CS425 Fall 2017 – Homework 3  
(a.k.a. “The Buggy Martian”)  

Out: Oct 19, 2017. Due: Nov 7, 2017 (Start of Lecture. 2 pm US Central time.)  

**Topics:** Snapshots, Multicast, Consensus, Paxos, Leader Election, Mutual Exclusion (Lectures 13-18)  

**Instructions:**  
1. **Questions 7 and 10 are mandatory – everyone must answer these questions.**  
2. **For questions 1-6, 8-9, attempt any 6 out of these 8 problems.** Do not attempt more (if you do, we will grade only the first 6 out of these 8). Choose wisely!  
3. 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. Online students can email solutions, and MCS-DS students must upload on Coursera. If you’re not online/MCS-DS, and are traveling, please make other arrangements – we don’t accept email submissions.  
4. Please **start each problem on a fresh sheet (not just page), and type your name at the top of each sheet.** Staple all your sheets together.  
5. Homeworks will be **due at the beginning of class on the day of the deadline.**  
   **No extensions. For DRES students only:** once the solutions are posted (typically a few hours after the HW is due), subsequent submissions will get a zero. **All non-DRES students must submit by the deadline time+date.**  
6. Each problem has the same grade value as the others (10 points each).  
7. 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.  
8. 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 middle age. 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 2017 A.D. - this means an entire AWS zone from 2017 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, things, 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=7 machines, however they have only configured the consensus to run for 4 rounds before terminating. Your fellow astronauts believe this is not a problem, but you know they are wrong. 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. You have detachment from the rocket! You switch communications on. You see a chart of the multicast communications between your spacecraft New Horizons X (NHX), Earth station, Moon station, and Mars (unmanned). There’s a bug in the FIFO ordering algorithm. To debug, mark the timestamps at the point of each multicast send and each multicast receipt. Also mark multicast receipts that are buffered, along with the points at which they are delivered to the application.
3. As New Horizons X is passing through the Van Allen belts, the spacecraft’s reactor and engines suddenly shut down. Oops, you realize that you should have used causal ordering in the previous timeline (Question #2). Can you redo it quickly before your spacecraft crashes? Mention all timestamps and buffering.

4. It’s still not stabilized! For the figure in Question 2, if we were using causal-total (hybrid) ordering, give ALL the possible orderings in which multicasts may be received at different processes. (You don’t need to show buffered messages).

5. To fix the consensus algorithm, one of your fellow astronauts has written a variant of the stock implementation of Paxos. In a datacenter with N processes, you know already (via previous experiments) that you will never have more than N/3 failures (anything more, and the spacecraft is doomed anyway, so this is a safe assumption to make). While perusing the code you realize that instead of majority (for a quorum), it uses ((N/3)+1) processes (everywhere in the protocol wherever a quorum was previously needed). You need to quickly figure out the answer to three questions:
   a. Is this new version live?
   b. Is this new version safe?
   c. Is this new version faster or slower than using the majority? Why?

6. As your spaceship is passing by the Dark Side of the Moon, catastrophe strikes as the spacecraft communications go out! You narrow it down to a problem with the Chandy-Lamport. Recovered logs show the following partial timeline, which captures all application messages but not all marker messages. Calculate ALL possible snapshots that could result from this partial timeline. (Hint: study the figure carefully. Hint 2: The “Earth” process is the initiator of the snapshot).
7. (Mandatory Question – Everyone must answer.) Your spacecraft needs to perform a slingshot (gravity assist) in order to land on Mars. Before the slingshot, you realize the single leader election algorithm will not work, and that for fault-tolerance you will need multiple leaders. Nodes have IPv4 addresses with 4 octets a.b.c.d. Solve this group leader election problem:

- Safety: For each non-faulty process \( q \), \( q \)’s elected = NULL, or set of processes \( p \) such that if \( p \) has IP address a.b.c.d then no other process \( p' \) with subnet mask a.* is a leader if \( \text{IP}(p') \) is lexicographically smaller than \( \text{IP}(p) \).
- Liveness: For all runs of election, the run terminates AND for each non-faulty process \( q \), \( q \)’s elected is not NULL.

Modify the Bully Algorithm described in lecture to create a solution to the group leader election problem. You may make the same assumptions as the Bully Algorithm, e.g., synchronous network. Note that processes may be in different subnets (i.e., there might be multiple a.* subnets in the group). Briefly discuss why your algorithm satisfies the above Safety and Liveness, even when there are failures during the algorithm’s execution.

8. Bam! Your New Horizons X spacecraft has just suffered a massive strike from an asteroid! Alarms are going off all around you. You quickly figure out that the fault lies with the mutual exclusion algorithm implemented in the system – if you can fix it, the spacecraft will return to normal operations.

You see that the datacenter uses the Ricart-Agrawala algorithm for mutual exclusion but the implementation has a bug: instead of implementing the (Lamport timestamp, process id) pair, the bug uses the reverse, i.e., (process id, Lamport timestamp) pair. The rest of the Ricart-Agrawala algorithm remains unchanged. Your fellow astronaut says this algorithm, even without failures: a) satisfies safety, b) satisfies liveness, and c) satisfies causal ordering. Is he right on any of these counts (which ones)? Give a proof or counter-example for each.

9. 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 the virtual synchrony implementation in the datacenter is correct. You see the following instances in the log. For each of the following executions, say whether it is a) correct (and why), or b) if it is incorrect (and what change in the timeline would have made it correct).

a. \( p1, p2, p3 \) each deliver a view \( V11=\{p1,p2,p3\} \). Then \( p1 \) multicasts message \( M32 \), however then \( p3 \) fails, and \( p1 \) and \( p2 \) have deliver the next view \( V12=\{p1,p2\} \), and only then do \( p1 \) and \( p2 \) deliver \( M32 \).
b. p1, p2, p3 each deliver a view V11={p1,p2,p3}. Then p1 multicasts message M32, however then p3 fails, and p1 and p2 deliver the next view V12={p1,p2}. Neither p1 nor p2 deliver M32 in either view.

c. p1, p2, p3 each deliver a view V11={p1,p2,p3}. Then p1 multicasts message M32, however then p3 fails, and p1 and p2 deliver the next view V12={p1,p2}. Neither p1 nor p2 deliver M32 in either view. But p3 delivers M32 right before it fails (but does not send out any further messages after this delivery).

d. p1, p2, p3 each deliver a view V11={p1,p2,p3}. Then p1 multicasts message M32, and p1 delivers it immediately. However then p3 fails and p1 and p2 deliver the next view V12={p1,p2}. Only then does p2 deliver M32.

e. p1, p2, p3 each deliver a view V11={p1,p2,p3}. Then p1 multicasts message M32 and concurrently p2 multicasts message M45. Both p1 and p2 deliver each others’ messages and their own. But p3 fails and may have received some of these messages. Then p1 and p2 deliver the next view V12={p1,p2}.

f. p1, p2, p3 each deliver a view V11={p1,p2,p3}. Then p1 multicasts message M32, and delivers it immediately, and then p1 fails. p2 and p3 each respectively deliver the views {p2} and {p3}. M32 is never delivered at p2 or p3.

g. p1, p2, p3 each deliver a view V11={p1,p2,p3}. Then p1 multicasts message M32. A fourth process p4 joins, and all processes p1-p4 deliver the next view V12={p1,p2,p3,p4}. M32 is delivered then at processes p1-p4.

h. p1, p2, p3, p4 each deliver a view V11={p1,p2,p3}, while p4 delivers a view V11={p1,p2,p3,p4}. Then p1 multicasts message M32 and all 4 processes deliver it in the view V11.

10. **(Mandatory Question – Everyone must answer.)** 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 the spacecraft doors won’t open! You’re stuck in the exit hatch. Thankfully you have access to a computer, and you quickly figure out the problem lies with the snapshot algorithm that someone implemented. Causality is not cutting it, and you need a snapshot at a physical instance of time. Thankfully you know that you have a synchronous system where there is perfect time synchronization (zero clock skew and zero clock drift). Quick! Write a modified version of the Chandy-Lamport algorithm discussed in class to calculate a snapshot. You can still use marker messages, but the goal is to
calculate the snapshot at a physical instance of time. Don’t forget that the snapshot needs to capture both process states and channel states.

a. Explain your algorithm clearly. We recommend writing (loose) pseudocode similar to what is in the slides.

b. Prove that your algorithm captures a consistent cut. (Yes, a formal proof.)

c. (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!).