

Lecture 2: Simple Applications

Guillermo Angeris

June 2022

Outline

Overview and details

NFTs

Voting

Outline

Overview and details

NFTs

Voting

Recap

- ▶ We covered a simple model of a blockchain
- ▶ An interface for the blockchain
- ▶ The simplest possible (useful) application

This lecture

- ▶ More 'interesting' applications: (soulbound?) NFTs, voting
- ▶ Some real-world phenomena
- ▶ An on-chain voting game

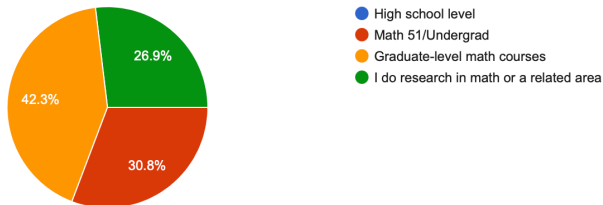
Survey responses

Survey responses

Wide range of backgrounds (as expected):

What's the highest math you've taken?

26 responses

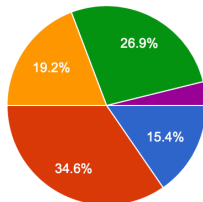


Survey responses (cont.)

Hmm...

How much experience do you have with crypto/blockchain/etc

26 responses



- These terms mean almost nothing to me
- I've messed around a little bit
- I own a few coins and/or have developed toy contracts
- I've developed real contracts in production or have done research in blockchains/consensus/DeFi, etc.
- My brain is literally organized as a blockchain. I know the gas costs of every EVM opcode in existence. I eat,...

Survey responses (cont.)

- ▶ Many responses for the final lecture
 - Optimization and blockchains
 - Zero-knowledge, homomorphic encryption, privacy
 - Attacks and failures of protocols
 - *etc.*
- ▶ Mostly to gauge interest...
- ▶ I will send out a later form closer to then

Outline

Overview and details

NFTs

Voting

What are NFTs?

- ▶ JPEGs? Weird random collections?

What are NFTs?

- ▶ JPEGs? Weird random collections?
- ▶ Stands for *Non-Fungible Token*

What are NFTs?

- ▶ JPEGs? Weird random collections?
- ▶ Stands for *Non-Fungible Token*
- ▶ On-chain existence means it's possible to verify ownership

Interface

Interface

- ▶ An NFT is usually specified by an easily-computable unique identifier
- ▶ (Hash, unique combination of properties, *etc.*)
- ▶ The easier it is to verify, the better

Interface (cont.)

► Implements:

`mint(id, acc)`

`ownerOf(id)`

Interface (cont.)

► Implements:

`mint(id, acc)`

`ownerOf(id)`

► Here, we have:

- `mint(id, acc)`: mint NFT id with acc as owner
- `ownerOf(id)`: returns the account owning id

Interface (cont.)

► Implements:

`mint(id, acc)`

`ownerOf(id)`

► Here, we have:

- `mint(id, acc)`: mint NFT id with acc as owner
- `ownerOf(id)`: returns the account owning id

► Usually mint is permissioned!

Non-transferrable NFTs

- ▶ Very bare bones...

Non-transferrable NFTs

- ▶ Very bare bones...
- ▶ Including no way to transfer the NFT?

Non-transferrable NFTs

- ▶ Very bare bones...
- ▶ Including no way to transfer the NFT?
- ▶ These are (sometimes) called *soul-bound tokens*
- ▶ (After the World of Warcraft property)

Non-transferrable NFTs (cont.)

- ▶ A number of interesting use cases...

Non-transferrable NFTs (cont.)

- ▶ A number of interesting use cases...
- ▶ Though also vaguely dystopian

Non-transferrable NFTs (cont.)

- ▶ A number of interesting use cases...
- ▶ Though also vaguely dystopian

ethereum people, sometimes:
freedom and stuff

ethereum people, other times:

WE WILL ASSIGN UNIQUE
PERSONAL IDENTIFIERS AND
CREDENTIALS TO YOUR SOUL.
YOUR CREDIT SCORE AND
DEGREES WILL DEFINE YOU.
NOTHING WILL EXPUNGE YOUR
ETHEREUM SOUL SCORE, NOT
EVEN THE FIRES OF HELL

8:46 AM · 5/27/22 · [Twitter Web App](#)

144 Retweets **20** Quote Tweets **997** Likes

'Normal' NFTs

- ▶ 'Normal' NFTs have the additional ability to transfer NFTs

`mint(id, acc)`

`ownerOf(id)`

`transfer(id, acc)`

'Normal' NFTs

- ▶ 'Normal' NFTs have the additional ability to transfer NFTs

`mint(id, acc)`

`ownerOf(id)`

`transfer(id, acc)`

- ▶ In this case we recover the 'usual' NFTs

NFTs continued

- ▶ In many ways, NFTs act as cryptographically-verifiable collectibles
- ▶ There are a number of obvious uses (art)
- ▶ A number of less-obvious uses, too: awards, tickets, *etc.*

NFTs continued

- ▶ In many ways, NFTs act as cryptographically-verifiable collectibles
- ▶ There are a number of obvious uses (art)
- ▶ A number of less-obvious uses, too: awards, tickets, *etc.*
- ▶ Simple to implement, difficult to reason about!

Outline

Overview and details

NFTs

Voting

On-chain voting

- ▶ Another simple on-chain application is *voting*
- ▶ Useful in many contexts:
 - Decentralized autonomous organizations (DAOs)
 - Voting for protocol specification
 - Among many others...

Interface

- ▶ The interface is

`vote(option, voteShare)`

`delegate(acc, voteShare)`

`totalVotes(option)`

Some notes

- ▶ vote should fail after a certain time (*block height*)
- ▶ Note that any votes and delegations are public at voting time
- ▶ From transaction history, we can view who voted for what

Some “interesting” votes

- ▶ Context: Juno is a chain
- ▶ ‘Airdropped’ a number of tokens to holders
- ▶ A user (‘whale’) gamed the airdrop, controlled a huge stake
- ▶ On chain vote proposed to remove whale’s tokens

Some “interesting” votes

- ▶ Context: Juno is a chain
- ▶ ‘Airdropped’ a number of tokens to holders
- ▶ A user (‘whale’) gamed the airdrop, controlled a huge stake
- ▶ On chain vote proposed to remove whale’s tokens
- ▶ Hilarity ensues

Initial account

- ▶ Whale account transfers very large amount of JUNO tokens to one account
- ▶ Address:
 juno1aeh8gqu9wr4u8ev6edlgfq03rcy6v5twfn0ja8
for future reference

Holders notice

- ▶ JUNO holders notice and create a vote(!)

PROPOSAL DETAILS

SummaryJSON

#16

Correcting the gamed stakedrop

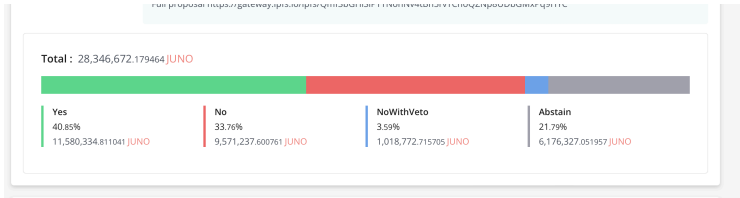
PASSED

Proposer	juno1jnxl8gxfjz4pqswpa2e8j5h9wcgxcsvratfxf
Initial Deposit	0.000000 JUNO
Total Deposit	500.000000 JUNO
Voting Start	2022-03-10 08:21:02
Voting End	2022-03-15 09:21:02
Type	Text
Submit Time	2022-03-10 08:09:16
Deposit End Time	2022-03-20 09:09:16
Details	<div><p>Correcting the gamed stakedrop - Proposed by Core-1 after numerous discussions with the community.</p><p>By voting yes on this proposal you agree to reduce the gamed whale address to 50k (Whalecap that was originally set per entity prior to genesis).</p></div>

- ▶ The vote, if passed, removes all but 50k JUNO (~ 125,000 USD, right now)

Holders vote

► The vote passes (!!)



Holders vote

- ▶ And... just like that...

Holders vote

- ▶ And... just like that...

News Analysis

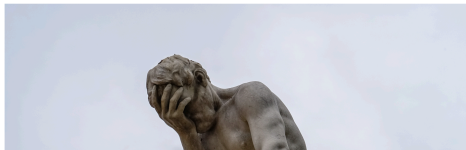
Typo Moves \$36M in Seized JUNO Tokens to Wrong Wallet

Validators, developers and token holders grapple with who is to blame for the copy-paste error that moved the tokens to an address no one can access.



Sam Kessler

🕒 May 5, 2022 at 11:57 a.m. PDT Updated May 6, 2022 at 5:10 a.m. PDT



- ▶ Wait *what*???

The end

- ▶ If there's a lesson here, it is, perhaps:
- ▶ Voting is not always a good replacement for good mechanisms
- ▶ But it can be very useful

Implementing voting mechanisms

- ▶ In some cases even the voting *mechanism* can be dangerous

Implementing voting mechanisms

- ▶ In some cases even the voting *mechanism* can be dangerous

Beanstalk DeFi project robbed of \$182 million in flash loan attack

Reserves were drained after the attacker awarded themselves voting rights.



Written by Charlie Osborne, Contributor on April 21, 2022



Decentralized finance (DeFi) project Beanstalk has lost \$182 million in a flash loan attack.

Non-public voting

- ▶ In some cases, we might not want votes to be public until after vote has ended

Non-public voting

- ▶ In some cases, we might not want votes to be public until after vote has ended
- ▶ Achieve this via *commit-reveal schemes*

Commit-reveal schemes

- ▶ A **commitment** of m is a value s and function f such that
 - a. $f(m, s) = \text{true}$
 - b. Given only s it is 'hard' to compute m' such that $f(m', s) = \text{true}$

Commit-reveal schemes

- ▶ A **commitment** of m is a value s and function f such that
 - a. $f(m, s) = \text{true}$
 - b. Given only s it is 'hard' to compute m' such that $f(m', s) = \text{true}$
- ▶ A commit-reveal voting scheme: vote for m but post *commitment* s on chain

Commit-reveal schemes

- ▶ A **commitment** of m is a value s and function f such that
 - a. $f(m, s) = \text{true}$
 - b. Given only s it is 'hard' to compute m' such that $f(m', s) = \text{true}$
- ▶ A commit-reveal voting scheme: vote for m but post *commitment* s on chain
- ▶ After voting period ends, reveal m

Commit-reveal schemes

- ▶ A **commitment** of m is a value s and function f such that
 - a. $f(m, s) = \text{true}$
 - b. Given only s it is 'hard' to compute m' such that $f(m', s) = \text{true}$
- ▶ A commit-reveal voting scheme: vote for m but post *commitment* s on chain
- ▶ After voting period ends, reveal m
- ▶ (Why does this work?)

A game

- ▶ Called 'A vs. B'
- ▶ Run on chain, contract address:
0xa51594Ba644ef4Da9830F62EB9dC505EB4fC0394
- ▶ Rules
 - Two options, A and B
 - Non-publicly vote for one, in ETH
 - After voting period ends, votes are revealed
 - Option with least votes takes complete pool

Some (quick) analysis

- ▶ There is a simple strategy (assuming no fees, *etc.*)
- ▶ Can you guess it?

Some (quick) analysis

- ▶ There is a simple strategy (assuming no fees, *etc.*)
- ▶ Can you guess it?
- ▶ In fact, in general, no zero-sum symmetric game can have positive payoff
- ▶ (if everyone is rational)

Some (quick) analysis

- ▶ There is a simple strategy (assuming no fees, *etc.*)
- ▶ Can you guess it?
- ▶ In fact, in general, no zero-sum symmetric game can have positive payoff
- ▶ (if everyone is rational)
- ▶ Proof? See homework! (To be posted soon...)

Next lecture(s)

- ▶ We will start talking about AMMs/CFMMs
- ▶ Get to quantitative results!
- ▶ Building blocks to “real” DeFi
- ▶ Such as trading, oracles, stablecoins, *etc.*