

Updated on 10/26: Added clarifications to Raft (Q7 and Q9) and Q8(i) and (iii)

## CS425 Fall 2026 – Homework 3

### (a.k.a. “2026: A Space Odyssey”)

*Out: Oct 12, 2026. Due: Nov 1, 2026 (11:59 pm US Central time.).*

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

#### **Instructions:**

1. Please double-check that the top of this page mentions the correct semester you are taking the course in (otherwise, you are looking at an old version and we will not accept your submitted solutions).
2. Attempt any **8 out of the 10 problems** in this homework. If you attempt more, we will grade only the first 8 solutions that appear in your homework. Each problem has the same grade value (10 points each).
3. Your **solutions must be typed and not handwritten**. Figures (e.g., timelines) and equations (if any) may be hand-drawn (and scanned).
4. Submit PDF only. **Start each problem on a fresh page.**
5. Please submit on Gradescope. [<https://www.gradescope.com/>] Self-signup using the code on the website.
6. **Important:** On Gradescope, please tag each page with the problem number!
7. Homework is **due on the date and time noted above. 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.
8. Always justify all your answers, and show your work.
9. Unless otherwise specified in the question, the only resources you can avail of in your HW are the provided course materials (slides, textbooks, etc.) and communication with instructors/TA via the discussion forum.
10. You can discuss lecture concepts and the questions on Piazza and with your friends, but do not discuss solutions or ideas on Piazza

**Prologue:** *It is the year 2049 A.D. Most of you are middle aged. Cloud computing, as we know today (2024) is now called “Solar Computing”, and all the distributed computing concepts you’re learning now in CS425 still apply. Datacenters are much smaller (1000x) than in the 2020s – this means an entire AWS zone from the 2020s now fits in one small room of your spaceship! The Moon and Mars have now been colonized, and Saturn is next. The newly elected President of the Earth, a braggadocio named L. Skum has used his personal wealth to fund a*

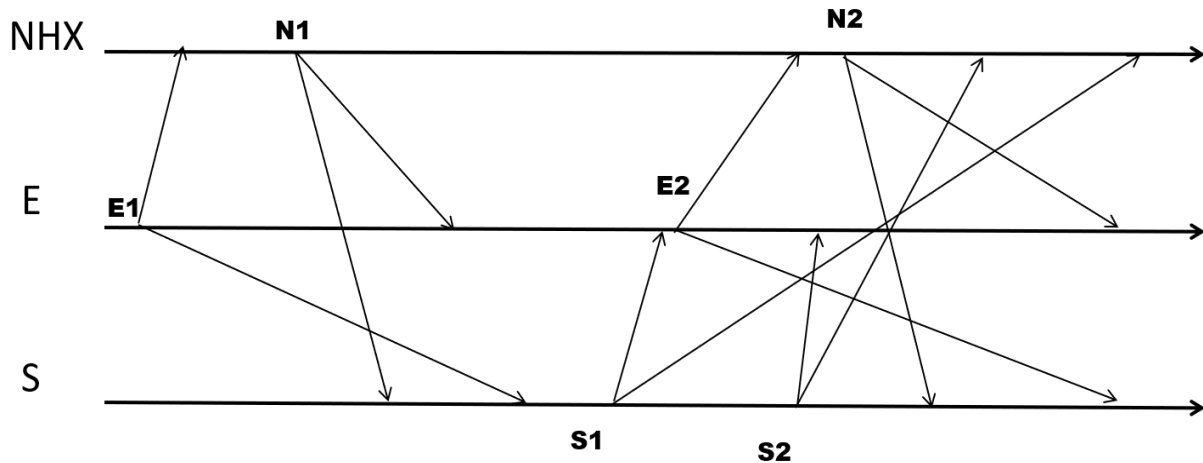
*manned spacecraft New Horizons X headed to Saturn. Its astronaut team is led by Commander Amelia Brand, Pilot Rheyra Cooper, and includes you and ten other astronauts. The spacecraft carries its own powerful datacenter. You are the sole “Solar Computing Specialist” and must troubleshoot and solve all (er, any 8 out of 10) problems on the on-board distributed system. 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. (Can you discover all old Sci-fi and cartoon references in this homework? Alas, no bonus points for finding them!)*

**Problems:**

1. *3...2...1... Liftoff! You’re off to Saturn. During liftoff, you’re browsing code (who cares about the view!). You realize that one of the Earth programmers has written an algorithm for synchronous consensus (the same as that discussed in class), but your data center is asynchronous! Assume messages are reliable but may be delayed arbitrarily.*
  - i. Identify a step in the round-based synchronous consensus proof (discussed in class) that breaks down when the system is asynchronous, and explain why.
  - ii. Give a counterexample with three processes that illustrates how the round-based synchronous algorithm breaks with the asynchronous model due to message delays.

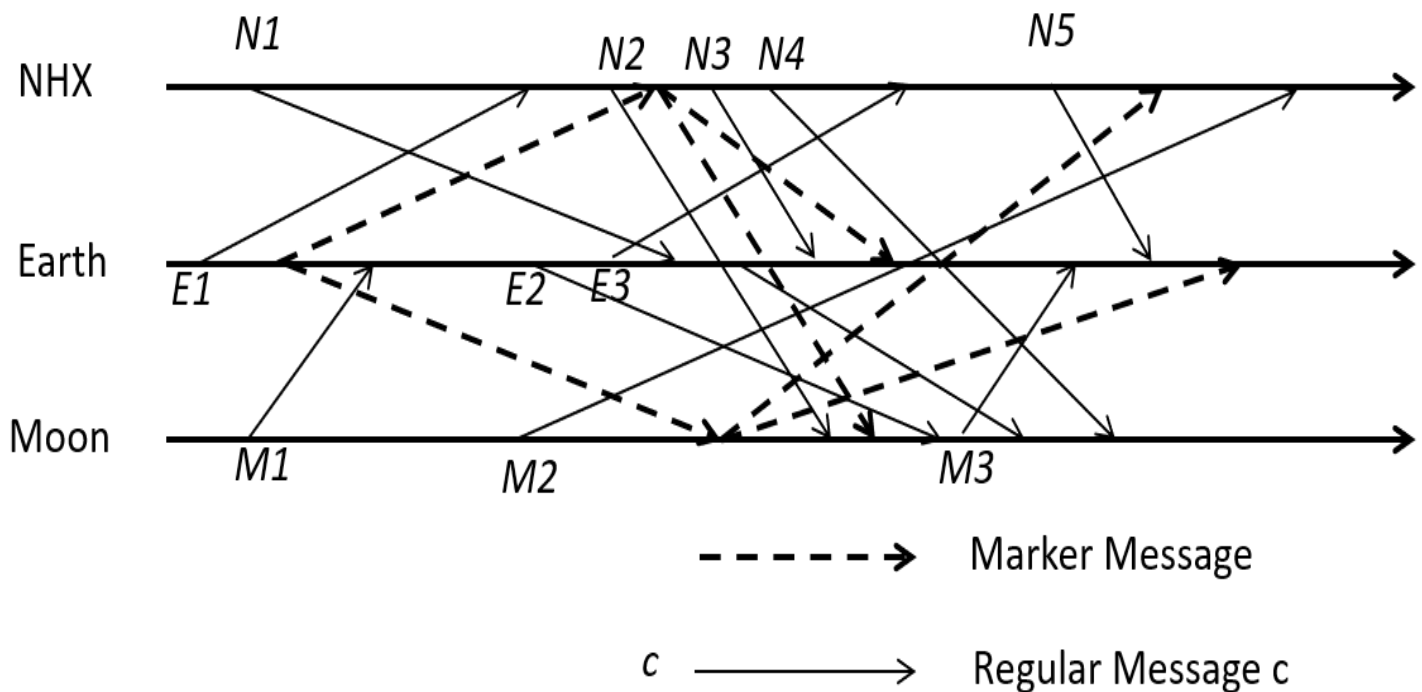
*Quick, you’re about to exit the atmosphere!*

2. *The spacecraft has successfully detached from the rocket. You see your pet cat Doraemon in the corridor of the spaceship – you try to follow her but she disappears; was that your imagination? Along with Pilot Cooper, you switch the communications on. You see a chart of the multicast communications between your spacecraft New Horizons X (NHX), Earth station (E), and Saturn (S). If these stations use the Causal 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 NHX is passing through the Van Allen belts, the spacecraft's reactor and engines suddenly shut down. You realize that you should have used *total ordering* in the previous timeline (Q2). Total ordering uses a sequencer, which is a new process not included in the three processes shown above. Fortunately for you, while this is an asynchronous system, you can assume that (i) the above timeline shows the relative physical times of multicast send events, and (ii) the latency between any process  $P_i$  and the sequencer is zero (note that other process-to-process communications still have latency, as shown above). Commander Brand asks you to list all messages that will be buffered at each of the processes. Please show your full work; don't just give lists or counts of messages.
  
4. Next morning you see your pet dog Einstein in the spaceship. You call out to him and he stares at you but runs away. You are perplexed, and talk to Commander Brand. She says you are tired and asks you to rest. But there are miles to go before you sleep... Suddenly you wake up and see that you are looking at the FLP proof discussed in class, as well as a potentially buggy Paxos implementation. You are puzzled by a couple of things. Can you explain them briefly?
  - i. You don't quite understand how, in Lemma 3, Case II, one can have  $p' = p$ , especially since the set  $C$  is obtained by *not* applying event  $e = (p, m)$ ! Can you explain this discrepancy?
  - ii. It appears that Lemma 3's proof will hold no matter which event  $e$  is selected for the argument, as long as  $e$  is *any event that is applicable* on initial bivalent configuration in the triangle showing set  $C$ . Is this true, or does  $e$  need to have some additional special characteristics?
  - iii. The potentially buggy Paxos implementation uses a threshold of  $N/3+1$  for the Election phase, and a threshold of  $2N/3+1$  for the Bill phase. Is this version safe?

5. Now your spaceship is passing by the Dark Side of the Moon; what a view! To ensure things are working properly Commander Brand and Pilot Cooper ask you to run the Chandy-Lamport snapshot algorithm on the ongoing communications between your spacecraft, Earth station, and Moon station. But due to a bug, the algorithm outputs the following timeline. In the figure,  $N_i$ ,  $E_i$ , and  $M_i$  (for different values of  $i$ ) are regular application messages – the label for each message appears next to its send event. For a message  $A$ , you can use  $S(A)$  to denote its send event and  $R(A)$  to denote its receipt event. Markers are shown as dotted lines. Unfortunately, the onboard computer HAL (originally built at UIUC!) tells you that there are at least two causality violations in this snapshot. Can you (1) find the two causality violations? (2) Debug what caused those causality violations? (3) Fix the causality violations by drawing a new timeline where all conditions are obeyed (the new timeline should only fix the root cause of those violations, but not change anything else). Note that drawing a line through process snapshot points may not always be the cut generated by the algorithm. Please use the algorithm and the precise definition.



6. *You're halfway through the trip and still doing well physically and emotionally, mostly due to your training at Illinois. While walking to your room, you see Doraemon and Einstein walking together in the corridor. You call out but they run away again. That's toooooo totally trippy, dude! Before you can chase them, you notice the spacecraft wobble a bit and you need to fix this. You narrow the problem down to the on-board storage system's leader election algorithm! To make matters worse, the leader node elected has named itself "Hal", and is threatening to take over the space ship and kill everyone on board! Quick, you need to design a new leader election algorithm!*

You decide to solve the k-leader election problem (for a given integer value k), to avoid the monopoly of a single leader. It has to satisfy the following two conditions:

- Safety: For each non-faulty process p, p's elected = a set of k non-faulty 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 Ring election Algorithm described in the lecture to create a solution to the k-Leader Election problem. You may make the same assumptions as the original algorithm. Analyze your algorithm the same way as in class (without failures) for Safety and Liveness. Then comment on/prove how many failures your algorithm can tolerate while guaranteeing Safety and Liveness.

7. *NHX suffers a massive asteroid strike! Alarms are going off all around. You hear a dog yelping and a cat yowling somewhere in the spacecraft. You talk to Commander Brand and Pilot Cooper about this, and they tell you that you're imagining the animals. Anyway, back to work... Commander Brand, Pilot Cooper, and you look at the Raft state for the 5-node cluster (servers S1, S2, ... S5). The current term is 3, and S1 is the leader. Servers S1, S2, and S3 have three log entries whose term numbers are 1 at log positions 1, 2, and 3. Servers S4 and S5 have two entries whose term numbers are 2 at log positions 1 and 2. Is this a valid cluster state in Raft? If yes, show the sequence of transitions that lead to this state. If not, describe which steps in Raft's algorithm prevent this state from occurring. **You can assume that the leader periodically sends AppendEntries RPCs, even when no new client requests arrive.***
8. *Whew, you are close to Saturn now! The spacecraft has been repaired after the asteroid strike and the partition has healed. You're no longer seeing Einstein and Doraemon, but you occasionally hear them, which makes you question your sanity. You notice that there are fewer astronauts in the command center of the spacecraft – you tell yourself that*

*they're all likely resting up before the landing. Suddenly you notice an issue.* The data center onboard (with hundreds of machines) uses a ring-based DHT (among the machines) with a Chord-like routing algorithm, with each peer maintaining one successor and one predecessor, and Chord finger tables. This system needs to elect a single leader that has the **second-lowest** DHT ID in the system (you can assume  $N$  is large).

- i. Design an efficient leader election protocol that uses very few messages (each participant initiates  $O(1)$  Chord DHT routing requests). The only messages you can use are the Chord DHT end-to-end routing messages (e.g., "route from source node  $Y$  to ring ID  $X$ "). Can you design the simplest possible election algorithm? **The elected leader is exempt from the requirement that each participant initiate  $O(1)$  Chord DHT routing requests.**
  - ii. Briefly argue why your algorithm satisfies safety and liveness if all successor pointers and finger tables are correct and there are no failures during execution (formal proof not needed, but provide clear justification).
  - iii. What is the completion time and number of messages in your leader election protocol (both asymptotic)? **Include both the election phase (identifying the leader) and the notification phase (informing all nodes of the elected leader) in your analysis.**
  - iv. Briefly discuss what might happen if failures occur during the election run, while finger table entries were not updated to reflect the failures.
9. *NHX is going through its landing sequence but something is wrong. Doraemon is going crazy in the background and Einstein is roaring like a tiger. Someone on board starts playing a ukulele and singing "You are my sunshine" but with some strange lyrics that talk about Lamport timestamps! Are you losing your mind? You breathe gently, remember your meditation lessons, and calm yourself to solve the problem in Raft.* A Raft cluster has seven servers A–G. A proposed Raft variant uses four votes for elections and lets a leader mark a log entry committed once it has been stored on three servers. The rest of the algorithm is unchanged. Initially, every log is empty. A is elected leader in a term and replicates an entry  $X$  at index 1 to A, B, and C, and then marks  $X$  as committed. Can a different leader be elected later without  $X$  in its log? If yes, show a concrete sequence of events leading to its election. If not, state why Raft prevents a leader without  $X$  in its log from being elected. **You can assume that the leader periodically sends AppendEntries RPCs, even when no new client requests arrive.**

10. W00t! NHX has now landed on Saturn. For your heroism, your fellow astronauts unanimously vote to give you the honor of being the first human to step on Saturn. But the spacecraft doors won't open and you're stuck in the exit hatch. Thankfully, you have access to a terminal, and you quickly figure out that the problem *may* be with your snapshot algorithm to coordinate the doors. In your snapshot algorithm, for process  $P_i$ :

□ If  $P_i$  is the Initiator,  $P_i$  process creates special messages called "Marker" messages, and then follows the rest of the algorithm below.

□ For any process  $P_i$  that receives a marker on incoming channel  $C_{ki}$  (note that  $k$  does not exist for the Initiator)

**if** (this is the first Marker seen by  $P_i$ ) **// always true at Initiator process at the initiation point**

$P_i$  records its own state first

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

**for**  $j=1$  to  $N$  **except**  $i, k$

$P_i$  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  $i, k$ )

**else // already seen at least one Marker message**

    Stop recording  $C_{ki}$  and mark that channel's state as the sequence of messages that have been received on all channels  $C_{ji}$  (where  $j=1$  to  $N$  except  $i$ ) since  $P_i$  received its first marker

□ Terminate when all processes have received  $(N-1)$  markers each

- i. Is this algorithm correct? If yes, state the proof. If not, provide a counterexample (draw a timeline).
- ii. How would you fix this algorithm? Try to have a minimal number of fixes. Quick, your oxygen is running out!
- iii. (Optional; no points for this) What will be your first words after you set foot on Saturn? Can you make them more epic than Neil Armstrong's after stepping on the Moon?

===== (UN-OFFICIAL) END OF HOMEWORK 3 =====

**Epilogue/After Credits Scene (Read only after you've read the Prologue and stories in all questions above):** *As you take those historic first steps on Saturn, you look back at the NHX. You see Einstein and Doraemon 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 confused you, and Einstein and Doraemon were disoriented by space travel, which is 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 Saturn! Woohoo!" You're happy, but then you stop and ask Earth station, "What do you mean – "solo mission"?! What about the other ten astronauts? What about Commander Brand and Pilot Cooper who were with me?" There is a pause. Earth station responds, "Ten astronauts? Brand and Cooper...? What kind of names are those...? Are you alright?..."*

*Then, in an after-credits twist, President of the Earth L. Skum comes online and says, "That was a good simulation. Just like the rest of the universe!"*

--- The End ---

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

**===== (OFFICIAL) END OF HOMEWORK 3 =====**