EE 374: Fundamentals of Blockchain Infrastructure **Stanford, Spring 2025**

Lecture 8: The Road to Tendermint

April 23, 2025

Lecturer: Prof. David Tse Scribe: Natalia Kokoromyti

Lecture Overview

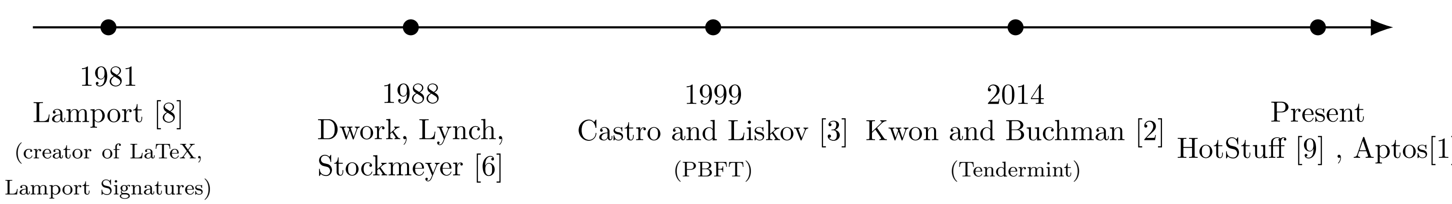
This lecture introduces a classical consensus protocol in contrast to Nakamoto-style consensus. The focus is on consensus in a proof-of-stake (PoS) setting, where participants have one vote each. The protocol is leader-based and the leader rotates dynamically. This lecture lays the groundwork for understanding deployed protocols like Tendermint by focusing on how to ensure both protocol liveness and safety.

Course Logistics

- Homework 3 due on Friday April 25, 11:59 PT.
- Take-home midterm date / time: from May 12, 4:30 pm to May 13, 4:30 pm. We have 24 hours to complete it.

1 A Brief Timeline of Consensus Protocols

- Lamport: "The Byzantine Generals Problem" (achieving consensus in a distributed system is impossible if one-third or more of the participants are malicious).
- Dwork, Lynch, Stockmeyer: consensus under partial synchrony.
- Castro and Liskov: Practical Byzantine Fault Tolerance (PBFT).
- Kwon and Buchman: deployed Tendermint, a PBFT variant for blockchain.
- HotStuff and improved linear PBFT (Facebook Libra consensus).
- Aptos: protocol variant from HotStuff which made it easier to build upon and scale web3 infrastructure.



1.1 Tendermint's Impact

We'll dive into the inner workings of Tendermint in a future lecture, yet it is important to contextualize its role in the evolution of consensus protocols. Tendermint marked a turning point by enabling modular blockchain development. As the consensus engine behind Cosmos, it introduced an SDK allowing anyone to easily launch application-specific blockchains. This modularity encouraged a surge of new projects and led to rapid progress. Several years after Tendermint launched, Ethereum transitioned to proof-of-stake. While both aimed to improve scalability and security, they took different architectural approaches.

2 Fundamentals: Proof-of-Stake Consensus

2.1 PKI and Message Authentication

Public-key infrastructure (PKI) ensures all messages can be authenticated in the network. Each of the n nodes possesses exactly one key pair, denoted by (pk_i, sk_i) for $i = 1, \ldots, n$. This mechanism differs from Nakamoto's protocol, where block proposal eligibility is determined by hash power rather than ownership of coins.
Note that in PoS systems, coins change hands over time as ownership is determined by the consensus protocols. Here, however, we assume that coin ownership is fixed. The more general case will be discussed in a future lecture.

2.2 Height-by-Height Consensus

In height-by-height consensus, consensus happens at each block height independently. Nodes agree on one block per height through rounds of proposal and voting. Unlike Nakamoto consensus, where blocks implicitly vote for earlier blocks, here consensus is reached explicitly through rounds of votes *per height*. That is, it's not blocks confirming blocks, but votes confirming blocks. This structure forms the foundation for protocols like Tendermint and PBFT.

2.3 Basic Protocol (Ideal Setting)

- Denote by Δ the network delay.
- At time 0, a designated leader (e.g., Node 1) proposes a block.
- Nodes wait until time Δ, and upon receiving one or more proposed blocks, they vote on the first one they received and broadcast these signatures to everyone else.
- If by time 2Δ, $\geq q$ votes are collected during the previous Δ, the block is confirmed. That is, any client that observes $\geq q$ votes for a block confirms that block.
- If no quorum is observed, no block is confirmed, and the next round starts at time 2Δ with the new leader proposing another block.
- Once a block is confirmed at a given height, the network proceeds to the next height.

Assumption: We assume that all nodes have access to perfectly synchronized clocks.

2.4 Why the Protocol Can't Stop at 2Δ

A malicious leader could make sure no block is confirmed by proposing multiple blocks instead of one, by proposing a block late (e.g. at time $\frac{\Delta}{2}$), or by proposing no block at all. However, the protocol cannot simply stop at 2Δ; since doing so would break liveness (there would be no block in this height and the blockchain would not be able to continue). To make sure the blockchain is always live, the next leader needs to take over at 2Δ and propose a new block to continue the process.

3 Adversarial Attacks and Protocol Robustness

3.1 Assumptions on the Adversarial Model

We assume a fixed number of nodes n, with f of them being adversarial ($f \leq n$). We also assume that the signature scheme is still secure—i.e., adversaries cannot forge signatures. The variable f represents the number of adversarial nodes, and the protocol's security guarantees depend on bounding f relative to the total number of nodes n.

3.2 Bounding f for Liveness: Attack #1 (Honest Leader)

If the adversary doesn't vote, an honest leader's block may not get confirmed. This would happen if f nodes didn't vote and the number of honest nodes ended up incapable of establishing a quorum. In other words, if $n - f < q \Rightarrow f > n - q$, the protocol wouldn't be live anymore. Therefore, a necessary condition for liveness is:

$$\boxed{f \leq n - q}$$

3.3 Bounding f for Safety: Attack #2 (Adversarial Leader)

In this attack, assume that the adversarial leader proposes 2 blocks during round 1 and both of them get confirmed (meaning each receives at least q votes). Equivocation refers to the behavior of a node voting for both blocks ("double voting"). In contrast, honest nodes only vote for the first block they observe and never equivocate. Let V_1 and V_2 be the sets of voters who voted for blocks B_1 and B_2, respectively, both proposed at the same height and round. We know that $|V_1 \cup V_2| \leq n$. Also,

$$|V_1| \geq q, \quad |V_2| \geq q \quad [*]$$

Honest nodes vote only once per round, on the first block they see, so any node appearing in both V_1 and V_2 must be adversarial. Therefore, the adversarial nodes are at the intersection $V_1 \cap V_2$ (see Figure 1 below).

The attack will be successful if $f \geq V_1 \cap V_2$. Can we bound the size of this intersection to find how big f needs to be for the safety attack to be successful? By the inclusion-exclusion principle, it follows that

$$|V_1 \cup V_2| = |V_1| + |V_2| - |V_1 \cap V_2|$$

By construction, the total number of voters is at most n, thus:

$$|V_1 \cup V_2| \leq n \Rightarrow |V_1| + |V_2| - |V_1 \cap V_2| \leq n \Rightarrow |V_1 \cap V_2| \geq |V_1| + |V_2| - n$$

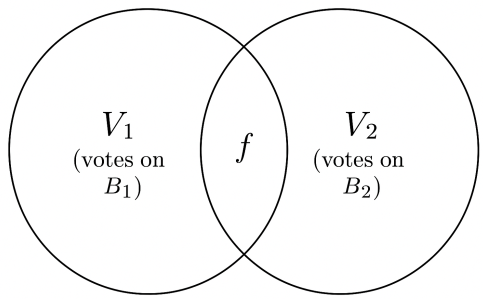


Figure 1: Venn Diagram of Voting Process

which combined with [$*$] gives:

$$|V_1 \cap V_2| \geq q + q - n = 2q - n$$

Thus, for the attack to succeed:

$$f \geq |V_1 \cap V_2| \geq 2q - n \Rightarrow f \geq 2q - n$$

This gives a necessary condition for safety:

$$\boxed{f < 2q - n}$$

Combining this with the earlier liveness condition $f \leq n - q$, we've derived two conditions:

$$\textbf{Liveness: } f \leq n - q \quad \text{and} \quad \textbf{Safety: } f < 2q - n$$

To ensure both safety and liveness, we choose q such that both conditions are satisfied. The optimal choice is where the two inequalities intersect (see Figure 2 below):

$$\boxed{q^* = \frac{2n}{3}}$$

By plugging $q^* = \frac{2n}{3}$ into the liveness and safety conditions, we find:

$$f \leq n - q^* = n - \frac{2n}{3} = \frac{n}{3}$$
$$f < 2q^* - n = 2\left(\frac{2n}{3}\right) - n = \frac{n}{3}$$

Thus, the protocol remains both safe and live as long as the number of adversarial nodes is at most $\lfloor \frac{n}{3} \rfloor$.

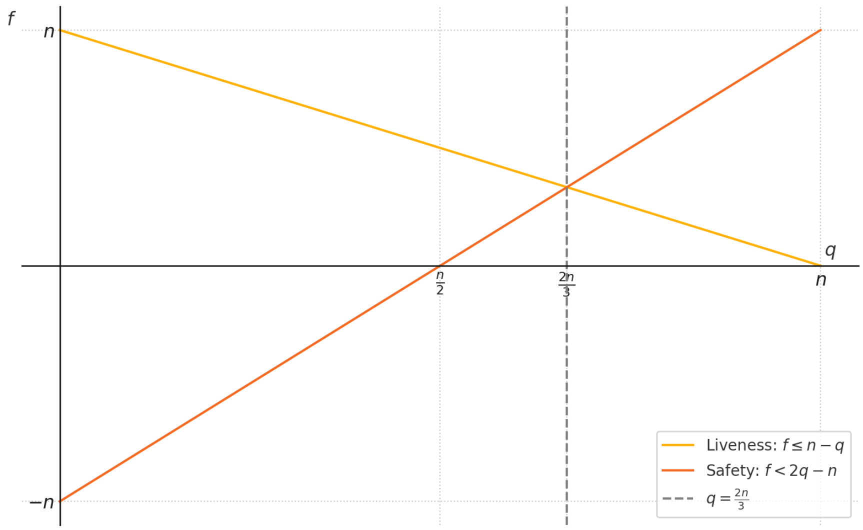


Figure 2: Conditions for Safety and Liveness

Notice that the protocol tolerates up to $f^* = \frac{n}{3}$ adversarial nodes, in contrast to Nakamoto's 51% attack. In terms of latency though, the protocol can achieve fast confirmation (on the order of seconds) compared to Nakamoto's 10 minute block time.

Remark: Liveness Failures Are Tolerable, Safety Failures Are Not. In practice, liveness failures where there is a temporary halt in block production can often be tolerated. An example that was discussed in class was the Solana blockchain that occasionally goes down but resumes operation without the token collapsing. Not only that but there have been instances when the network sees price spikes afterwards [5].

In contrast, safety violations such as conflicting blocks being confirmed result in irreversible damage and essentially kill the blockchain. An example discussed in class was Ethereum Classic, where a safety failure permanently damaged the trust in the chain [10].

So far, we have analyzed a safety attack where an adversarial leader proposes two blocks in the same round. We showed that as long as $f < 2q - n$, the blocks won't be confirmed. However, this condition does not prevent a different class of safety attacks: when two blocks are confirmed but in **different rounds**.

4 A Hidden (But Very Important) Attack Vector

Even though the protocol is supposed to be robust under the bounds we derived earlier, it still remains vulnerable to a more subtle class of attacks that are core to the PBFT family. The threat arises from strategic manipulation of vote timing *across* different rounds. In that case, the protocol cannot maintain safety even below the $\frac{n}{3}$ threshold due to this subtle yet incredibly catastrophic safety attack.

4.1 Setup of the Attack

Let $n = 3f + 1$ and the protocol can tolerate up to f faulty nodes. The quorum size required for confirmation is $q = 2f + 1$. According to the bound for f we found above, our setup should fulfill the necessary and sufficient condition for safety since $f = \left\lfloor \frac{n}{3} \right\rfloor$. However, consider the following scenario:

In round 1, the adversary acts as the leader and proposes two blocks at the same height, an action that violates the protocol rules. The honest votes between the blocks are split:

- Block 1 receives $f + 1$ honest votes.
- Block 2 receives f honest votes.

Because neither block reaches the quorum threshold $q = 2f + 1$, neither is confirmed. The protocol progresses to round 2, where a new (honest) leader proposes a third block. This new block receives all $n = 3f + 1$ votes and is confirmed by time 4Δ. However, the adversary can now reveal their previously withheld f votes for block 1. This introduces a significant safety failure, as clients, at any future point, will be able to see those votes, irrespective of when they were originally cast. We know that the release of the f votes by the adversary was delayed since the new honest leader would have seen the quorum and wouldn't have proposed the new block.

4.2 Why did the Quorum not Prevent the Double-voting?

In this attack, the quorum condition does not prevent the double-voting because the votes that ended up confirming two blocks happened during different rounds. Therefore, the honest nodes, unaware of the adversary's hidden votes, voted again. In Figure 2, we claimed that the intersection of the two circles only consists of adversarial nodes. We just saw though that through this attack, the honest nodes were misguided to double-vote too! This is proof that the protocol is not secure.

4.3 Fix: A Second Round of Voting

The attack described above relies on the adversary withholding votes in round 1 and releasing them after a block has already been confirmed in round 2. This creates ambiguity about which block should be finalized. How could we protect the blockchain from adversaries having hidden votes and releasing them in later rounds? A natural idea would be to reject any votes that arrive late, treating them as invalid.

However, this is a disservice to honest late-joining clients. These clients are not continuously online to monitor voting, meaning that they only observe a record of history. As a result, clients have a difficulty in distinguishing between the two worlds:

- Adversarial votes arrived *on time*.
- Adversarial votes were *withheld* and released later to mislead honest participants.

To disambiguate the two worlds outlined above for late-arriving, honest clients, we need to record additional information regarding the state of the quorum during the previous round. The key idea is that blocks will not be confirmed based on a single round of votes. Instead, honest clients who were present in the voting round should record what they observed and then vote again, but only if they saw quorum the first time. To that end, the protocol introduces the following rules:

- Nodes vote a second time only if they saw a quorum of votes the first time.
- Confirmation uses **only** stage 2 votes.

Through this **two-stage confirmation rule**, honest nodes can attest to whether quorum was seen at the time of voting. This gives late-joining clients a reliable signal and adversaries cannot fake this after the fact.

References

[1] Aptos Labs. The aptos blockchain: Safe, scalable, and upgradeable web3 infrastructure. `https://aptosfoundation.org/whitepaper/aptos-whitepaper_en.pdf`, 2022. Whitepaper.

[2] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938, 2018.

[3] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186. USENIX Association, 1999.

[4] B. Y. Chan and E. Shi. Streamlet: Textbook streamlined blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 1–11, 2020.

[5] CryptoRank. Solana (sol) feeds on outage fud to rocket back above $100, 2024. Accessed: 2025-04-25.

[6] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[7] J. Kwon. Tendermint: Byzantine fault tolerance in the age of blockchains. `https://tendermint.com/static/docs/tendermint.pdf`, 2014. Whitepaper.

[8] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[9] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 347–356. ACM, 2019.

[10] U. Zafar. Ethereum classic (etc) struggles to stay afloat while $8 downside target looms, 2025. Accessed: 2025-04-25.