CS425 Fall 2018 – Homework 3

(a.k.a. "First Human (on Mars)")

Out: Oct 18, 2018. Due: Nov 6, 2018 (Start of Lecture. 2 pm US Central time.)

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

Instructions:

1. **Attempt any 8 out of the 10 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 **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 (and scanned).
3. **All students (On-campus and Online/Coursera)** – Please submit PDF only! Please submit on Gradescope. [https://www.gradescope.com/]
4. Please **start each problem on a fresh page**, and **type your name at the top of each page**.
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.

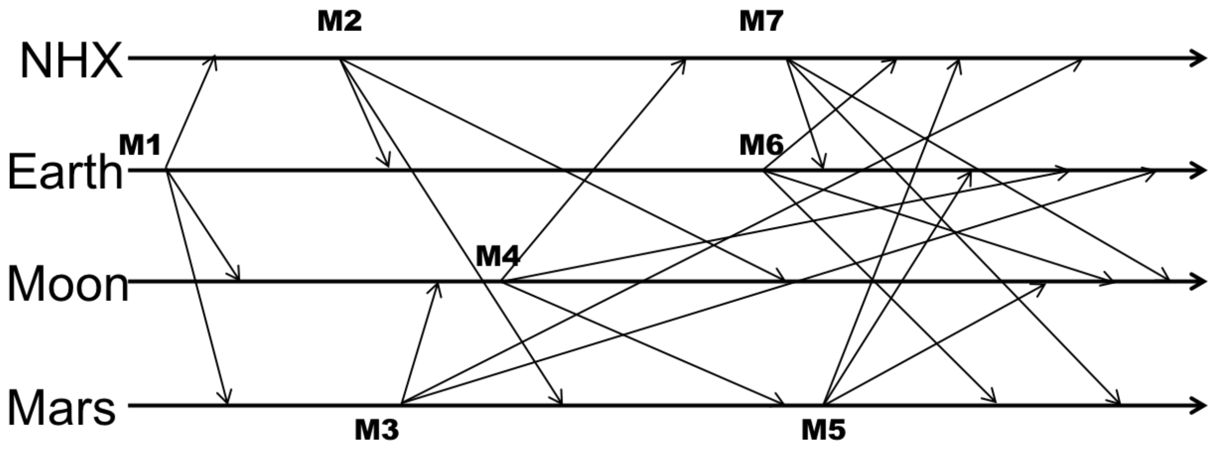
Prologue: It is the year 2039 A.D. Most of you are in your middle age. Cloud computing, as we know today, does not exist – it's now called "Solar Computing". Sure, there are a few quantum computers here and there, but transistor-based computers still rule the roost in 2039. Datacenters are still around, 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 the 2010s, but more powerful – this means an entire AWS zone from 2010s can now be stored in one rack!

Anyway, Moon has been colonized by humans. Man is next going to land on Mars. 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 10 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:

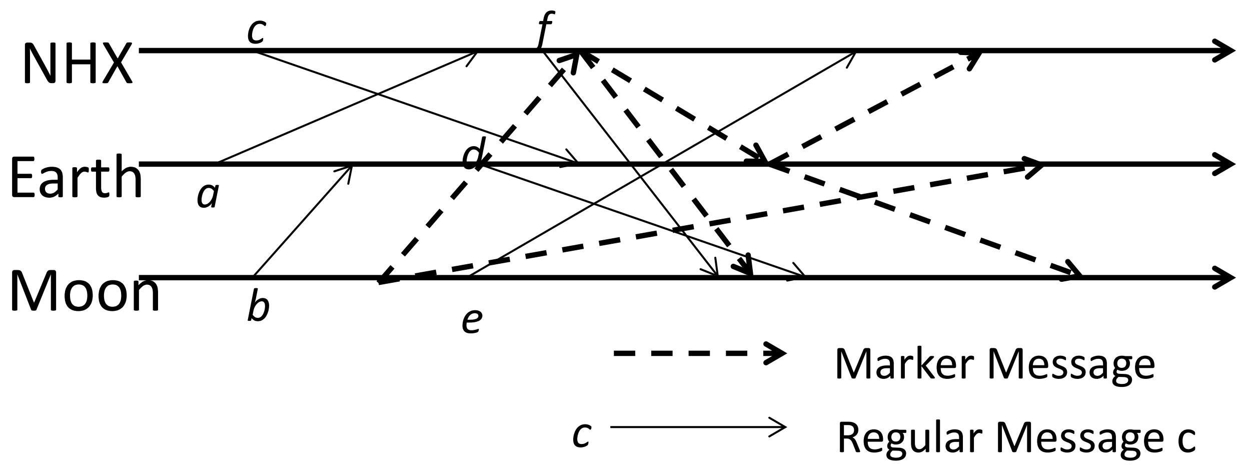
1. 3…2…1… Liftoff! You're off to Mars. During liftoff you're browsing code (what else?). Within the first minute after launch, you realize that one of the runs for the synchronous consensus (the same as that discussed in class) may have a bug. Concretely, see the synchronous consensus algorithm for a system of $N > 5$ machines, and imagine it being run in an asynchronous system. If both the 3rd and 4th rounds deliver NO messages at all (messages are sent, just dropped and never delivered, and the algorithm moves on to the next round), will all such runs of the algorithm still be correct? If yes, prove it. If no, show a counter-example. 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! You see in the log a timeline of 3 processes that are using causal-total (i.e., hybrid) ordering for their multicasts (as per the definition discussed in class). Each process sends 3 multicasts. The *j*-th multicast (in sequential order) sent by process P*i* is named as M*ij*. Thus, the multicasts are M11, M12, M13, M21, M22, M23, M31, M32, and M33. You are told the causal relationships between multicasts are as follows: M12 → M31, M21 → M31, M33 → M13, M33 → M22, M12 → M21, M22 → M13. Write down **ALL** possible causal-total orderings of these multicasts.
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 (N large enough), you realize that in some places in the implemented algorithm, instead of majority (for a quorum), it uses L=((5N/12)+1) processes. There are 3 variant algorithms:
   i. Election phase uses L instead of N/2+1, but Bill phase uses N/2+1.
   ii. Election phase uses N/2+1, but Bill phase uses L.
   iii. All phases use L instead of N/2+1.

   For each of the above 3 cases, answer the following 3 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. Now your spaceship is passing by the Dark Side of the Moon. It's a glorious view! To ensure things are working properly you decide to run the Chandy-Lamport snapshot algorithm on the ongoing communications between your spacecraft, and the manned Earth station, and manned Moon station. But due to a crash at the different stations, the algorithm only outputs the following timeline. Quick, it's up to you to manually calculate the snapshot!



7. Your spacecraft needs to perform a slingshot (gravity assist) in order to land on Mars. However, this means going through the dreaded Asteroid belt between Mars and Jupiter! Before the slingshot, you realize the above leader election algorithm will not work, and that for fault-tolerance you will need multiple leaders. Solve the k-leader election problem (for a given value of k). It has to satisfy the following two conditions:
   • Safety: For each non-faulty process p, p's elected = of a set of k processes with the lowest ids, OR = NULL.
   • Liveness: For all runs of election, the run terminates AND for each non-faulty process p, p's elected is not NULL.
   Modify the Bully Algorithm described in lecture to create a solution to the k-Leader Election problem. You may make the same assumptions as the Bully Algorithm, e.g., synchronous network. 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 need to quickly design a file system to back up data. In doing so you encounter a new mutual exclusion problem. Consider a file F that is present in a distributed system of N processes (N large). There are no failures or message losses in the system. The mutual exclusion required on this file has the following safety and liveness conditions (different from those discussed in lecture):

Safety: At most one process may obtain write access to the file at a time. At most *k* processes may obtain read access to the file at a time. If any process has write access to F, no other process should be able to read it. If any process has read access to F, no other process should be able to write it.

Liveness: Requests to access and release the resource eventually succeed.

Answer these three parts:

a. Briefly describe a token ring-based distributed algorithm for the above problem (pseudocode would be a good idea). Your algorithm must not have more than *k* token messages in the system at any point of time (simultaneously).
b. Argue briefly that your algorithm guarantees both Safety and Liveness properties defined above (a formal proof is not required, however you are free to write one).
c. What are the bandwidth usage (for enter and exit separately), client delay, and synchronization delay for your algorithm (consider the worst case)? You need to calculate the two delays only for the enter operation. Show separate calculations for read and write operations. Show all your calculations.

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 else 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 it is not delivered at p1, p2 or p3. Then p3 fails and p1 and p2 deliver the next view V12={p1,p2}.
   c. 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.
   d. 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, but they never deliver their own multicasts. But p3 fails and never receives either message. Then p1 and p2 deliver the next view V12={p1,p2}.
   e. 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.

f. 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.

10. 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. Oops, this is embarrassing! Thankfully you have access to a computer, and you quickly figure out the problem *may* lie with the snapshot algorithm that you re-implemented. Here it is:

```
First, Initiator P$i$ records its own state

Initiator process creates special messages called "Marker" messages

for $j=1$ to $N$ except $i$

    P$i$ sends out a Marker message on outgoing channel $C_{ij}$

    Starts recording the incoming messages on each of the incoming channels at
P$i$: $C_{ji}$ (for $j=1$ to $N$ except $i$)

Whenever a process P$i$ receives a Marker message on an incoming channel $C_{ji}$

if (this is the first Marker P$i$ is seeing)

        P$i$ records its own state first

        Marks the state of channel $C_{ji}$ as "empty"

        for $j=1$ to $N$ except $i$

                P$i$ sends out a Marker message on outgoing channel $C_{ij}$

        Starts recording the incoming messages on all of the incoming channels at
        P$i$: $C_{ji}$ (for $j=1$ to $N$ except $i$)

else // already seen at least one Marker message

    – if this is the (N-1)th (last) marker being received at P$i$,

                for $j=1$ to $N$ except $i$

                        Mark the state of channel $C_{ji}$ as all the messages that have
                arrived on it (until now) since recording was turned on for $C_{ji}$
        else do nothing
    Terminate when all processes have received (N-1) markers each
```

Answer the following questions:

i. Is this algorithm correct? If yes, prove so. If no, give a counterexample (draw a timeline).
ii. How would you fix this algorithm? Quick, your oxygen is running out!
iii. (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!).