CS425 Fall 2015 – Homework 3
(a.k.a. “The Martian”)  

Topics: Consensus, Networks, RPCs, Concurrency Control, Replication Control, 2PC/Paxos, Key-value stores/NoSQL (Lectures 13-21)

Instructions:

1. Attempt any 8 out of the 11 problems in this homework (regardless of how many credits you’re taking the course for). If you attempt more, we will grade only the first 8 solutions that appear in your homework (and ignore the rest). Choose wisely!
2. Please hand in hardcopy solutions that are typed (you may use your favorite word processor. We will not accept handwritten solutions. Figures and equations (if any) may be drawn by hand.
3. Please start each problem on a fresh sheet (not just page), and type your name at the top of each sheet.
4. Homeworks will be due at the beginning of class on the day of the deadline. No extensions.
5. Each problem has the same grade value as the others (10 points each).
6. Unless otherwise specified, the only resources you can avail of in your HWs are the provided course materials (slides, textbooks, etc.), and communication with instructor/TA via discussion forum and e-mail.
7. You can discuss lecture concepts and the questions on Piazza and with your friends, but you cannot discuss solutions or ideas. All work must be your own.

Prologue: It is the year 2030 A.D. Most of you are in your thirties. Cloud computing, as we know today, does not exist – it’s now called “Solar Computing”. Sure, there are a few quantum computers here and there, but transistor-based computers still rule the roost in the 2030s. Datacenters are still around, and all the distributed computing concepts you’re learning today in CS425 still apply. The only catch is that datacenters are much smaller (100x) than they were back in 2015 A.D. – this means an entire AWS zone from 2015 can now be stored in one rack!

Anyway, Moon has been colonized by humans. Man has been to Mars. The next step is Mars colonization. In order to kickstart the Mars colonization process, a manned spacecraft New Horizons X is being launched to Mars with ten astronauts on board. The
spacecraft carries its own powerful datacenter. You are one of the astronauts on board. You are the sole “Solar Computing Specialist.” You must ensure that you troubleshoot and solve all problems that arise in the on board distributed system (solving any 8 out of 11 problems would also suffice to save the mission).

Any resemblance to persons, places, or events, living or dead, past, present, or future, is purely coincidental.

**Problems:**

1. 3…2…1… Liftoff! You’re off to Mars. During liftoff you’re browsing code (what else?). Within the first minute after launch, you realize that one of the Earth programmers has written an algorithm for synchronous consensus (the same as that discussed in class) for a rack of N=10 machines, however they have only configured the consensus to run for 5 rounds. Your fellow astronauts believe this is not a problem, and you know they are wrong. You know that once you exit the Earth’s atmosphere, cosmic rays will increase in frequency and they can knock out an arbitrary number of machines simultaneously. Now all you have to do is to show a counter-example to convince your fellow astronauts that with an arbitrary number of failures this programmed synchronous consensus will not work. Quick, you’re about to exit the atmosphere!

2. To fix things up, you decide to implement Paxos in the datacenter on board. You use a stock implementation of Paxos, but while perusing the code you realize that instead of majority (for a quorum), it uses 2/3rds of the processes (everywhere in the protocol where a quorum is needed). You need to answer three questions:
   a. Is this new version safe?
   b. Is this new version live?
   c. Is this new version faster or slower than the majority (just greater than 50%) version of the protocol? Why?

3. When New Horizons X is passing through the Van Allen belts, the spacecraft’s reactor and engines suddenly shut down. You trace this to a bug in the transaction management system on board. You discover a log of two transactions executed concurrently by two clients – T1 was executed by the Command and Control Center, and T2 was executed by the reactor computer (a and b are objects at the server):
   T1: read(a, T1); write(b, caz, T1); read(a, T1);
   T2: read(b, T2); write(b, bar, T2); write(a, baz, T2);
The transaction management system seems to think each of the following interleavings is serially equivalent. For each of the following interleavings, say if (and why/why not) it is serially equivalent:

a. read(a, T1); write(b, caz, T1); read(b, T2); write(b, bar, T2); read(a, T1); write(a, baz, T2);
b. read(b, T2); write(b, bar, T2); read(a, T1); write(b, caz, T1); write(a, baz, T2); read(a, T1);
c. read(a, T1); read(b, T2); write(b, bar, T2); write(b, caz, T1); write(a, baz, T2); read(a, T1);
d. read(b, T2); read(a, T1); write(b, caz, T1); read(a, T1); write(b, bar, T2); write(a, baz, T2);

4. Now your spaceship is passing by the Dark Side of the Moon. It’s a glorious view! While solving the above problem, you get a chance to look at the exclusive locking code that transactions use, and you see that the programmer has each transaction acquiring a lock for an object just before each access to that object, and then the transaction releases the lock object right after that access – if the object is needed later, it will be locked again by the transaction. Will this system satisfy serial equivalence? In either case, present a proof/counterexample.

5. About a quarter of the way to Mars, things are calm and everyone on board (and on Earth) seems laidback. You decide to spend your time optimizing the part of the code that automatically logs everything that happens on board during each hour. The system maintains a separate object for each hour of each week, – thus there are $24 \times 7 = 168$ total objects. While looking at the code for the locking system, you realize that it uses exclusive locks on objects, but you feel you can do better. You decide that a read-write lock system is more appropriate. You also realize that the objects have a hierarchical relationship to each other. To make it even more efficient, you decide to implement a hierarchical locking scheme. Thus you use additional objects “Week”, “Monday”, “Tuesday”, … “Sunday”. The hierarchy has Week at its root, which has seven children (sub-objects) for the days of the week, each of which in turn has 24 children (sub-objects) for the hours of the day. Four types of locks are possible on each of these objects (168 + 7 + 1 = 176 total objects) – i) the usual read lock and write lock (indicating that that particular is being read), but it also means that none of the contained objects (sub-objects in the hierarchy) should be accessed in inconsistent ways, and ii) an I-read-lock and I-write-lock, also called an intention lock, on an object which indicates that one of the contained objects (sub-objects in the hierarchy) is going to be locked in that mode. Your job is to come up with the table that specifies for each lock request whether it conflicts with an existing lock on an object (and thus
should wait), or whether it does not (and thus can proceed down the hierarchy). Your goal should be to maintain correctness while improving performance (TPS).

6. You’re about halfway through the trip to Mars. Unfortunately, the spacecraft has been on a wobble over the last 6 hours, and this is affecting all the astronauts adversely – many are throwing up, and two are unconscious. You’re still doing well (physically and emotionally speaking), mostly because you were trained well at Illinois. You trace the wobbling problem to the on board storage system, and after a drill down you realize it’s because of false positives in the Bloom filter that this storage system is using. The Bloom filter uses m=32 bits, and 4 hash functions h1, h2, h3, and h4, where hi(x) = ((x +x^2)*i) mod m. In the run you’re looking at, the following integers have been inserted into the system: 1997, 2003, 2005, 2007, 2012 (for those of you who care, these are years during which a spacecraft landed on Mars). The Bloom filter code thinks all the following are present in the Bloom filter. For each case, say whether it is a false positive or not (for those of you who care, the first three are years of manned landings on the Moon by the US):
   a. 1969
   b. 1971
   c. 1972
   d. 2015
   e. 2030

7. To lower the probability of these false positives, you decide to double the amount of memory in the Bloom filter. Your spaceship captain suggests merely using a double-sized Bloom filter with 64 bits and 4 hash functions. You think differently – you want to use two identical Bloom filters, each with 32 bits and 4 hash functions (hash functions different), and insert all elements in both, and when checking for membership return true if and only if the element is present in both Bloom filters. Does your captain’s approach or your approach yield the lower false positive rate? (If needed, you can assume that about 5 elements are present in the Bloom filter typically). You can use the Web as a resource to find false positive rates for Bloom filters (but solve the problem yourself!).

8. The replication control in the datacenter on board is configured to use either active or passive replication. However, if there is a failure, it must failover quickly and not lose any data. Fortunately, you know that during some (known) portions of the journey, cosmic ray frequencies will be low and the chance of failures is almost zero. At other durations (also known), especially during solar flares, the chance of failures is high.
   a. During the no-failure times, will you choose active or passive replication? Say why. Think which is faster for clients!
b. During the high failure times, will you choose active or passive replication? Say why.

9. Bam! Your New Horizons X spacecraft has just suffered a massive strike from an asteroid! The damage to the spacecraft has cut off communications between the fore and aft portions of the craft. You are left in the fore partition with two other astronauts, and they are panicking. Calmly, you tell your fellow astronauts that this is called a partitioned system, you learnt this in CS425, and you’ve got this. You’ve got a few choices for how to handle this partition – these choices are listed below. For each of these choices, say if it: i) violates consistency (and why), and ii) if it violates availability (and why).

   a. Allow both fore and aft partitions to process reads, but only the fore partition to process writes.
   b. Allow both fore and aft partitions to process writes, but only the aft partition to process reads.
   c. Allow only the partition that has at least a quorum number of servers (measured across both partitions) to execute writes; reads can be executed in both.
   d. Allow reads and writes in a partition only if it has a quorum number of servers responsive (measured only within that partition).
   e. Until partitions are repaired, allow only reads but no writes.
   f. Allow only partitions with a quorum of servers (measured across both partitions) to execute writes and reads.

10. Whew! Now that the spacecraft has been repaired (after the asteroid strike) and the partition has healed, you realize you’re almost at Mars! To make sure nothing goes wrong during landing, it’s time to make sure your database system is using the “best” consistency model. For some of these questions, you may need to use the Web to find out definitions of the consistency models (be careful about the veracity of your sources, and try to corroborate sources if needed!). You have implementations for the following five consistency models: eventual consistency, linearizability, causal, red-blue, FIFO, sequential. Among these:

   a. Which is the “fastest” (most available) models among these?
   b. Which are the two “most consistent” models among these?
   c. What is the difference between the two most consistent models among these? If necessary, given an example.

11. W00t! Your spacecraft has landed on Mars! As a sign of respect for your firefighting skills as the “Solar Computing Specialist” and for rescuing the mission multiple times, your fellow astronauts have unanimously decided to give you the honor of being the first person to land on Mars! But before you go, you want to make sure that the Cassandra cluster on board is configured right,
otherwise it might send wrong signals to your spacesuit and cause it to blow up (like in the movie “Total Recall”, the older one with Arnie). The Cassandra cluster is running a RackInferring snitch. You see a log that makes the following statements. Is there a bug in the system? For each statement say if it’s true or false, and why.

a. IP addresses 111.123.213.1 and 111.123.213.122 and 111.123.213.56 are in the same rack in the same datacenter.

b. IP addresses 4.5.6.7 and 4.123.213.122 and 4.123.211.56 are in different datacenters, though the last two are in the same datacenter (though different racks).

c. IP addresses 3.4.5.6 and 3.5.6.7 and 3.6.7.8 are in three separate datacenters.

d. IP addresses 3.4.5.6 and 3.6.5.6 and 3.7.5.4 are in different racks in different datacenters.

e. (Optional, no points for this part, answer only if you want to) When you set your foot on Mars, as the first human to do so, what will be your first words to the world? (Neil Armstrong had great words, but try to make yours epic!).