovery? (There are several—list the one(s) you think is most important)

   Solution: Recovery can be done by only executing the work that was being handled on the failed machine, not the entire cluster.

   There is no need to take snapshots or to barrier sync the cluster.

   (d) How does MapReduce handle "straggler" tasks that take longer than all of the others (e.g., perhaps they're running on a slower machine or one with buggy hardware)?

   Solution: It executes them in duplicate on another machine towards the end of execution. (Not needed for answer: It can do so because of the same properties of the tasks above - namely, that they can be freely re-executed and produce the same result.)

   (e) MapReduce and the Google File System (GFS) were designed to work well together. What important optimization in MapReduce is enabled by having GFS expose block replica locations via an API?

   Solution: The MapReduce scheduler can arrange for Map tasks to execute on the same node that stores the data, avoiding a copy across the network.