EE374 Blockchain Infrastructure **Stanford, Spring 2025**

Homework #5 Solution

Due: Friday, May-09-2025, 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 explore if Tendermint can be made accountably-safe. Suppose there are $n = 3f + 1$ nodes in total, and the quorum size is $q = 2f + 1$. **We do not apriori assume any fraction of the nodes are honest! So f here is just a parameter, not the number of adversarial nodes.** We denote a client by the letter c, and the nodes by the letter v.

   Recall that Tendermint 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 rounds attempts to confirm a block for the height h. Round structure of Tendermint is described below:

   ***

   At each round $r = 0, 1, 2, \ldots$, each honest node keeps track of the *step* of the protocol within the round. It can be one of proposal, pre-vote, or pre-commit. Each step lasts Δ time, and thus, each round lasts 3Δ time.

   At the beginning of the **proposal step** (time $t = 3\Delta r$), an honest leader proposes a round-r block B by sending a proposal message $\langle \mathsf{Proposal}, r, vr, B \rangle$. We will later see what vr refers to – it can be -1 or a positive integer denoting some round.

   Let $\langle \mathsf{Proposal}, r, vr, B \rangle$ be the first round-r proposal message observed by an honest node. At the beginning of the **pre-vote step** (time $t = 3\Delta r + \Delta$), this honest node sends a round-r pre-vote, denoted by $\langle \mathsf{Prevote}, r, vr, h(B) \rangle$, for block B (vr is copied from the proposal message) if the following conditions are satisfied:

   - When $vr = -1$, the node can directly send the pre-vote.
   - When $vr > 0$, the honest node will send the pre-vote only if it has also observed $2f + 1$ round-vr pre-votes for B, i.e., $2f + 1$ messages of the sort $\langle \mathsf{Prevote}, vr, vr', h(B) \rangle$ (here, vr' does not matter).

If an honest node does not observe a round-r block it can vote for, it sends a round-r pre-vote, denoted as $\langle \mathsf{Prevote}, r, vr = -1, \perp \rangle$, for a special *nil* (empty) value.

At the beginning of the **pre-commit step** (time $t = 3\Delta r + 2\Delta$), each honest node sends a round-r pre-commit, denoted by $\langle \mathsf{Precommit}, r, h(B) \rangle$, for the round-$r$ block B, for which it has first observed $2f + 1$ round-r pre-votes of the form $\langle \mathsf{Prevote}, r, vr, h(B) \rangle$ by distinct nodes. If an honest node does not observe such $2f + 1$ pre-votes for any round-r block, it sends a round-r pre-commit, denoted as $\langle \mathsf{Precommit}, r, \perp \rangle$, for a special *nil* value.

Confirmation Rule: At the end of the round ($t = 3\Delta r + 3\Delta$) 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.

Proposal Rule: Let r' be the largest round such that the node has observed $2f + 1$ round-r' pre-votes for a round-r' block B'. Then, if that node is elected as a leader in a future round r'', it proposes this block B' for round r'' by sending the message $\langle \mathsf{Proposal}, r'', vr = r', B' \rangle$. If there is no such round r', i.e., if the node has not observed $2f + 1$ (same-round) pre-votes for any block B', it can propose any block B'' by sending the message $\langle \mathsf{Proposal}, r'', vr = -1, B'' \rangle$.

Locking Rule: An honest node v locks on a round-r block B at round r upon sending a round-r pre-commit for B. In a future round r'', the node v does not send pre-votes for other blocks $B'' \neq B$ **unless the block B'' comes as part of a proposal** $\langle \mathsf{Proposal}, r'', vr = r', B'' \rangle$ **and** v **observes** $2f + 1$ **round** $vr = r'$ **pre-votes for** B''. *Note that an honest node never releases a lock permenantly: it can acquire a lock at a higher round in the future, or it might temporarily drop the lock to vote for block $B'' \neq B$ in the example above.*

***

a) (%30) Suppose two different blocks B_0 and B_1 from rounds 0 and 1 are confirmed by clients c_0 and c_1 respectively. Re-do the attack from Lecture 10, where we saw that the clients cannot identify the nodes that violated the protocol rules.

Answer:

Since c_0 confirmed B_0 and c_1 confirmed B_1, c_0 must have observed $2f + 1$ round-0 pre-commits for B_0, and c_1 must have observed $2f + 1$ round-1 pre-commits for B_1. The $2f + 1$ nodes that sent the round-0 pre-commits for B_0 must have locked on B_0 at round 0 (**locking rule**). Thus, these $2f + 1$ nodes that locked on B_0 should

never send a pre-vote for any block other than B_0 at round 1! However, they are allowed to send a round-1 pre-commit for a different block, namely B_1, if they do observe $2f + 1$ round-1 pre-votes for B_1. When there are $2f + 1$ round-1 pre-votes for B_1 by quorum intersection, at least $f + 1$ nodes must have locked on B_0 at round 0, yet sent a round-1 pre-vote for $B_1 \neq B_0$, which sure enough is a protocol violation. However, the clients c_0 and c_1 cannot identify these $f + 1$ nodes, since they do not necessarily observe the round-1 pre-votes.

---

b) (%35) To mitigate the attack, suppose we modify the confirmation rule of Tendermint as follows (we stick with the new confirmation rule throughout this homework):

***

New Confirmation Rule: At the end of the round ($t = 3\Delta r + 3\Delta$) or later, if an honest node observes $2f + 1$ **round-r pre-votes and** $2f + 1$ 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-votes and** $2f + 1$ 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.

Note that the *certificate* now consists of not only the $2f + 1$ pre-commits, but also the $2f + 1$ pre-votes.

***

Again, two clients c_0 and c_1 confirm blocks B_0 and $B_1 \neq B_0$ from rounds 0 and 1. Then, can c_0 and c_1 come together and identify at least $f + 1$ nodes that violated the protocol rules? If so, write down the proof. If not, argue why.

Answer:

Yes, c_0 and c_1 can come together and identify at least $f + 1$ nodes that violated the protocol rules. Note that c_0 must have observed $2f + 1$ round-0 pre-votes and pre-commits for B_0, and c_1 must have observed $2f + 1$ round-1 pre-votes and pre-commits for B_1. The $2f + 1$ nodes that sent the round-0 pre-commits for B_0 must have locked on B_0 at round 0 (**locking rule**).

Now, there is no round r between rounds 0 and 1 such that there can be $2f + 1$ round-r pre-votes for any block other than B_0. Thus, the $2f + 1$ nodes that locked on B_0 at round 0 should never send a pre-vote for any block other than B_0 at round 1! However, $2f + 1$ nodes sent round-1 pre-votes for $B_1 \neq B_0$. By quorum intersection, at least $f + 1$ nodes must have locked on B_0 at round 0, yet sent a

round-1 pre-vote for $B_1 \neq B_0$, which is a protocol violation. These $f+1$ nodes can be identified by c_0 and c_1 as protocol violators; since c_0 and c_1 did observe the round-0 pre-commits for B_0, and the round-1 pre-votes for B_1 by these $f+1$ nodes.

---

c) (%35) Now, suppose two different blocks B_0 and B_2 from rounds 0 and 2 are confirmed by clients c_0 and c_2 respectively. Block B_0 was proposed via the message $\langle \mathsf{Proposal}, r = 0, vr = -1, B_0 \rangle$, and block B_2 was proposed via the message $\langle \mathsf{Proposal}, r = 2, vr = 1, B_2 \rangle$. Then, can c_0 and c_2 come together and identify at least $f+1$ nodes that violated the protocol rules? If so, write down the proof. If not, argue why.

Answer:

The answer is no, and the attack is very similar to the one we saw in Lecture 10. Note that c_0 must have observed $2f+1$ round-0 pre-votes and pre-commits for B_0, and c_2 must have observed $2f+1$ round-2 pre-votes and pre-commits for B_2. The $2f+1$ nodes that sent the round-0 pre-commits for B_0 must have locked on B_0 at round 0. As in the answer to part (a), by a quorum intersection, at least $f+1$ nodes (denoted by the set S_2) must have locked on B_0 at round 0, yet sent a round-2 pre-vote for $B_2 \neq B_0$. These $f+1$ nodes are allowed to do so, only if they have a valid reason to send a round-2 pre-vote for B_2, i.e., only if they have observed $2f+1$ round-1 pre-votes for block B_2. Now, again by a quorum-intersection argument, at least $f+1$ nodes (denoted by the set S_1) must have locked on B_0 at round 0, yet sent a round-1 pre-vote for $B_2 \neq B_0$, which is a protocol violation. However, these latter $f+1$ nodes in the set S_1 **cannot** necessarily be identified by c_0 and c_2; since c_0 and c_2 do not necessarily observe the $2f+1$ round-1 pre-votes for B_2 (they only observe the $2f+1$ round-0 and 2 pre-votes and pre-commits).
