EE374 Internet-scale Consensus **Stanford, Spring 2025**

Homework #4 Solution

Due: Friday, May-02-2025, 11:59pm – Gradescope entry code: `R57ZN7`

Please upload your answers timely to Gradescope. Start a new page for every problem. We strongly suggest LaTeX to type your answers. For the programming/simulation questions you can use any reasonable programming language (please no assembly, brainfuck, etc. ☺). Comment your source code and include the code and a brief overall explanation with your answers. A tentative point distribution (in % of the total) is provided in brackets. For most problems there is more than one valid way of solving them!

1. In this question, we consider a version of Tendermint, where the nodes, having acquired a lock, never release that lock. Suppose there are f adversarial nodes (Byzantine adversary) and $n = 3f+1$ nodes in total, and the quorum size is $q = 2f+1$. We denote a client by the letter c, honest nodes by $v_1^h, v_2^h, \ldots, v_{2f+1}^h$, and the adversarial nodes by $v_1^a, \ldots, v_f^a$ (the clients and honest nodes do not know which nodes are honest or adversarial). This version of Tendermint (just like the original Tendermint protocol) attempts to confirm a single block per height. Within each height h, it proceeds in *rounds*, each with a unique, known leader that proposes a block. Each of these round attempts to confirm a block for the height h.

   At each round r, each honest node keeps track of the *stage* of the protocol within the round. It can be one of proposal, pre-vote, or pre-commit. Each stage lasts Δ time, and thus, each round lasts 3Δ time.

   At the beginning of the proposal stage (time $t = 0$), an honest leader proposes a round-r block. At the beginning of the pre-vote stage (time $t = \Delta$), each honest node sends a round-r pre-vote for the first round-r block it observes. If an honest node does not observe a round-r block, it sends a round-r pre-vote for a special *nil* (empty) value. At the beginning of the pre-commit stage (time $t = 2\Delta$), each honest node sends a round-r pre-commit for the round-r block, for which it has first observed $2f + 1$ round-r pre-votes by distinct nodes. If an honest node does not observe $2f + 1$ pre-votes for any round-r block, it sends a round-r pre-commit for a special *nil* value. At the end of the round ($t = 3\Delta$) or later, if an honest node observes $2f + 1$ round-r pre-commits for B, it confirms B for its height (h), and terminates the protocol for height h. Otherwise, it goes into the next round $r + 1$.

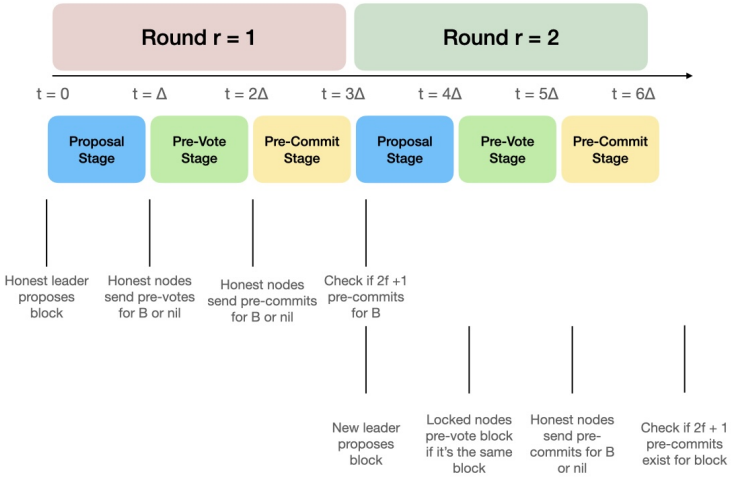
   A client c confirms a round-r block B, if it observes (at any time) $2f + 1$ round-r pre-commits for B by distinct nodes. In that case, we say that the round-r block B became confirmed by the round-r pre-commits.

   An honest node locks on a round-r block B at round r upon sending a round-r pre-

commit for B. In future rounds, the node does not send pre-votes for other blocks $B' \neq B$. If it is elected as a leader in a future round, it proposes the same block it is locked on.

a) (%10) Draw the time-line for an execution of the protocol that lasts for two rounds, illustrating all the stages it has gone through.

Answer:



---

b) (%10) We first explore if this version of Tendermint is safe. Suppose two different blocks B and B' are proposed for the *same* round r. Is it possible for one client c to confirm B, while another client c' confirms B'? (Here, c and c' can also be the same client.)

Answer:

We prove by contradiction that this is not possible. Suppose c confirms B, while c' confirms B'. This implies that c observes $2f + 1$ or more round-r pre-commits for B by distinct nodes, while c' observes $2f + 1$ or more round-r pre-commits for B' by distinct nodes. Since there are $3f + 1$ nodes in total, at least $f + 1$ nodes must have sent round-r pre-commits for both B and B', i.e., at least one honest node must have sent round-r pre-commits for both B and B'. This is a contradiction, since an honest node sends a single pre-commit per round (either for a block or for the *nil* value).

c) (%10) Suppose a client c confirms a block B proposed for round r. Does this say anything about how many honest nodes must have locked on B at round r?

Answer:

Since c confirms the round-r block B, c must have observed $2f+1$ or more round-r pre-commits for B by distinct nodes. Now, out of the $2f+1$ (or more) round-r pre-commits for B, at least $f+1$ are by distinct honest nodes, which sent these pre-commits for B at round r. Therefore, at least $f+1$ honest nodes must have locked on block B at round r.

d) (%10) Now, suppose two different blocks B and B' are proposed for *different* rounds r and $r' > r$ respectively. Is it possible for one client c to confirm B, proposed for round r, while another client c' confirms B', proposed for round r'? (Here, c and c' can also be the same client.)

Answer:

We prove by contradiction that this is not possible. Since c confirms the round-r block B, by part (b), we know that at least $f+1$ honest nodes must have locked on block B at round r.

Now, as c' confirms the round-r' block B', c' must have observed $2f+1$ round-r' pre-commits for B'. Out of the $2f+1$ round-r' pre-commits for B', at least $f+1$ are by distinct honest nodes, which sent pre-commits for B' at round r'. For an honest node to send a round-r' pre-commit for B', it must have observed $2f+1$ round r' pre-votes at round r' for B'. However, since there are at most $2f+1$ honest nodes in total, there must be at least one honest node that have locked on block B at round r, and yet, sent a round-r' pre-vote for $B' \neq B$ at round $r' > r$. This is a contradiction.

e) (%10) Is this protocol safe? If so, prove safety referring to the parts above. If not, describe an attack on safety referring to these parts.

Answer:

Yes! The protocol is safe; since it is not possible for more than one block to become confirmed in the clients' view, whether these blocks are for the same or different rounds, by parts (a) and (c).

---

f) (%10) We next explore if the protocol is live. First, suppose at round 1, the leader is honest. Will a block become confirmed?

Answer:

Yes, the block proposed by the leader will gather $2f + 1$ pre-votes and pre-commits by the honest nodes, and become confirmed.

---

g) (%10) Now suppose at some round r, there is an adversarial leader, which proposes a block B_1, but it initially shows B_1 to only the $f + 1$ honest nodes $v_1^h, \ldots, v_{f+1}^h$, right before the pre-vote deadline. As a result, out of the $2f + 1$ nodes, only these $f+1$ honest nodes send round-r pre-votes for B_1. Now, right before the pre-commit deadline, the adversary shows f round-r pre-votes for B_1 by the adversarial nodes $v_1^a, \ldots v_f^a$, *only* to a single honest node, v_1^h. In this case, does block B_1 become confirmed by the round-r pre-commits? Which honest nodes send round-r pre-commits for B_1? Which honest nodes lock on B_1 at round r?

Answer:

B_1 cannot become confirmed by the round-r pre-commits, since out of the honest nodes, only v_1^h sends a round-r pre-commit for it, implying that there can be at most $f + 1$ round-r pre-commits for B_1. Out of the honest nodes, only v_1^h sends a round-r pre-commit for B_1 and locks on B_1 at round r.

---

h) (%10) At round $r + 1$, the honest node v_2^h becomes the leader and proposes a block $B_2 \neq B_1$. The adversarial nodes do not send any pre-votes or pre-commits at round $r + 1$. Does B_2 become confirmed by the round-$(r + 1)$ pre-commits? Which honest nodes send round-$(r + 1)$ pre-votes for B_2? Which honest nodes send round-$(r + 1)$ pre-commits for B_2? Which honest nodes lock on B_2 at round $r + 1$?

Answer:

No, B_2 cannot become confirmed by the round-$(r+1)$ pre-commits. Node v_1^h became locked on B_1 at round r; so it will not send a round-$(r + 1)$ pre-vote for $B_2 \neq B_1$.

As a result, there will be $2f$ round-$(r+1)$ pre-votes for B_2 (adversarial nodes are silent), and thus, no honest node will send a round-$(r+1)$ pre-commit for B_2. Now, all honest nodes except v_1^h send round-$(r+1)$ pre-votes for B_2, but no honest node sends a round-$(r+1)$ pre-commit for B_2, or becomes locked on B_2 at round $r+1$.

---

i) (%10) Now, suppose that instead of remaining silent, the adversary shows a single round-$(r+1)$ pre-vote (by an adversarial node) for B_2 to a single honest node, v_2^h, right before the pre-commit stage starts. Does B_2 become confirmed by the round-$(r+1)$ pre-commits? Which honest nodes send round-$(r+1)$ pre-votes for B_2? Which honest nodes send round-$(r+1)$ pre-commits for B_2? Which honest nodes lock on B_2 at round $r+1$?

Answer:

No, B_2 still cannot become confirmed by the round-$(r+1)$ pre-commits. As in part (f), all honest nodes except v_2^h still observe $2f$ round-$(r+1)$ pre-votes for B_2 and do not send round-$(r+1)$ pre-commits for B_2. However, v_2^h now observes $2f+1$ round-$(r+1)$ pre-votes for B_2 and *sends* a round-$(r+1)$ pre-commit for B_2. Block B_2 cannot become confirmed with a single round-$(r+1)$ pre-commit.

In summary, all honest nodes except v_1^h send round-$(r+1)$ pre-votes for B_2, but only v_2^h sends a round-$(r+1)$ pre-commit for B_2, and only v_2^h becomes locked on B_2 at round $r+1$.

---

j) (%10) Suppose the adversary remains silent after round $r+1$. Is this protocol live? If so, prove liveness referring to the parts above. If not, describe an attack on liveness referring to these parts.

Answer:

No! We observe that the adversary essentially made v_1^h and v_2^h locked on different blocks B_1 and B_2. Now, in future rounds, whenever a block other than B_1 is proposed, v_1^h will not send a pre-vote. Similarly, whenever a block other than B_2 is proposed, v_2^h will not send a pre-vote. Thus, no block will ever get $2f+1$ pre-votes or pre-commits at round r, $r+1$ or any future round, implying a lack of liveness.
