CS425 Fall 2019 – Homework 3

(a.k.a. "Mad Astra")

Out: Oct 17, 2019. Due: Nov 12, 2019 (Start of Lecture. 2 pm US Central time.)

Topics: Snapshots, Multicast, Consensus, Paxos, Leader Election, Mutual Exclusion (Lectures 12-18)

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 **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.

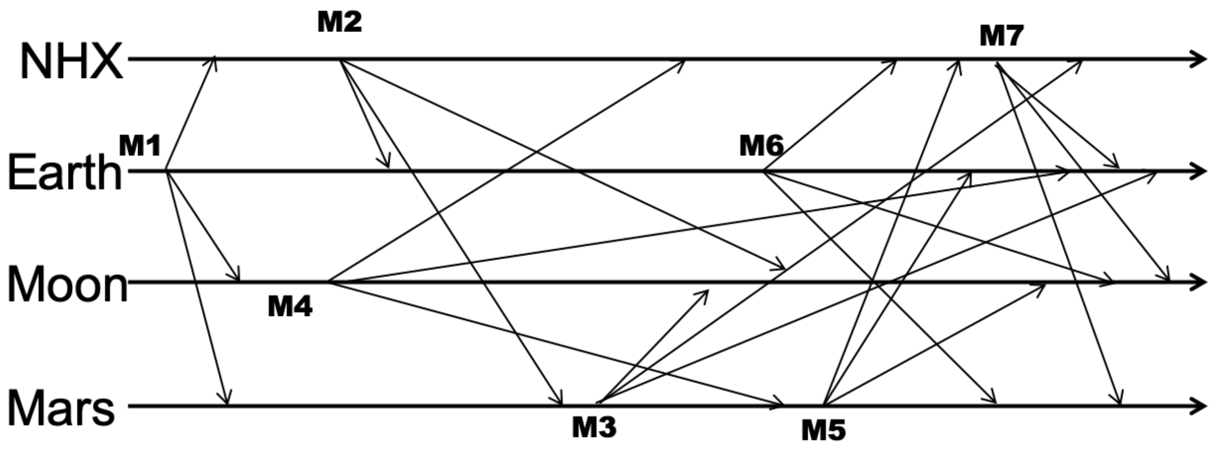
Prologue: It is the year 2049 A.D. Most of you are in your middle age. Cloud computing, as we know today, does not exist – it's now called "Solar Computing". Sure, there are a few quantum computers here and there, but transistor-based computers still rule the roost in 2049. Datacenters are still around, 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 small spaceship!

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. Once on board, you meet the astronaut team led by Commander Amelia Brand, Pilot Rheya Cooper, and including you and ten other astronauts. The spacecraft carries its own powerful datacenter. 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).

All characters and storylines are fictitious, and purely intended to keep the reading entertaining; these are not intended to be educational. Any resemblance to persons, places, animals, things, or events, living or dead, past, present, or future, is purely coincidental.

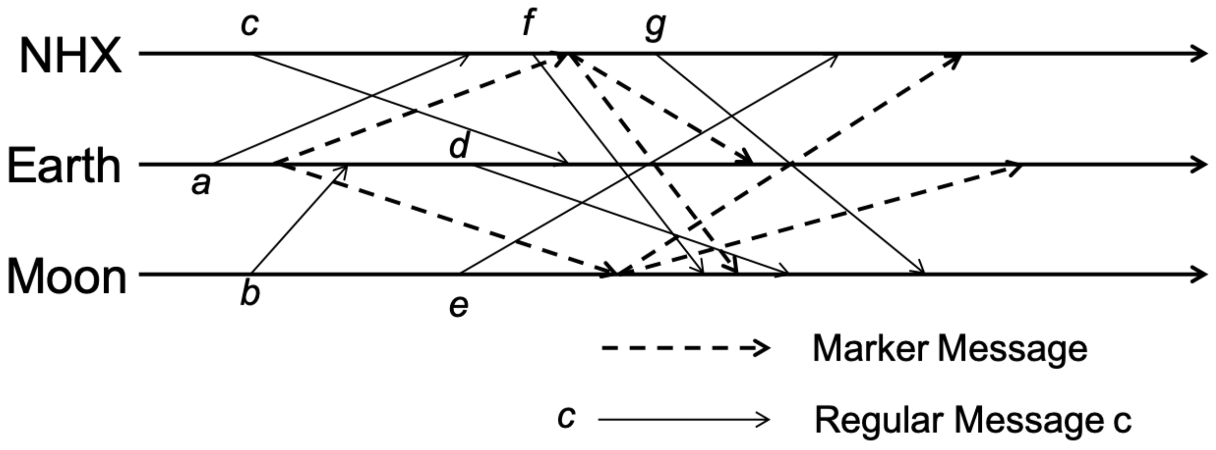
Problems:

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=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. You have detachment from the rocket! Suddenly you see your pet cat (who you had named "Doraemon" when you adopted her) in the corridor of the spaceship—you try to follow her but she disappears. You wonder if it's your imagination. Anyway, bigger cats to catch… 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). If these stations use the FIFO Ordering algorithm, 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 #3). Can you redo it quickly before your spacecraft crashes? Again, mention clearly all timestamps and all buffered messages.
4. Just this morning you also saw your pet dog (whom you had named "Einstein" when you adopted him on Earth) roaming inside the spaceship's corridor. You called out to him by name and he stared at you, but then ran away. You are perplexed, and you talk to the captain of your spaceship, Commander Amelia Brand. She asks you to take some rest. But there are miles to go before you sleep… To fix the consensus algorithm, one of your fellow astronauts has written a variant of the stock implementation of Paxos. You realize there is a bug. 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 5/6th-s of the processes (everywhere in the protocol where a quorum is needed). You need to answer three questions:
    i. Is this new version safe?
    ii. Is this new version (eventually) live?
    iii. Is this new version faster or slower than the majority (just greater than 50%) version of the protocol? Why?
5. 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. In the figure, a, b, c, … are regular application messages. You can use S(a) to denote the send event of a and R(a) to denote its receipt event. Markers

shown as dotted lines. Can you help the intern find the snapshot recorded by this run? Don't forget to include both process states and channel states. For process states, you can use the name of the latest event at that process (For initial state, just say "Initial state". Note that Markers don't count as events). Quick, it's up to you to manually calculate the snapshot!



6. You're still doing well physically and emotionally in this long trip, mostly because you were trained well at Illinois. You're about halfway through the trip to Mars. As you're retiring to your room to sleep, you see both your cat Doraemon and dog Einstein walking together in the spaceship corridor. You call out to them, but they run away again. Before you can chase them, you notice the spacecraft wobbling quite a bit, and you need to fix this. You trace the wobbling problem to the on-board storage system, and the fact that there is no leader election algorithm in there! Quick, you need to design one!
   The datacenter onboard (with hundreds of machines) uses a ring-based DHT (among the machines) with a Chord-like routing algorithm with each peer maintaining 3 ring successors and 3 ring predecessors. This system needs to elect a leader that has the **highest** DHT Id in the system.
   i. Design a leader election protocol that is efficient in that it uses very few messages (O(1) per participant). The only messages you can use are the DHT routing messages.
   ii. Argue briefly why your algorithm satisfies safety and liveness when finer tables are all correct and there are no failures during execution (proof not needed).
   iii. What is the completion time and number of messages in your leader election protocol (both asymptotic)?

iv. Discuss briefly what might happen if failures occur during the election run, while finger tables stay inconsistent.

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. And a dog can be heard yelping and a cat can be heard welping in agony, somewhere inside the spacecraft. You talk to Commander Brand and your colleague on board Pilot Rheya Cooper about this, and they counsel you that the animals are your imagination. Anyway, back to work… You quickly figure out that the alarms are because of 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 instead of using the usual and boring (Lamport timestamp, process id) pair, the algorithm instead uses (Lamport timestamp, FIFO local sequence number) pair, where FIFO local sequence number is the local sequence number of that event at that process. The rest of the Ricart-Agrawala algorithm remains unchanged. Your fellow astronaut says this algorithm, even without failures: a) violates safety, b) violates liveness, and c) does not satisfy causal ordering. Is he right on any of these counts (which ones)? Give a proof or counter-example.

9. Well you fixed one, but now you've uncovered another bug! Alarms are still 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 *1* process may obtain write access to the file simultaneously. At most *k* processes may obtain read access to the file simultaneously. 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). Hint: Token message can contain writable fields.
b. Argue briefly that your algorithm guarantees all the Safety clauses. (a formal proof is not required, however you are free to write one).
c. Can your algorithm livelock, i.e., violate Liveness? Suggest an idea to address this issue. Argue that your idea reduces the frequency of livelocks (you don't need to prove Liveness).
d. Given one process that is currently writing, and *k* processes waiting to read, what is the synchronization delay (i.e., time for *all* the *k* processes to start reading, i.e., the last one)? Calculate both best case and worst case. Since this calculation may be hard, you can assume for simplicity that: i) reads take quite long (i.e., tokens are not released by readers until everyone has started reading), ii) tokens *cannot* be combined into one message, and iii) in one time unit, only one message can be transmitted, anywhere in the system (the latter means that it suffices to calculate the total number of token transfers for the synchronization delay). Also N >> *k.* If you need to make other simplifying assumptions, be reasonable and specify them clearly. Show all your calculations.

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! You're no longer seeing your dog Einstein and cat Doraemon (though you kinda hear them sometimes, which makes you question your own sanity). You notice that there are fewer astronauts up in the command center of the spacecraft—you say to yourself they're all probably resting up for the landing.

    Anyway, 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 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 concurrently p2 multicasts message M45. p1, p2, and p3 all deliver M32 and M45 each. Then p1 and p2 deliver the next view V12={p1,p2}; when p3 receives this view, it delivers the view V12.
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.

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, all your fellow astronauts, and Commander Amelia Brand and Pilot Cooper, have unanimously decided to give you the honor of being the first human 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 *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 each 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
```

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!).

--- (HW2 Official End) ---

<u>Epilogue (If you'd like to not spoil surprises, don't read this epilogue until you've read the Prologue and even-numbered questions above)</u>: As you take humankind's first steps on Mars, you look back at the Horizons X lander spacecraft. You see your dog Einstein and cat Doraemon together peering down at you through the porthole window. You remember they are indeed real, and that you did indeed bring them along with you from Earth! The long trip and cryogenic sleep made you forgetful! You realize that Einstein and Doraemon were just too disoriented by the space travel experience, and that's why they kept running away from you. It all makes sense now!

The Earth station, from millions of miles away, speaks in your earpiece, "Congratulations! You just completed the first solo human mission to Mars! Woohoo!" You're happy, but then you stop and ask Earth station, "Solo?! What about the other ten astronauts? What about Commander Amelia Brand and Pilot Rheya Cooper who were with me?" There is a pause. Earth station responds, "Ten astronauts? Brand and Cooper…? Who…?"

"Just kidding!" says Earth station, following up with, "Brand, Cooper, and other astronauts were lifelike holograms we created to keep you company. On this long journey to Mars, without human company, we knew you'd go mad!... By the way, from their vital signs, we see your pet dog Einstein and cat Doraemon are finally feeling themselves. Say Hi to them for us!"

--- The End ---

(PS: Did you catch all the sci-fi references in this homework?)