Ethereum Consensus

An introduction

Alex Stokes
EPF Study Group

What is a blockchain good for anyway?

- A way to manufacture digital scarcity
- Humanity has figured out how to make digital goods
  - Cat jpegs, textbooks, more and more of our world…
- But, until recently, no way to make digital objects that are **scarce**

- Can make as many copies of your cat photo as I want
- Difficulty of prevention has led to big industries around IP violation, piracy, etc
  - See rise of p2p file sharing a la Bittorrent

What is a blockchain good for anyway?

- Blockchains are useful because they can generate **digital scarcity**
- Use this property to emulate all sorts of physical assets in digital realm
  - Money, tokens, property rights
  - We are still figuring out what all is possible

- Was not possible in the prior regime of “copyable” digital objects!

How to make digital scarcity?

- Without loss of generality, assume we want to make a digital money
- Units of money are “coins”
- “Scarcity” means that there are only ever N coins at one time, a user can’t spend more coins than they have

- Strawman: Someone runs a webserver that implements the money protocol
- Implication: Trust operator to ensure no “double-spends”
  - “If the grocery store with 100 apples sells me an apple, they only have 99 apples left”
  - “If I have 100 coins, and send you 5, I only have 95 coins left”

How to make digital scarcity?

- Problem: have never had a trusted operator be able to faithfully honor this prevention of double-spend
  - Bug in money protocol: accidentally print extra coins
  - Active attack in protocol server: compromise operator and print extra coins for profit
  - Dishonest operator: print extra coins for their own benefit
- Critically, scarcity _means_ there is incentive to attack!
  - If coins are valuable, then it is economically rational for me to abuse the money protocol
- Solution: don’t have a single trusted operator
  - To the extent you can remove trust, your digital protocol can more faithfully expose a scarcity abstraction that mirrors our intuition about scarcity in the real world
  - Lots of value creation in structures handling physical scarcity, expect lots of value creation in structures handling digital scarcity – so a worthy goal!

How to remove single trusted operator?

- Make a digital system without a single “leader”
- Computation is *distributed* across many nodes
- Luckily, rich academic literature on distributed computation…
  - Have N nodes which compute some output over the same inputs
  - For example:
    - Inputs are a sequence of transactions that represent transfers/mints in our money protocol
    - Output: current ledger of every user’s balance at a given time
- Consensus via “state machine replication”
  - Structure protocol as computing *some* state as a deterministic function of the inputs
  - Honest nodes *must* end up with the same output
- As number of nodes increases, the system becomes harder to attack

Byzantine fault tolerance

- If more nodes means more secure, then want a high N
  - In the limit, every node on the internet
- But in distributed setting, things can go awry
  - Missed messages
  - Bugs in implementation
  - Hardware failures
  - Active attack!
- Want system to be tolerant to these types of Byzantine faults
- Long history of “Byzantine fault tolerant” consensus protocols
  - Assume some Byzantine fraction of nodes in the system *f*
  - Want the system as a whole to still be able to come to consensus on the system’s inputs
- Two-phase commit
- PBFT: practical Byzantine fault tolerance
- Generally, limited in node count do to algorithmic overheads
  - E.g. N^2 message passing in PBFT
  - Byzantine General’s Problem

Satoshi solves Byzantine General's Problem

- Bitcoin is considered the first solution
- Scales to an ~arbitrary node count
- Open, permissionless participation

- Proof-of-work and "heaviest" chain algorithm mean any honest node will eventually converge to the same current state as any other

Bitcoin consensus

- As state machine replication
  - Inputs: transactions (organized in blocks) to spend bitcoin
  - Outputs: current state of bitcoin ledger
- Use cryptography to reduce possible state space
  - E.g. a user must sign their transaction before it is valid
  - A new block must include the hash of the previous block
- Use proof-of-work to implement consensus
  - Sybil protection: a new block must have a sufficient amount of “work” before it is considered valid
  - Consensus algorithm: the unique chain is the one with the most “work” done
- Issue native currency that nodes care about
  - Provides rewards to direct work to the current single chain with the most work

Ethereum consensus

- Putting aside details, this is how *proof-of-work* Ethereum also worked
- Move to *proof-of-stake*
  - Rather than use exogenous signal for Sybil protection (“work”), use endogenous signal (“stake”) in the system
  - Energy usage concerns with proof-of-work systems
  - Incentive concerns with proof-of-work systems
    - In-protocol signal allows for penalties, not just rewards
    - Reduce attack surface

Proof-of-stake Ethereum

- Looks more like traditional BFT-style consensus protocols
- Set of “validators” who put up stake (ether)
- Need BFT majority to determine the unique state of the chain
- Byzantine faults can be observed by the protocol and stake can be *slashed*

- Less resource intensive, more secure 😎

Break

- Next, look at proof-of-stake protocol in-depth

- Questions?