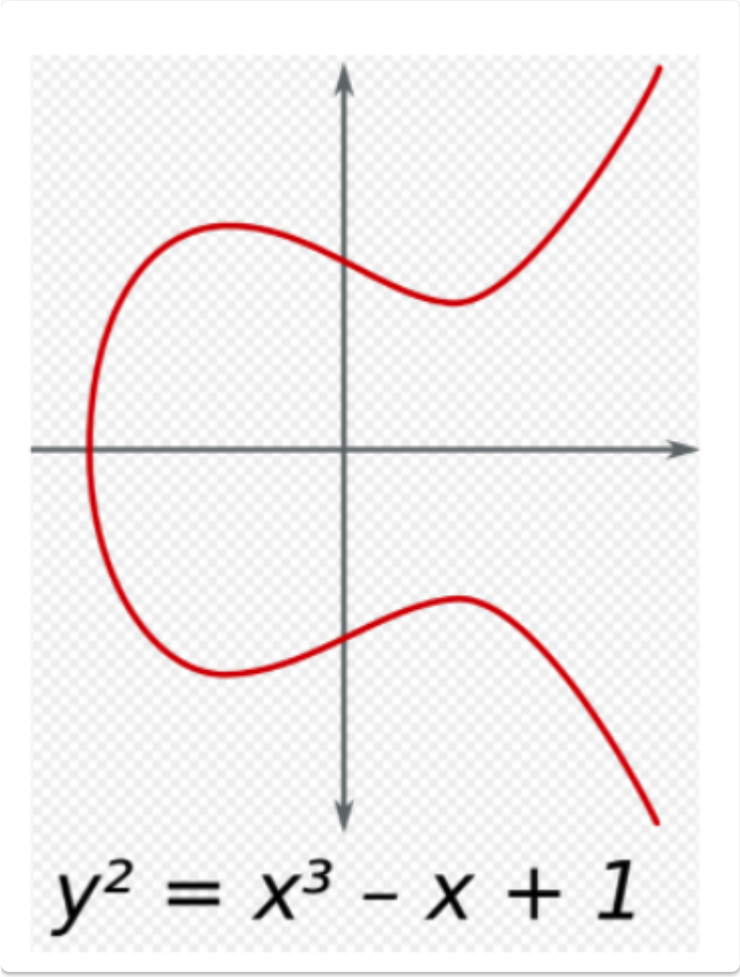
Lesson 2 - Solana Theory

Lesson Plan

- Cryptography
- Consensus
- Solana Network and History

Key Cryptography - Elliptic curves



Solana uses EdDSA (Edwards-curve Digital Signature Algorithm)

It uses the curve25519 curve.
Elliptic curves have a shorter key length for the same level of security as RSA

Keys and addresses

The key may be

- an ed25519 public key
- a program-derived account address (32 byte value from the ed25519 curve)
- a hash of an ed25519 public key with a 32 character string

Solana Architecture

Solana Blocks

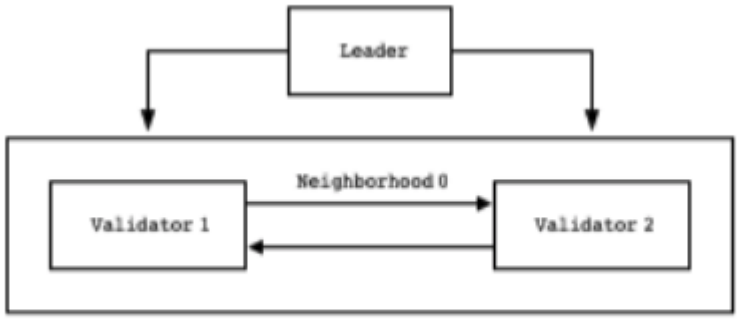
In Solana the concept is a little different, but we still gather transactions together with some meta data.

Overview	
Blockhash	3W9mwZRVvWvnCgwBuGcH4dvTjxp4ZvRceyYkE4aTpSwD
Slot	142,253,865
Timestamp (Local)	Jul 19, 2022 at 18:54:59 GMT+1
Timestamp (UTC)	Jul 19, 2022 at 17:54:59 UTC
Epoch	329
Parent Blockhash	CHJMYUhu7Z4pfZvnvw1EXLZw2B8aYgx3GpvWMfdmygSt
Parent Slot	142,253,864
Child Slot	142,253,866
Processed Transactions	2525
Successful Transactions	1522

Each block is 10MB, blocks are proposed roughly every 800ms,

Solana Network

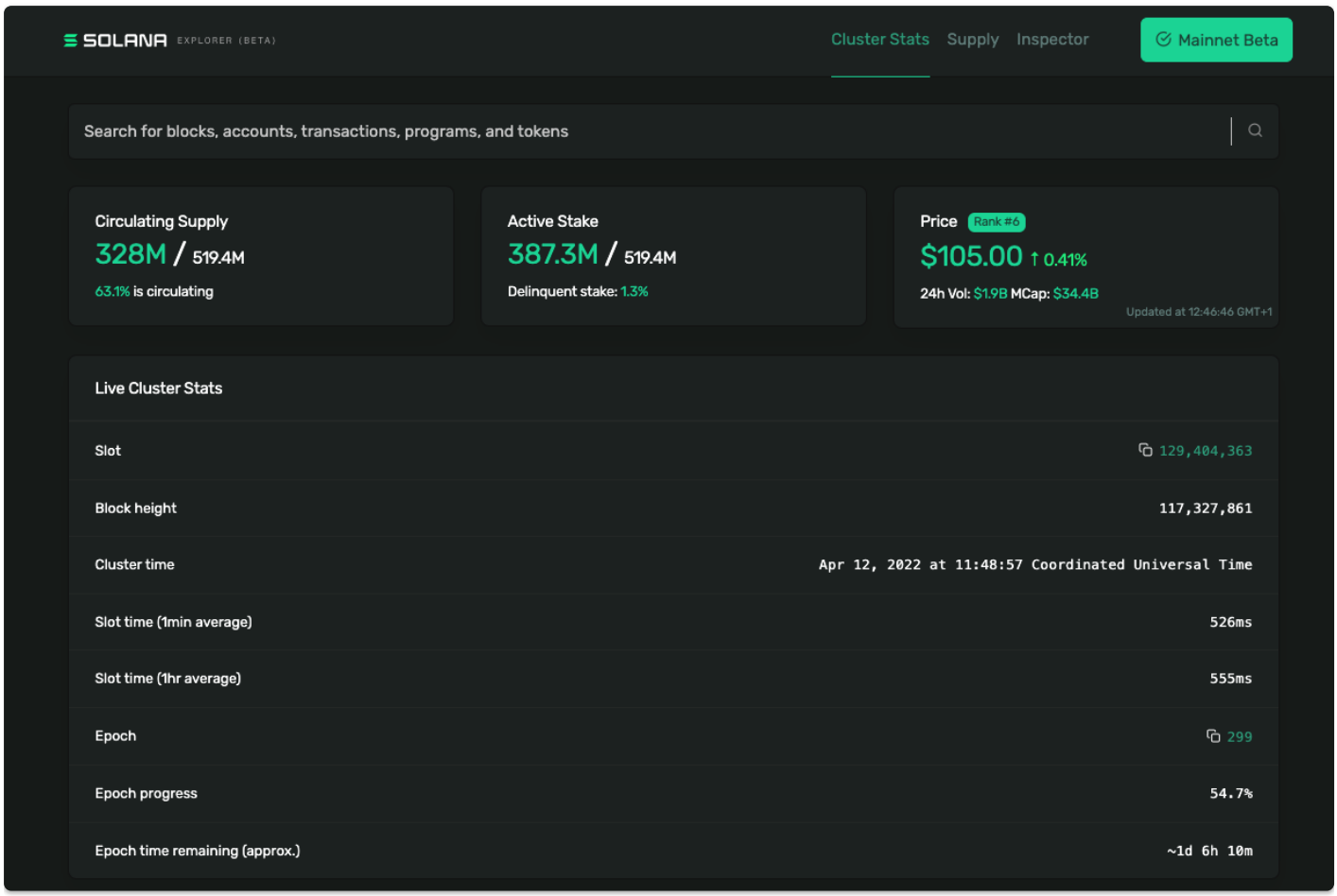
A Solana cluster is a set of validators working together to serve client transactions and maintain the integrity of the ledger. Many clusters may coexist. When two clusters share a common genesis block, they attempt to converge. Otherwise, they simply ignore the existence of the other. Transactions sent to the wrong one are quietly rejected.


Leader
Neighborhood 0
Validator 1
Validator 2

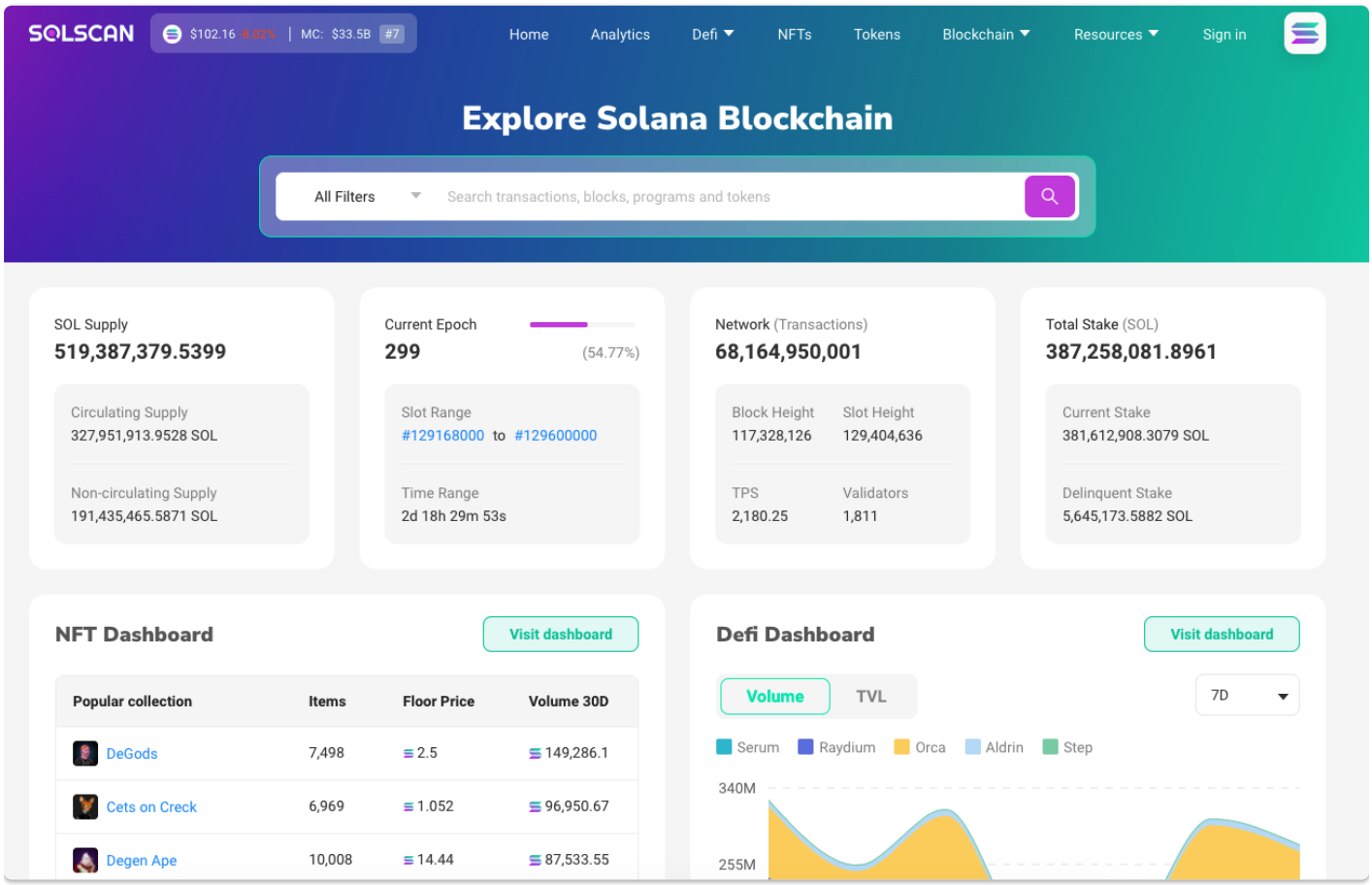

Neighborhood 0
Neighborhood 1
Neighborhood 2
Neighborhood 3
Neighborhood 4
Neighborhood 5
Neighborhood 6

Block Explorers

Solana Explorer



Solscan

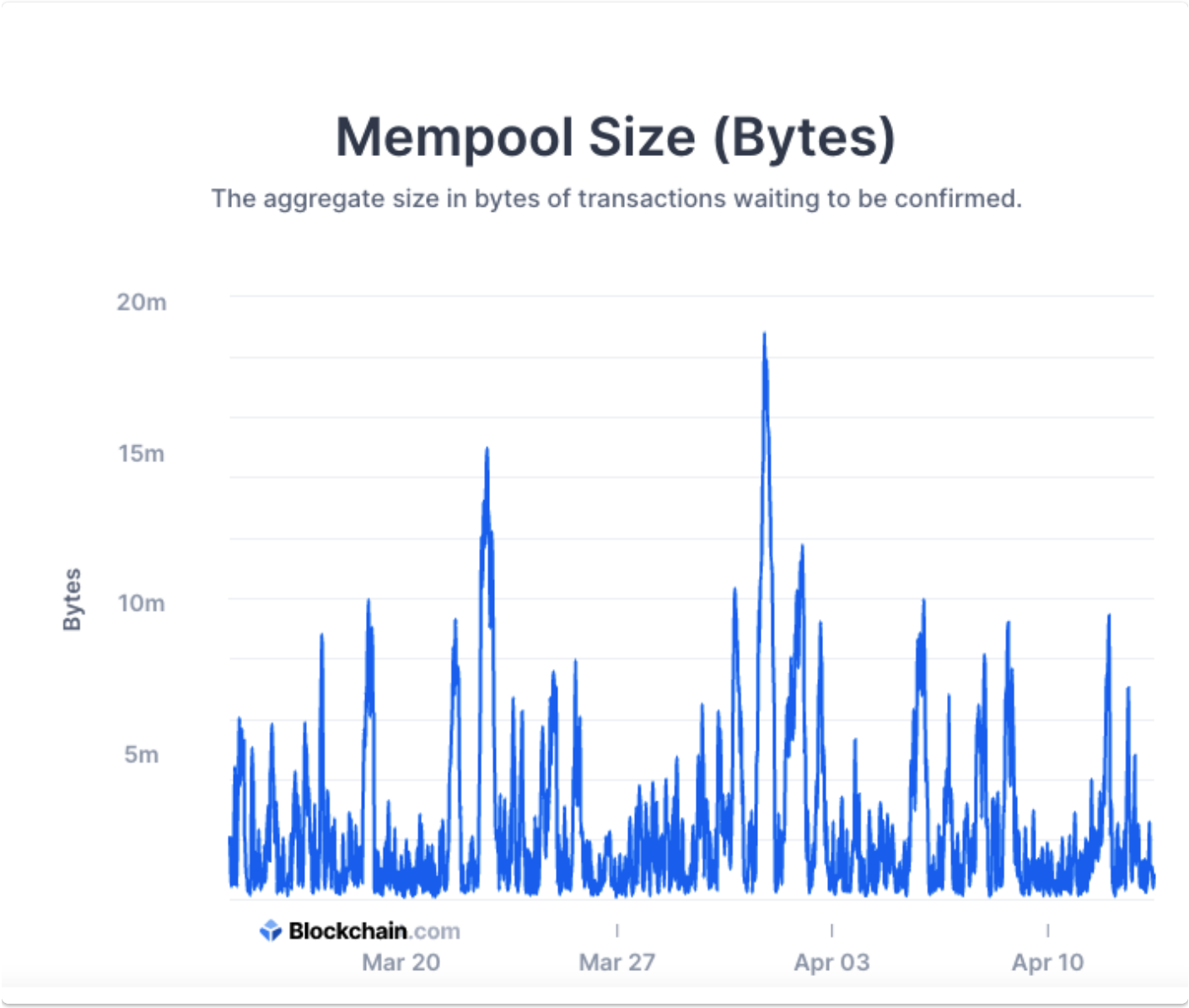


Solana Beach


Solana Beach
BY STAKING FACILITIES × VGNG
Mainnet Beta
Dashboard
Validators
Transactions
Blocks
Tokens
Supply
Search for slots, accounts, transactions, programs, tokens, and validators...
Slot Height
129,404,733
Current Slot Time
0.00 s
Epoch
299
55%
ETA 1d 6h
300
1694 Validators
1552 RPC Nodes
Current Leader
E8EV...iVU1
Next Leaders
kraken

The mempool (pending transactions) and Gulfstream

Mempool size in Bitcoin



From Gulfstream docs

"Mempools in Ethereum and Bitcoin are propagated between random nodes in peer-to-peer fashion using a gossip protocol. Nodes in the network periodically construct a bloom filter representing a local mempool and request any transactions that do not match that filter (along with a few others such as a minimal fee) from other nodes on the network. Propagation of a single transaction to the rest of the network will take at least log(N) steps, consumes bandwidth, memory and computational resources required to filter it."

Gulfstream

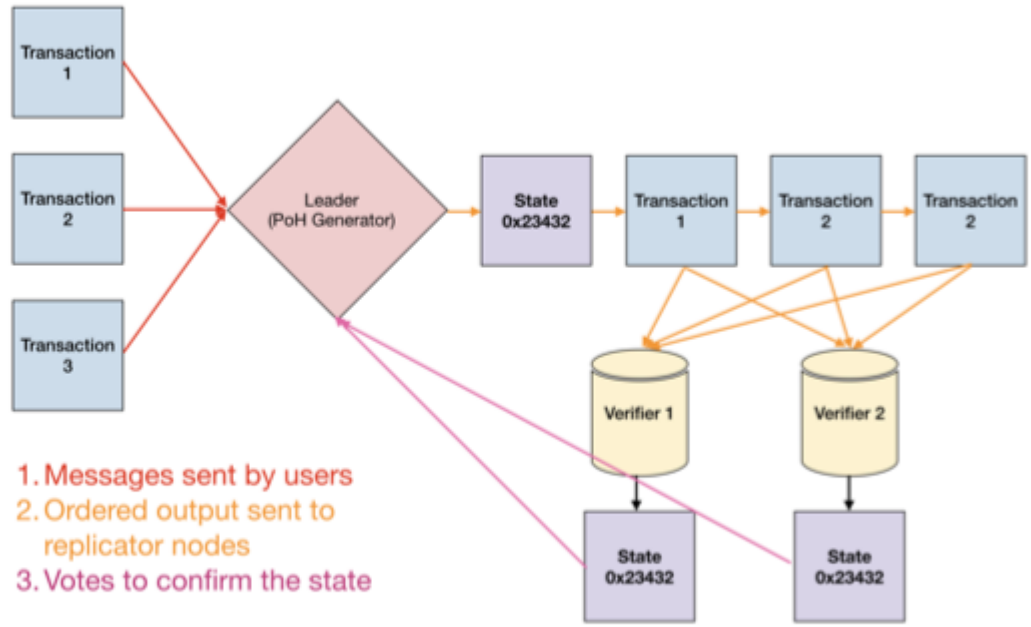


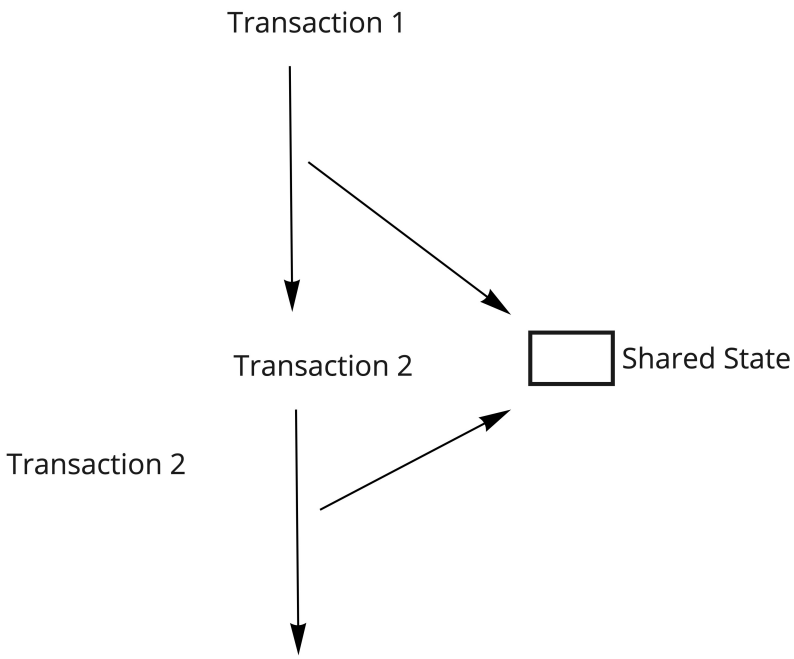
Figure 1: Transaction flow throughout the network.

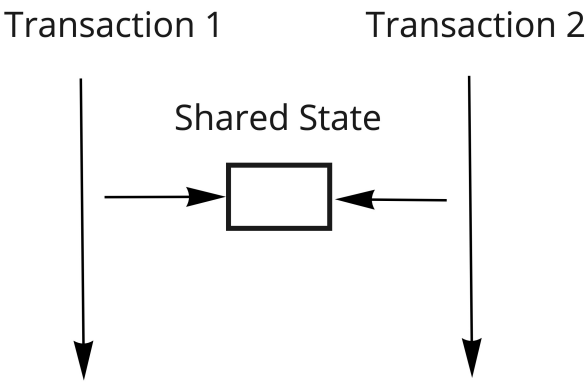
Since every validator knows the order of upcoming leaders, clients and validators forward transactions to the expected leader ahead of time. This allows validators to execute transactions ahead of time, reduce confirmation times, switch leaders faster, and reduce the memory pressure on validators from the unconfirmed transaction pool. This solution is not possible in networks that have a non-deterministic leader

Transactions reference recent blockhash and the transaction is valid only in the children of the referenced block, and is only valid for about 32 blocks.
Assuming block times of 800 ms, that equates to 24 seconds.

Once a transaction is forwarded to any validator, the validator forwards it to one of the upcoming leaders. Clients can subscribe to transaction confirmations from validators. Clients know that a block-hash expires in a finite period of time, or the transaction is confirmed by the network. This allows clients to sign transactions that are guaranteed to execute or fail. Once the network moves past the rollback point such that the blockhash the transaction reference has expired, clients have a guarantee that the transaction is now invalid and will never be executed on chain.

Concurrency and Throughput





How can we improve performance ?

- lets add more threads / cores / do more in parallel
  or
- lets keep everything single threaded

In Ethereum processing of contracts is single threaded
Solana has introduced a way to process programs in parallel

Transaction	Updates

Transaction	Updates
1	Account A
2	Account B
3	Account A
4	Account C

Consensus in distributed systems

- how can participants agree on the state of the system ?

Byzantine fault tolerance (BFT) is the dependability of a fault-tolerant computer system to such conditions where components may fail and there is imperfect information on whether a component has failed.

Why do we need consensus ?

“The double spending problem is a potential flaw in a cryptocurrency or other digital cash scheme whereby the same single digital token can be spent more than once, and this is possible because a digital token consists of a digital file that can be duplicated or falsified.”

Paper

Synchronisation in Distributed Systems

There is a general difficulty with agreeing on time / sequences in distributed systems

In Bitcoin and Ethereum there is no clock available, and participants in the system are given the ability to decide on the sequence of transactions, by mining a block.

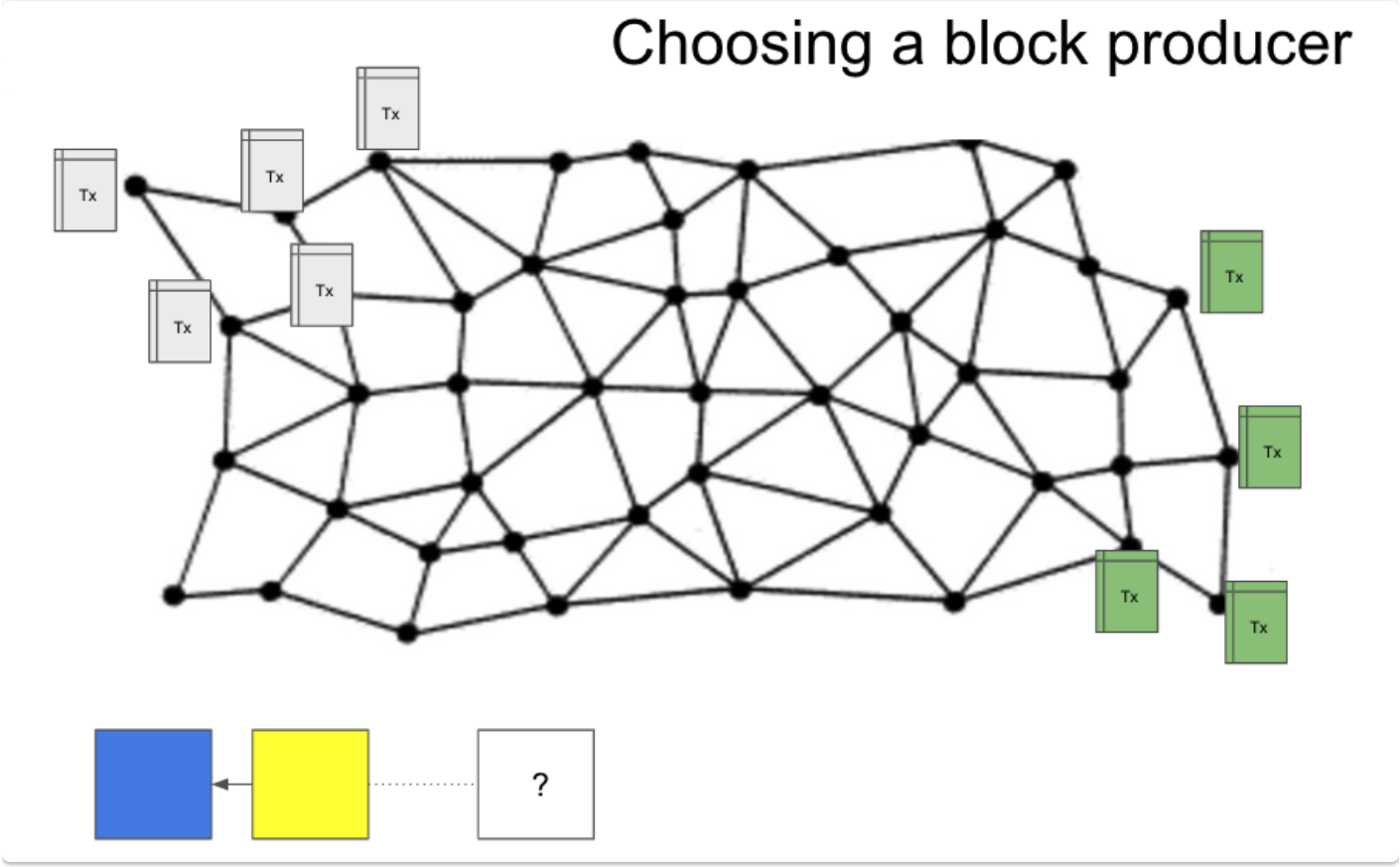
The big innovation from Solana was Proof of History which gives us a verifiable ordering to events

- Proof of Work uses economic incentives to give an ordering of blocks
- Hadera hashgraph uses a median of supplied timestamps

There are 2 Parts to Consensus mechanisms

- Choosing a block producer / leader
- Agreeing on which blocks (transactions) form the canonical chain

For blockchains in general



Typically we allow a block producer to

- choose transactions to be included
- decide on the ordering of transactions

Choosing a leader / block producer

We want this to be 'fair' and difficult to abuse.
PoW uses a race / lottery to solve a puzzle
PoS (DPoS) uses different methods, but often a VRF to assign a time slot to a potential block producer.

One potential problem is liveness

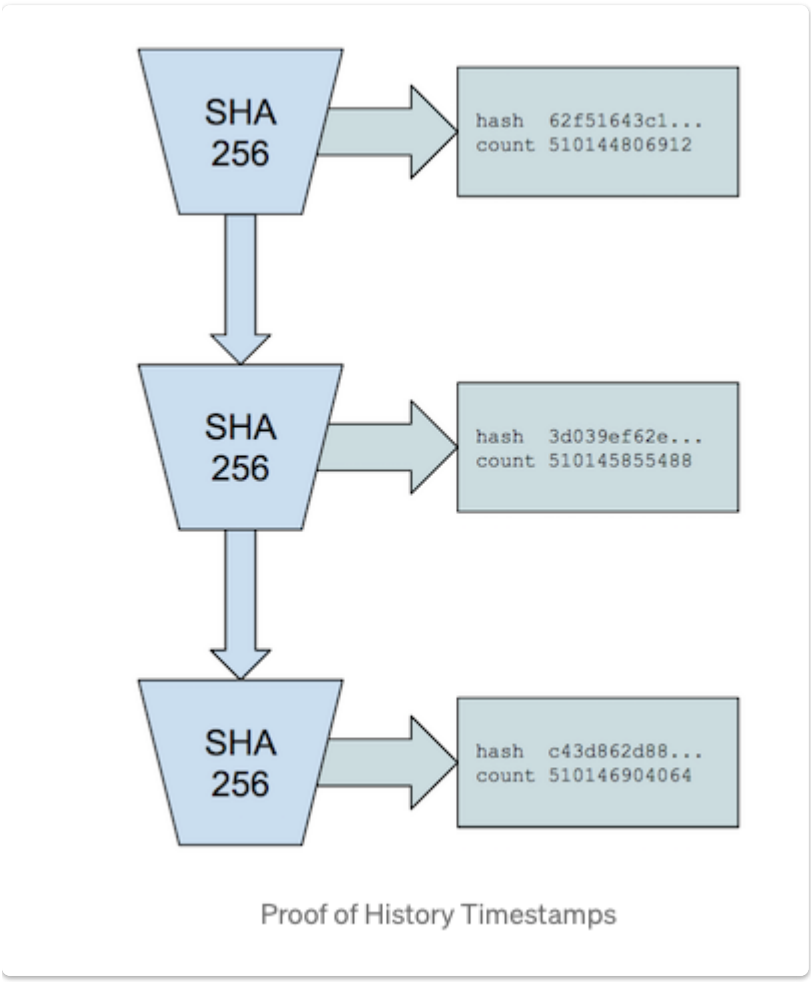
- what if no producer is chosen ?
- what if the chosen block producer fails to produce a block ?

Having a reliable source of time can safely solve timeout issues.

Proof of History

In PoH each node can 'run its own clock' without relying on a central clock.
From Solana docs

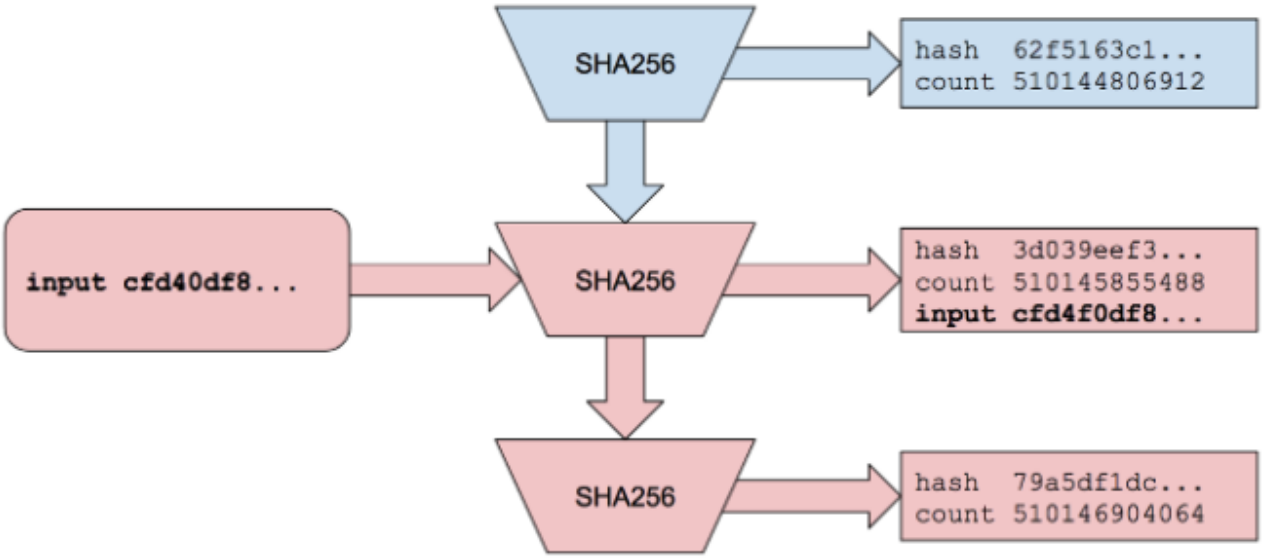
In proof of History we repeatedly hash values, the output from one step forming the input to the next step.



Proof of History Timestamps

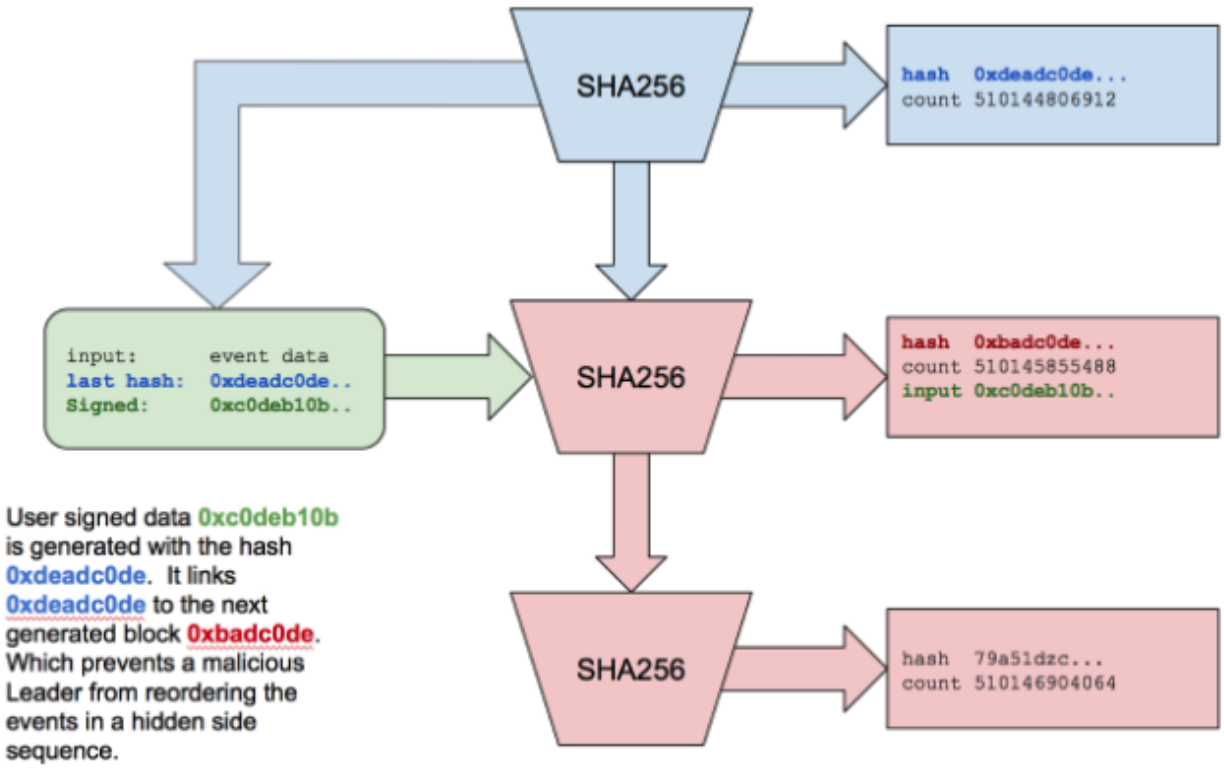
If we were given the hash values and counts out of sequence we would be able to put them into the correct order.

Upper bound on time



Recording messages into a Proof of History sequence

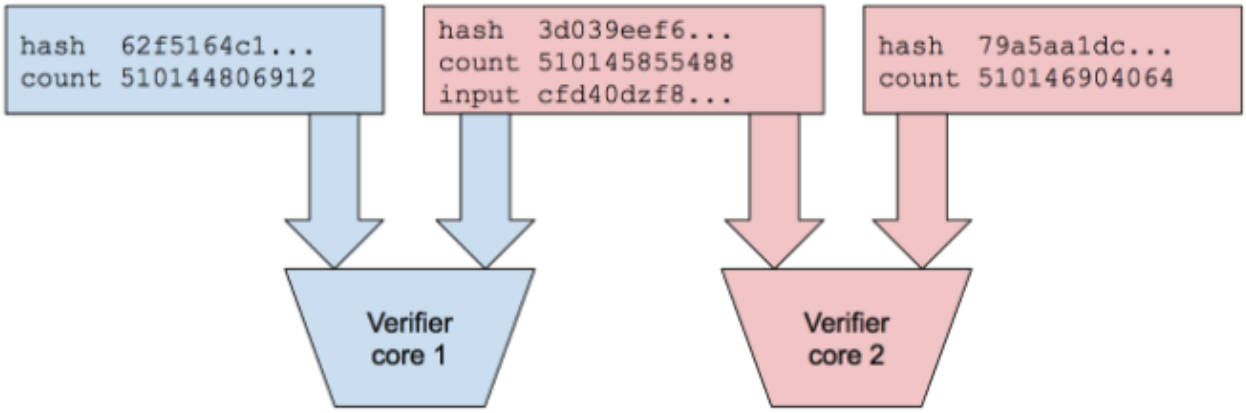
Lower Bound on time



Lower bound on time with Proof of History

Running this process cannot be done in parallel , we need to complete each step in turn.

However, once we have the steps we can verify in parallel.



Parallel Verification

Core 1		
Index	Data	Output Hash
200	sha256(hash199)	hash200
300	sha256(hash299)	hash300
Core 2		
Index	Data	Output Hash
300	sha256(hash299)	hash300
400	sha256(hash399)	hash400

A useful analogy is with a water clock, as in this article

Leader selection

[Leader Schedule Generation Algorithm] (https://docs.solana.com/cluster/leader-rotation#leader-schedule-generation-algorithm)

Leader schedule is generated using a predefined seed. The process is as follows:

1. Periodically use the PoH tick height (a monotonically increasing counter) to seed a stable pseudo-random algorithm.
2. At that height, sample the bank for all the staked accounts with leader identities that have voted within a cluster-configured number of ticks. The sample is called the *active set*.
3. Sort the active set by stake weight.
4. Use the random seed to select nodes weighted by stake to create a stake-weighted ordering.
5. This ordering becomes valid after a cluster-configured number of ticks.

The advantages of deterministic leader selection

Since every validator knows the order of upcoming leaders, clients and validators forward transactions to the expected leader ahead of time. This allows validators to execute transactions ahead of time, reduce confirmation times, switch leaders faster, and reduce the memory pressure on validators from the unconfirmed transaction pool. This solution is not possible in networks that have a non-deterministic leader

This system lowers latency and increases throughput because slot leaders can stream transactions to the rest of the validators in real-time rather than waiting to fill an entire block and send it at once.

As validators keep the count of time, they can stamp each incoming transaction with a time, or proof-of-history value, so the other nodes can order transactions within a block correctly even if they aren't streamed in chronological order. The other nodes can then verify these transactions as they come in rather than having to review an entire block of transactions at once.

Useful articles

Sol overview

Proof-of-history - Medium (by the founder)

Tower BFT (Proof of Stake)

Solana implements a derivation of pBFT, but with one fundamental difference. Proof of History (PoH) provides a global source of time before consensus. Solana's implementation of pBFT uses the PoH as the network clock of time, and the exponentially-increasing time-outs that replicas use in pBFT can be computed and enforced in the PoH itself.

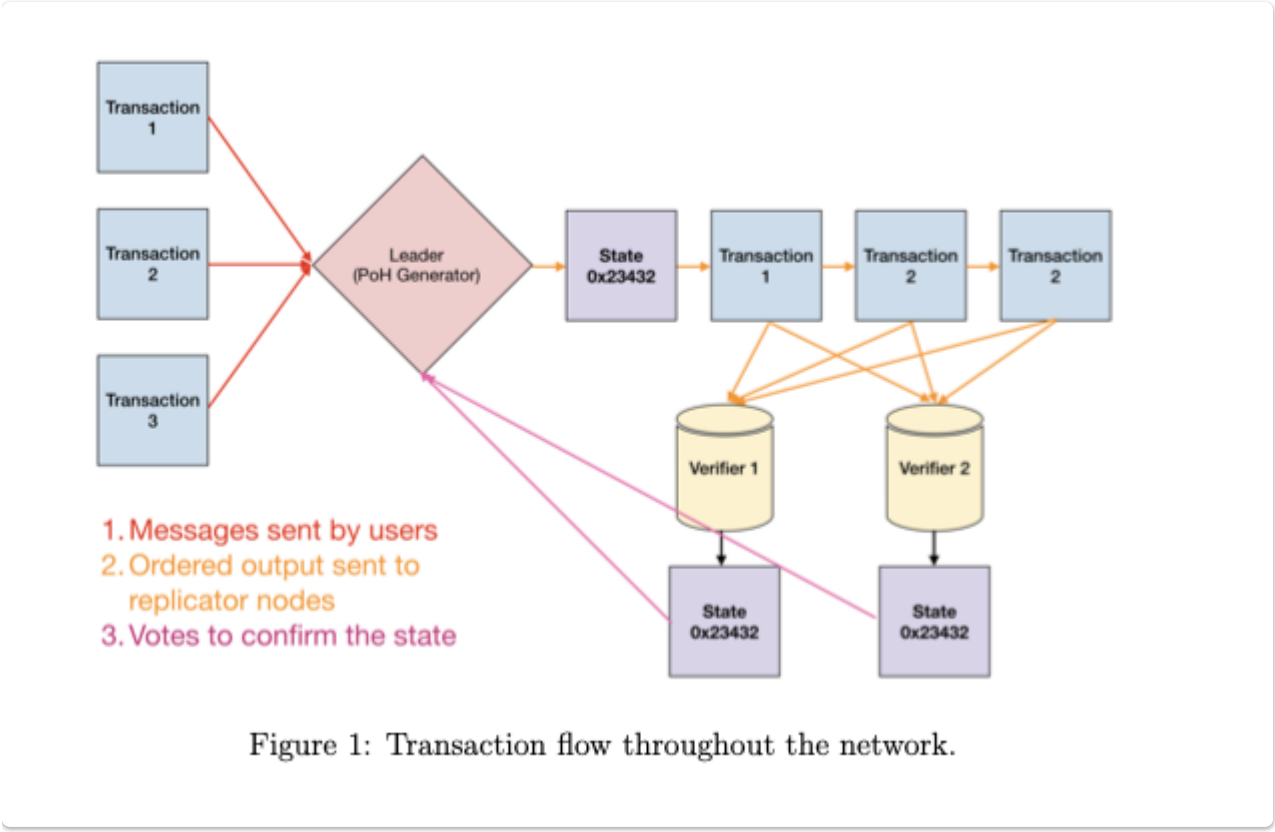


Figure 1: Transaction flow throughout the network.

The leader will be able to publish a signature of the state at a predefined period.
Each bonded validator must confirm that signature by publishing their own signed signature of the state. The vote is a simple yes vote, without a no.

If super majority of the bonded identities have voted within a timeout, then this branch would be accepted as valid.

Every 400ms, the network has a potential rollback point, but every subsequent vote doubles the amount of real time that the network would have to stall before it can unroll that vote.

Once ⅔ of validators have voted on some PoH hash, that PoH hash is canonicalized, and cannot be rolled back. This is distinct from proof of work, in which there is no notion of canonicalization.

Solana History

See Docs

Anatoly Yakovenko published a whitepaper in November 2017 specifying proof of history.
The codebase was moved to Rust and became project Loom.
In Feb 2018 a throughput of > 10K transactions per second was verified.
In March 2018 the project was renamed to Solana to avoid confusion with existing projects.
In July 2018 a testnet of 50 nodes was built which managed up to 250K transactions per second.
In December 2018 the testnet was increased to 150 nodes, and the throughput averaged 200K transactions per second , peaking at 500K.
October 2020 - Wormhole bridge launched (Solana to Ethereum)
November 2022 - Solana affected by collapse of FTX, some stablecoin trading halted.
April 2023 - Solana Saga phone available

Running a validator

Node requirements

1. Validator node:

- CPU
  - 12 cores / 24 threads, or more
  - 2.8GHz, or faster
  - AVX2 instruction support (to use official release binaries, self-compile otherwise)
  - Support for AVX512f and/or SHA-NI instructions is helpful
  - The AMD Zen3 series is popular with the validator community
- RAM
  - 128GB, or more
  - Motherboard with 256GB capacity suggested
- Disk
  - PCIe Gen3 x4 NVME SSD, or better
  - Accounts: 500GB, or larger. High TBW (Total Bytes Written)
  - Ledger: 1TB or larger. High TBW suggested
  - OS: (Optional) 500GB, or larger. SATA OK
  - The OS may be installed on the ledger disk, though testing has shown better performance with the ledger on its own disk
  - Accounts and ledger *can* be stored on the same disk, however due to high IOPS, this is not recommended
  - The Samsung 970 and 980 Pro series SSDs are popular with the validator community
- GPUs
  - Not strictly necessary at this time

    - Motherboard and power supply speced to add one or more high-end GPUs in the future suggested

2. RPC node:
    - CPU
        - 16 cores / 32 threads, or more
    - RAM
        - 256 GB, or more
    - Disk
        - Consider a larger ledger disk if longer transaction history is required
        - Accounts and ledger should not be stored on the same disk

Internet service should be at least 300Mbit/s symmetric, commercial. 1GBit/s preferred.

Price of operation

Solana validators must pay to be eligible to vote. This means a fixed cost of roughly 3 SOL every epoch (2-3 days), which at the time of writing equals costs of ~$100 every single day. This is for transactions which happen on chain and are part of the consensus mechanism.
The validator profits by charging commission on the accounts that delegate to it.

Right now it is required to have combined delegated and staked 50,000 SOL in order to run a node at a profit.

Rewards

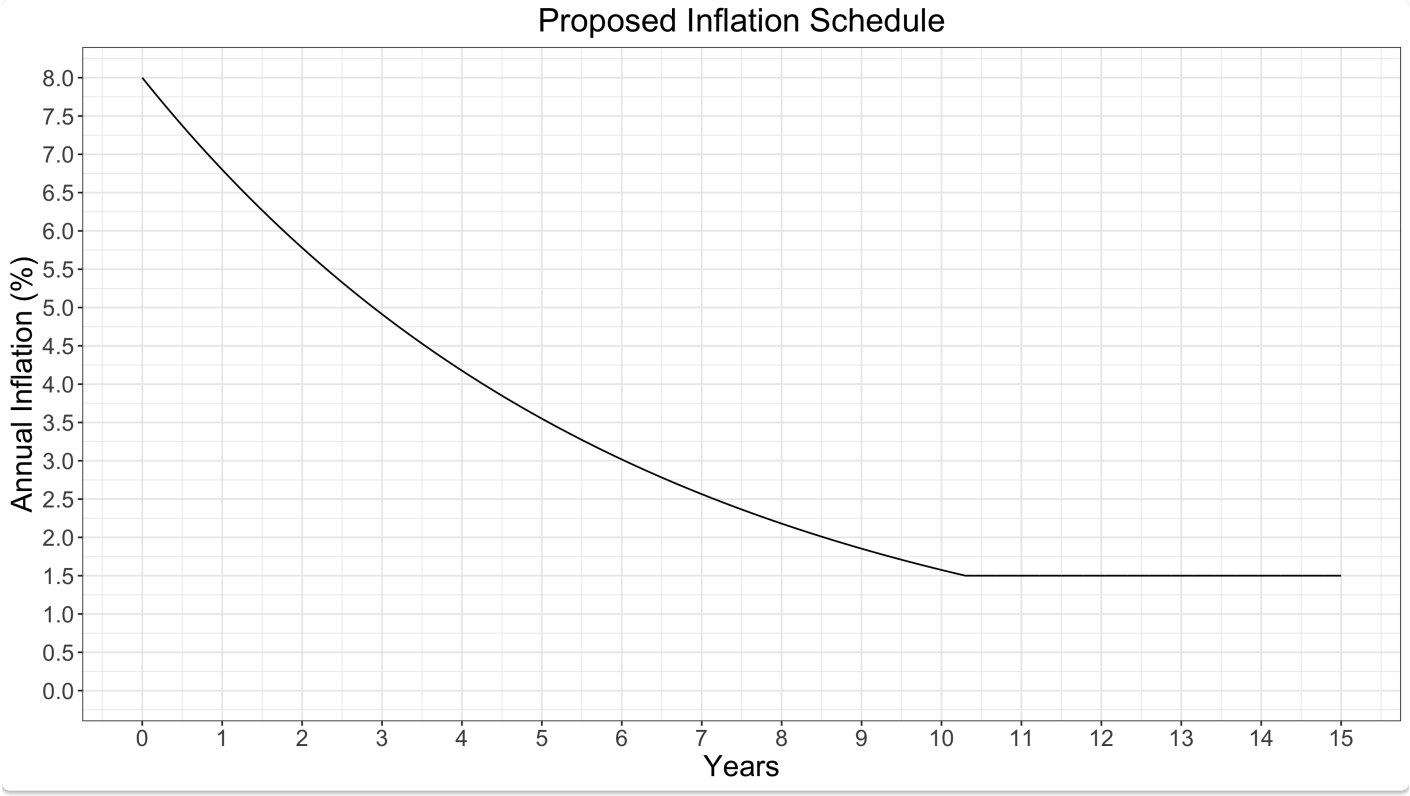
Validators earn SOL in two ways:

- staking reward
- commission on 3rd party stake
- 50% of the transaction fee

Rewards are fixed per epoch and are divided amongst the nodes the support the network.
Rewards are divided according to:

- stake weight
- participation

As time goes on inflation rate will decrease and with it associated epoch rewards.


Proposed Inflation Schedule
Annual Inflation (%)
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Years