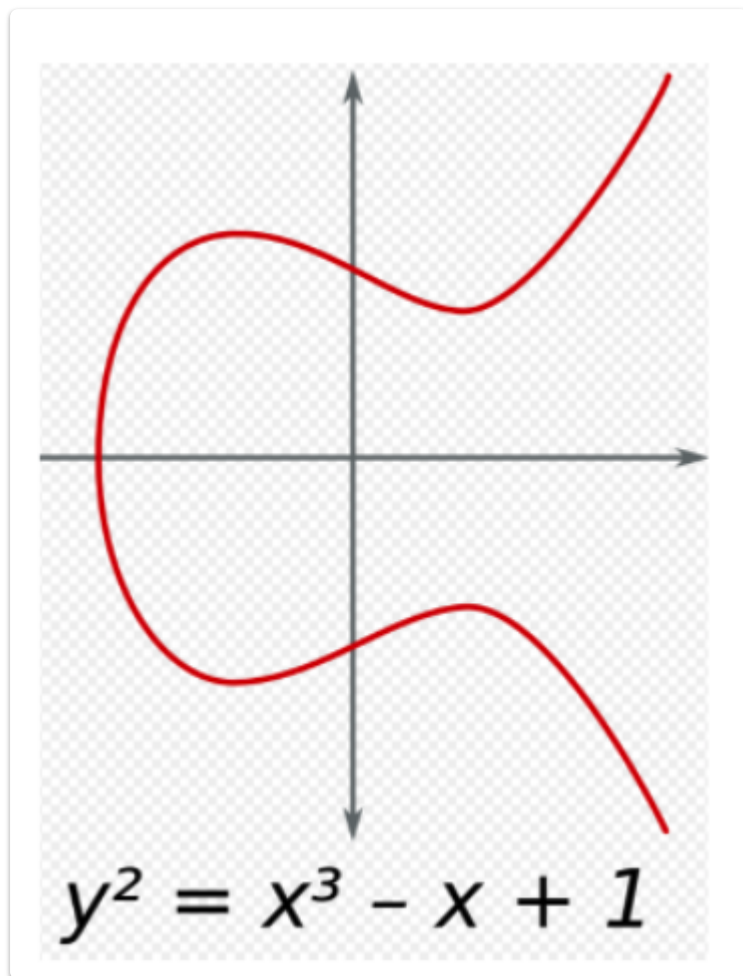


Lesson 2 - Solana Theory

Lesson Plan

- Cryptography
 - Consensus
 - Solana Network and History
 - Crypto-economics
 - Governance
-

Key Cryptography - Elliptic curves

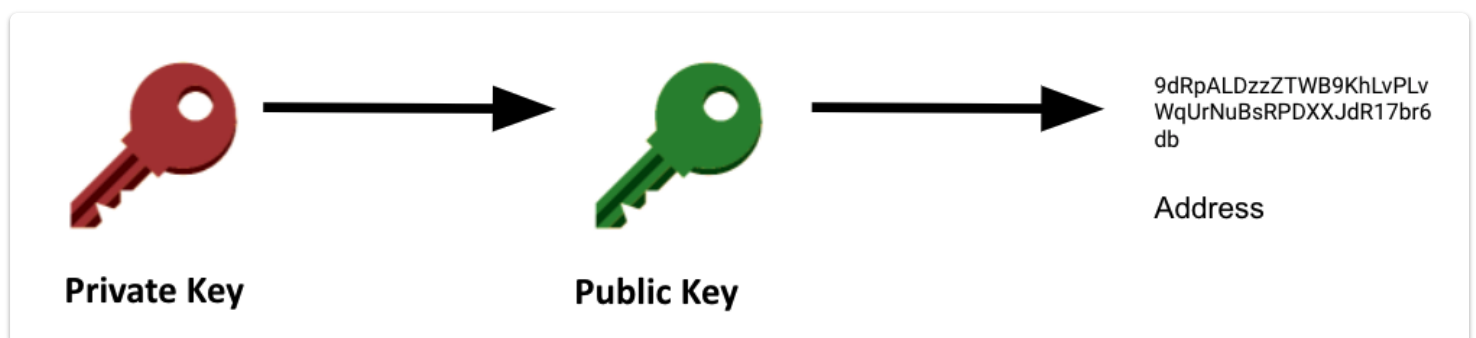


Solana uses EdDSA (Edwards-curve Digital Signature Algorithm)

It uses the curve25519 curve.

Elliptic curves have a shorter key length for the same level of security as RSA

KEYS AND ADDRESSES



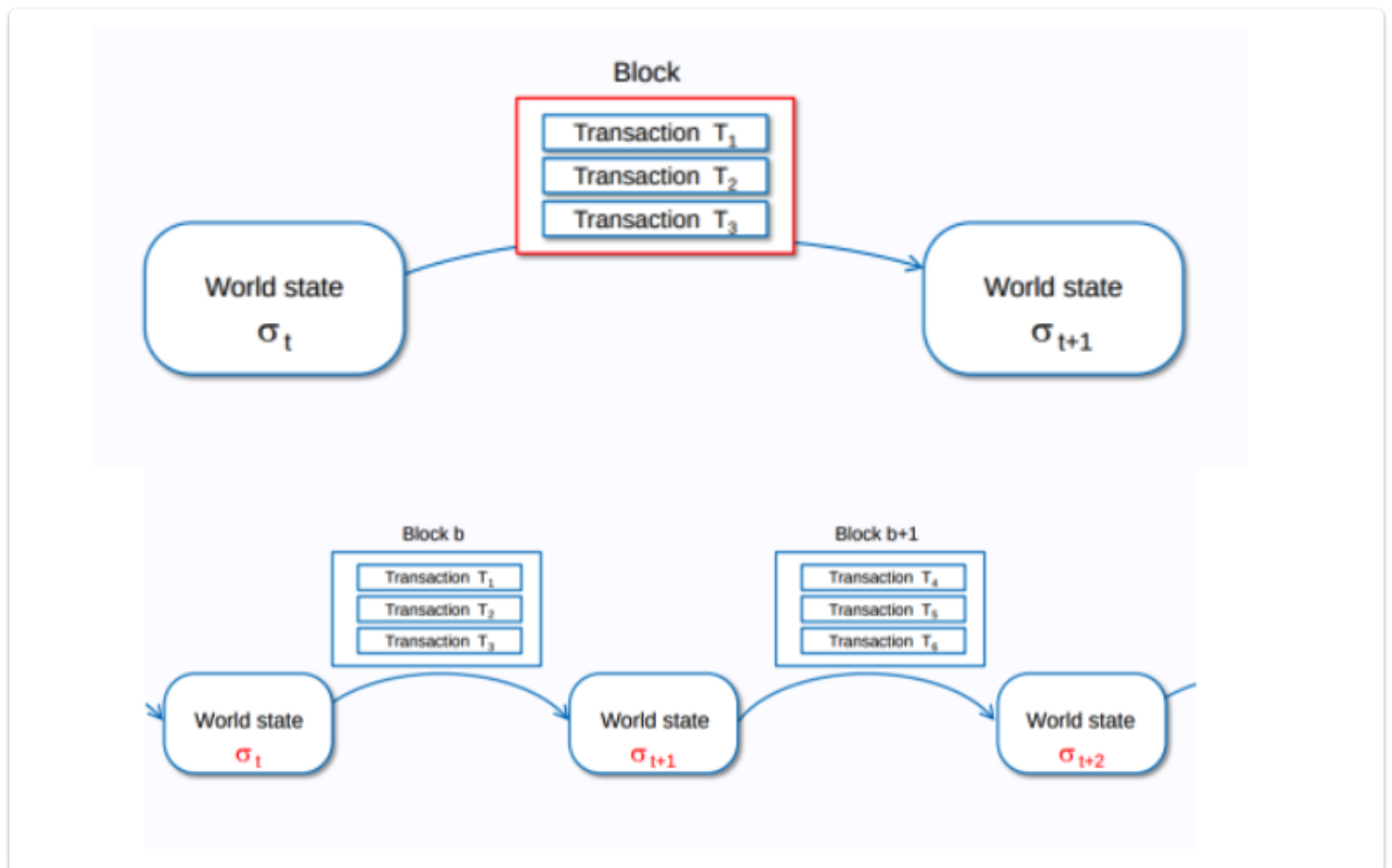
The key may be

- 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

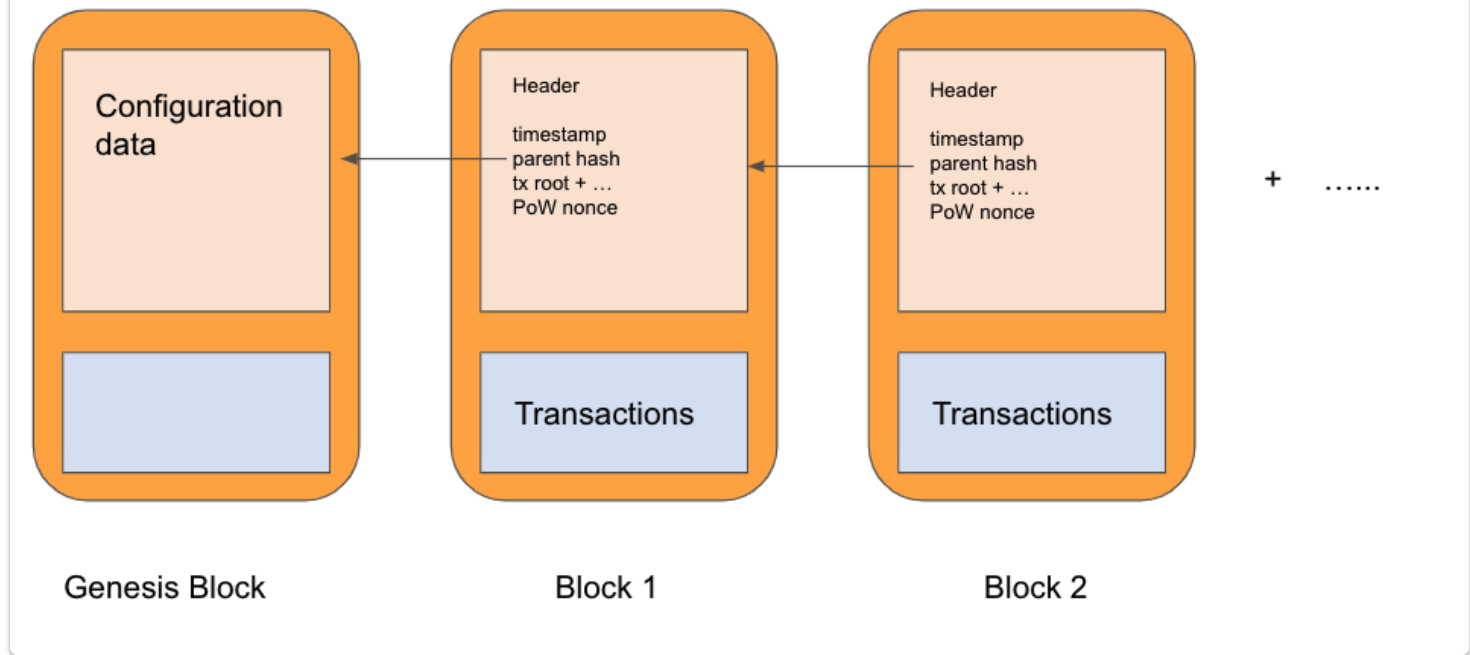
Blockchain components in more detail

- A peer-to-peer (P2P) network connecting participants and propagating transactions and blocks of verified transactions, based on a standardized "gossip" protocol
- Messages, in the form of transactions, representing state transitions
- A set of consensus rules, governing what constitutes a transaction and what makes for a valid state transition
- A state machine that processes transactions according to the consensus rules
- A chain of cryptographically secured blocks that acts as a journal of all the verified and accepted state transitions
- A consensus algorithm that decentralizes control over the blockchain, by forcing participants to cooperate in the enforcement of the consensus rules
- A game-theoretically sound incentivization scheme to economically secure the state machine in an open environment
- One or more open source software implementations of the above ("clients")

Blockchain as a state machine in Bitcoin



General Blockchain Structure for example Bitcoin






Genesis Block - the starting block

Bitcoin Genesis Block			
Raw Hex Version			
00000000	01 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000010	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000020	00 00 00 00 3B A3 ED FD	7A 7B 12 B2 7A C7 2C 3E;fíýz{.²zÇ,>
00000030	67 76 8F 61 7F C8 1B C3	88 8A 51 32 3A 9F B8 AA	gv.a.È.Ã^ŠQ2:Ÿ,ø
00000040	4B 1E 5E 4A 29 AB 5F 49	FF FF 00 1D 1D AC 2B 7C	K.^J)«_IÿŸ...¬+
00000050	01 01 00 00 00 01 00 00	00 00 00 00 00 00 00 00
00000060	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000070	00 00 00 00 00 00 FF FF	FF FF 4D 04 FF FF 00 1DÿÿÿÿM.ÿÿ..
00000080	01 04 45 54 68 65 20 54	69 6D 65 73 20 30 33 2F	..EThe Times 03/
00000090	4A 61 6E 2F 32 30 30 39	20 43 68 61 6E 63 65 6C	Jan/2009 Chancel
000000A0	6C 6F 72 20 6F 6E 20 62	72 69 6E 6B 20 6F 66 20	lor on brink of
000000B0	73 65 63 6F 6E 64 20 62	61 69 6C 6F 75 74 20 66	second bailout f
000000C0	6F 72 20 62 61 6E 6B 73	FF FF FF FF 01 00 F2 05	or banksÿÿÿÿ..ð.
000000D0	2A 01 00 00 00 43 41 04	67 8A FD B0 FE 55 48 27	*....CA.gŠŸ°pUH'
000000E0	19 67 F1 A6 71 30 B7 10	5C D6 A8 28 E0 39 09 A6	.gñ q0°.\\Ö"(à9.!
000000F0	79 62 E0 EA 1F 61 DE B6	49 F6 BC 3F 4C EF 38 C4	ybàê.a»¶IÖk?Lİ8Ä
00000100	F3 55 04 E5 1E C1 12 DE	5C 38 4D F7 BA 0B 8D 57	óU.Á.Á.▷\8M+ø..W
00000110	8A 4C 70 2B 6B F1 1D 5F	AC 00 00 00 00	ŠLp+kñ._¬....

Solana Architecture

Solana Blocks

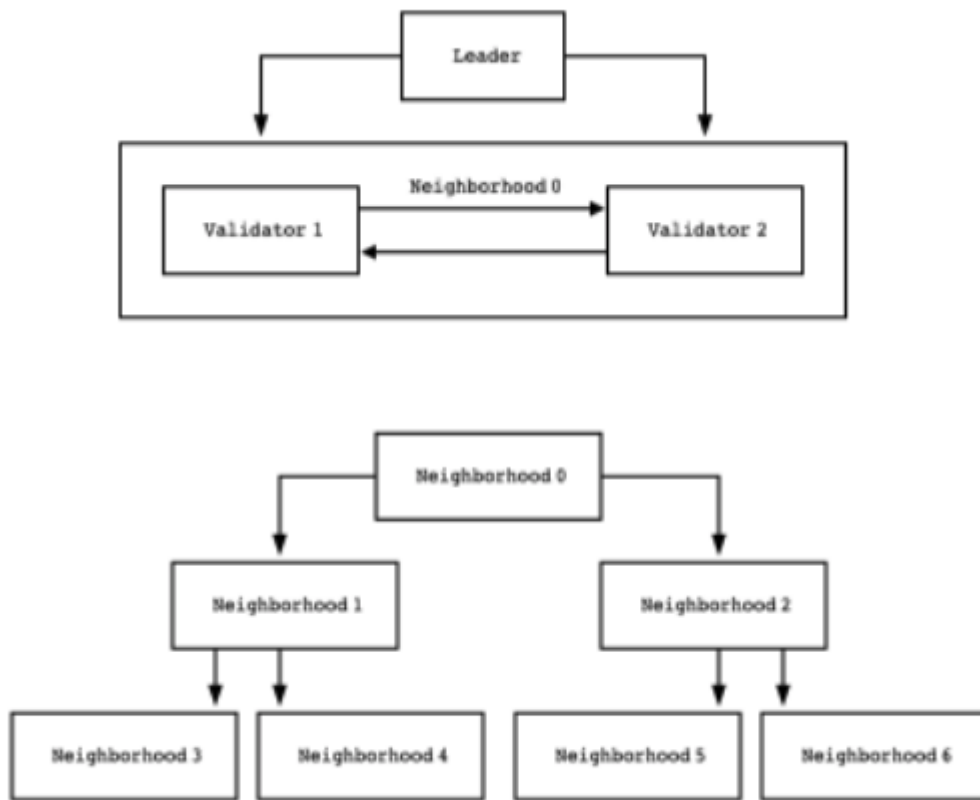
In Solana the concept is a little different, but we still gather transactions together with some meta data.

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	CHJMYUhu7Z4pfZvnmv1EXLZw2B8aYgx3GpvmMfdmygSt
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.



Block Explorers

Solana Explorer

SOLANA EXPLORER (BETA)

Cluster StatsSupplyInspector

Mainnet Beta

Search for blocks, accounts, transactions, programs, and tokens

Circulating Supply

328M / 519.4M

63.1% is circulating

Active Stake

387.3M / 519.4M

Delinquent stake: 1.3%

Price Rank #6

\$105.00 ↑ 0.41%

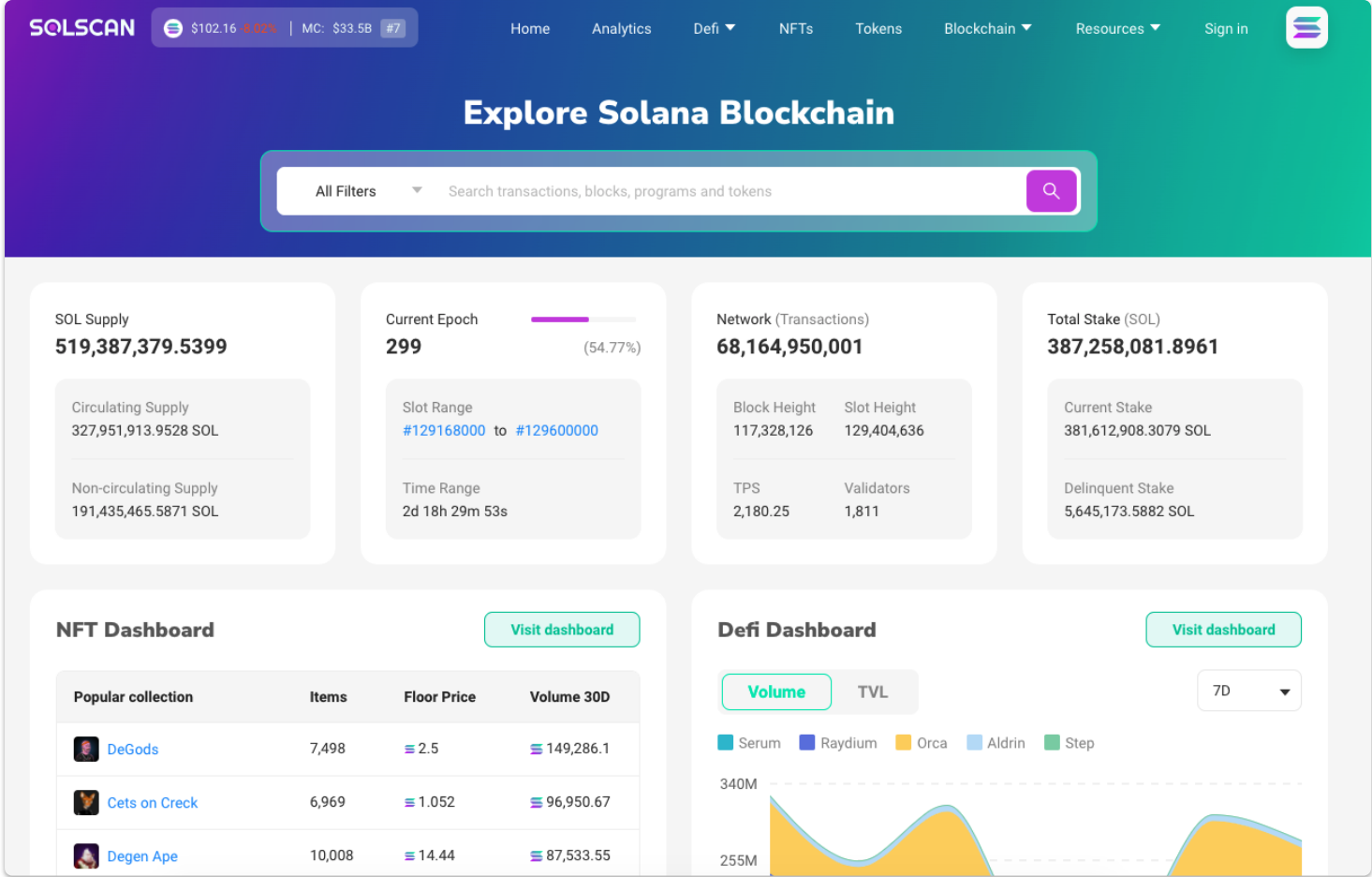
24h Vol: \$1.9B MCap: \$34.4B

Updated at 12:46:46 GMT+1

Live Cluster Stats

Slot	129,404,363
Block height	117,327,861
Cluster time	Apr 12, 2022 at 11:48:57 Coordinated Universal Time
Slot time (1min average)	526ms
Slot time (1hr average)	555ms
Epoch	299
Epoch progress	54.7%
Epoch time remaining (approx.)	~1d 6h 10m

Solscan



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.

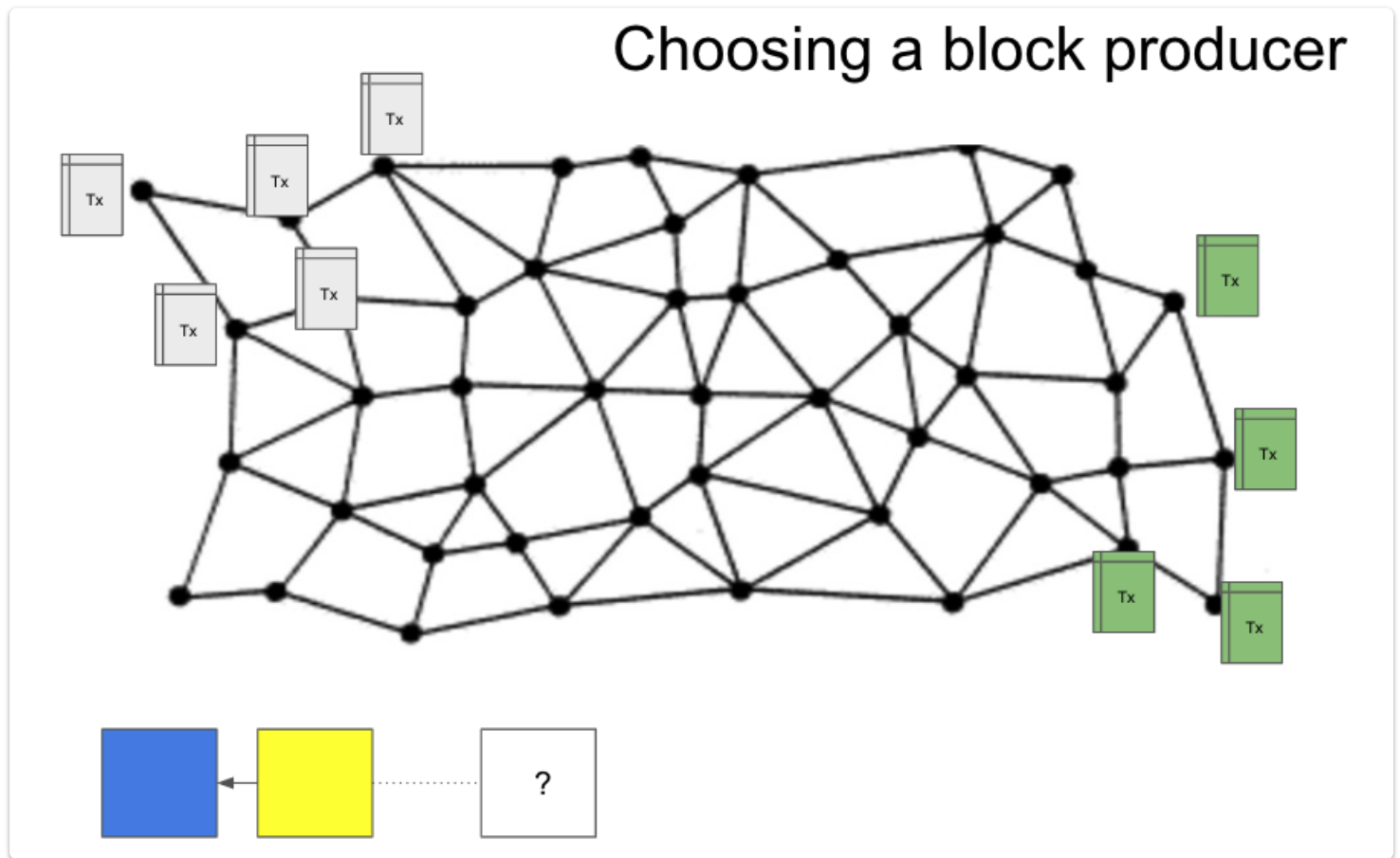
The big innovation from Solana was Proof of History which gives us a verifiable ordering to events

- 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.

Some Implementations of Sybil / Consensus Mechanisms

- Practical Byzantine Fault Tolerance (pBFT) Castro and Liskov 1999
- Nakamoto Consensus (PoW) 2008

Now there are many "Proof of

Stake / Authority / Burn / Elapsed Time / Spacetime"

Solana - Proof of History / Tower Consensus (BFT)

Proof of Stake

There are many implementations of PoS - Ethereum , Mina, NXT ...

Common features

- Potential block producers have to submit a stake of the native crypto currency to be eligible
- The current block producer is chosen at random, the probability of being chosen will depend on the amount of stake offered.
- If the block producer behaves maliciously they lose some or all of their stake

Ethereum PoS

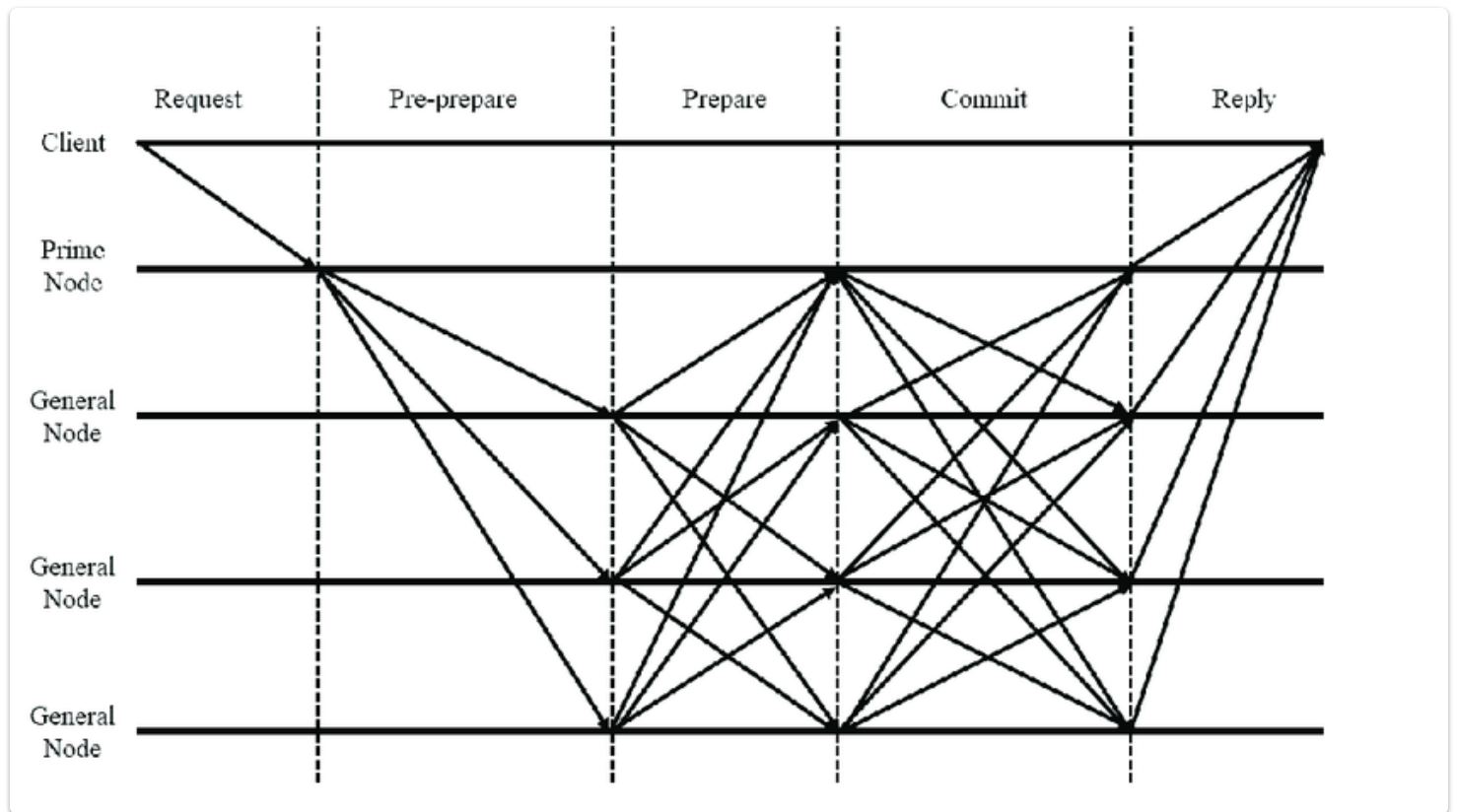
Ethereum now uses PoS as a means of choosing a block proposer.

- Slots are assigned to producers who have deposited a stake

The consensus mechanism has two parts

- LMD-GHOST – which adds new blocks and decides what the head of the chain is
 - Casper FFG which makes the final decision on which blocks are and are not a part of the chain.
-

pBFT - Practical Byzantine Fault Tolerance



pBFT requires $3f+1$ nodes in the system, where f is the maximum number of faulty nodes that the system can tolerate.

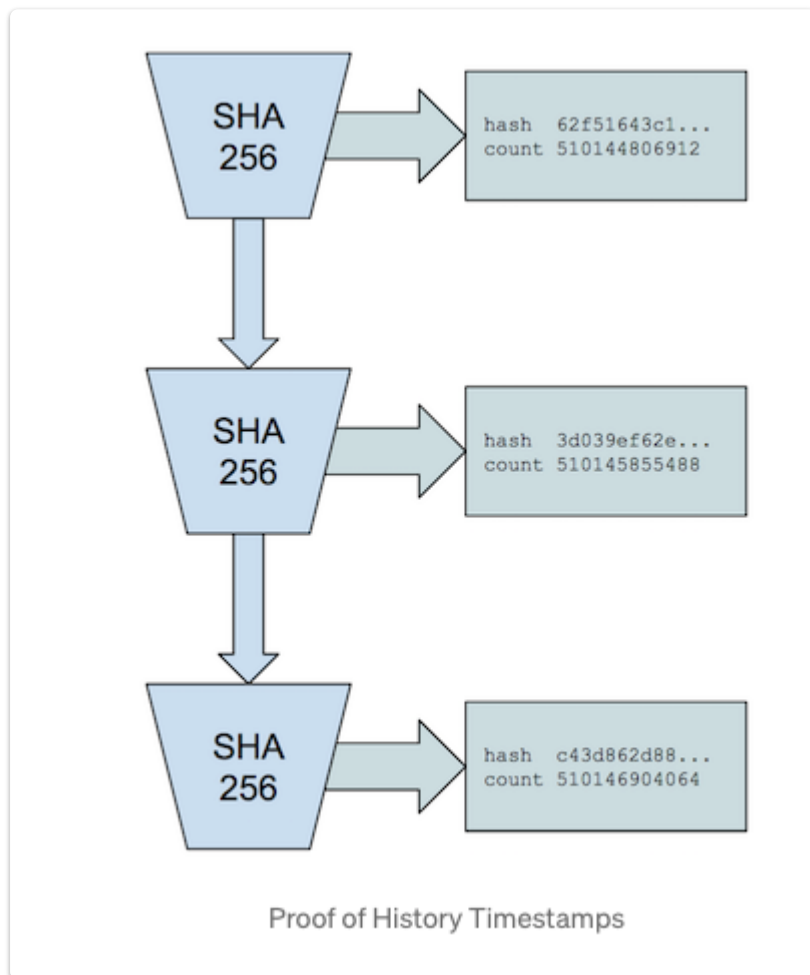
Therefore, for the group of nodes to make any decision, approval from $2f+1$ nodes is required.

Proof of History

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

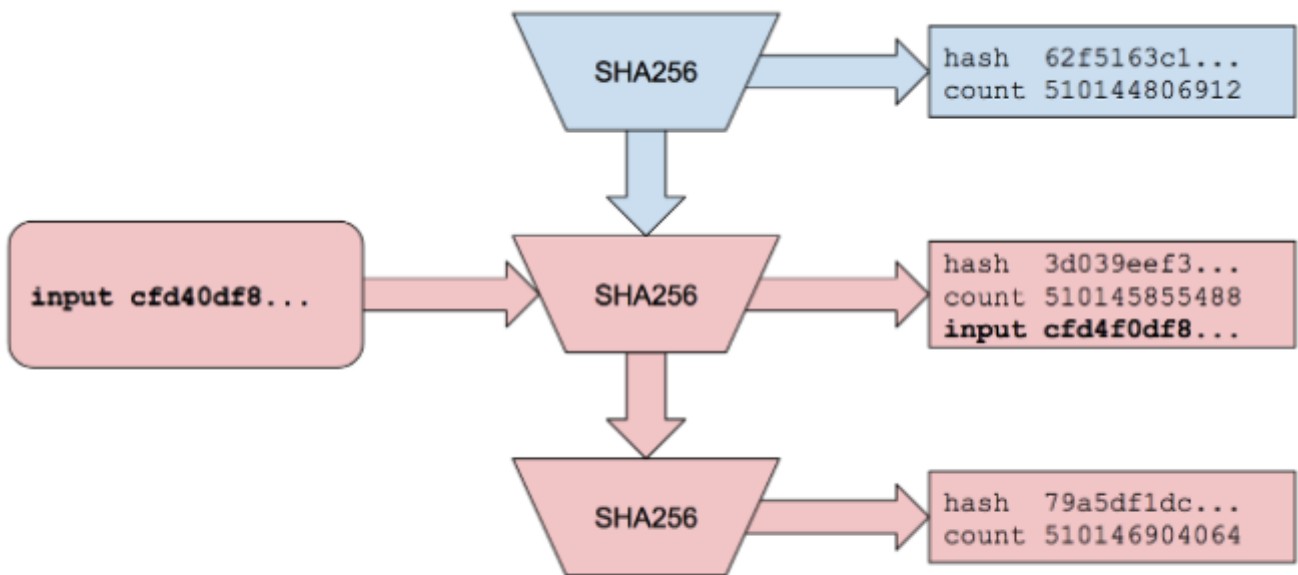
From Solana [docs](#)

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



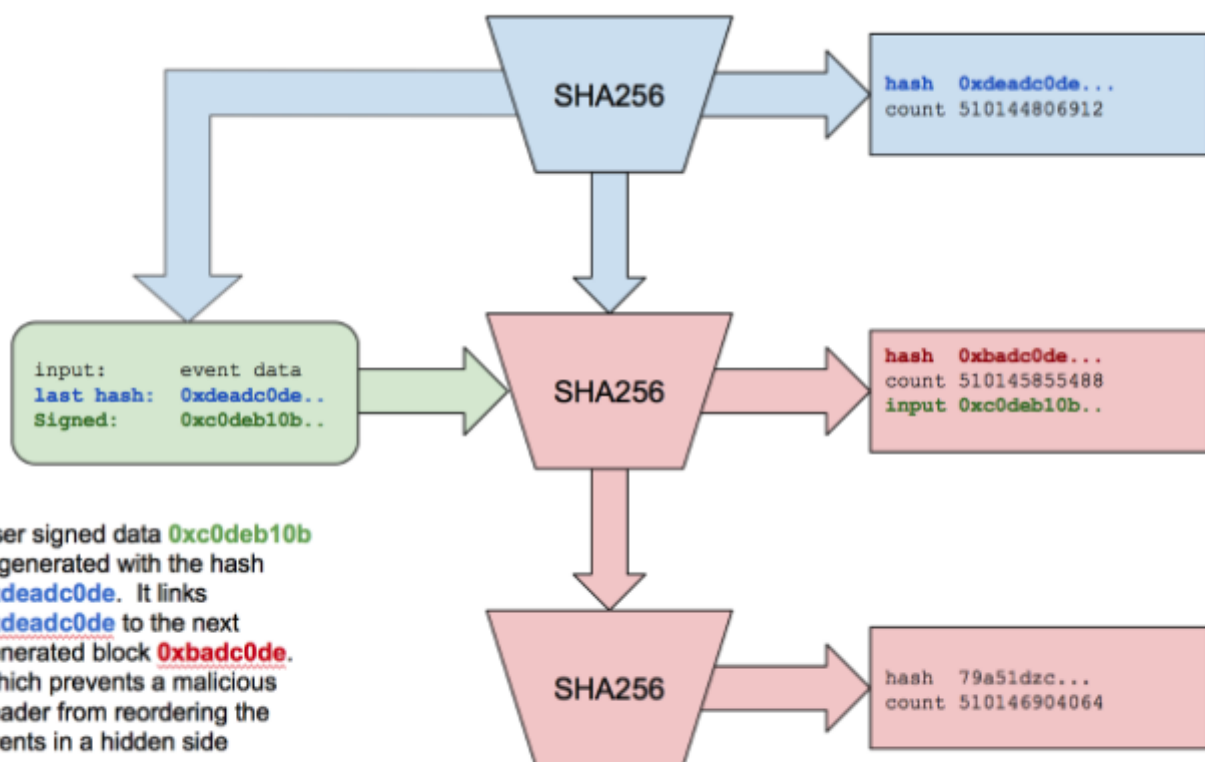
If we were given the hash values and counts out of sequence we would be able to put them into the correct order.

Upper bound on time



Recording messages into a Proof of History sequence

Lower Bound on time

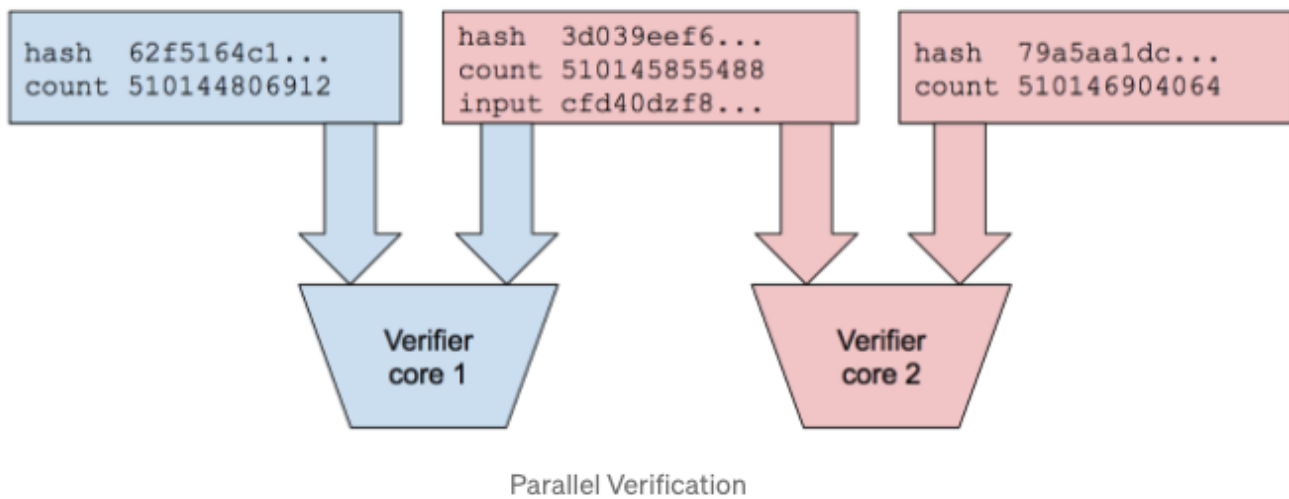


User signed data **0xc0deb10b** is generated with the hash **0xdeadc0de**. It links **0xdeadc0de** to the next generated block **0xbadc0de**. Which prevents a malicious Leader from reordering the events in a hidden side sequence.

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.



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.

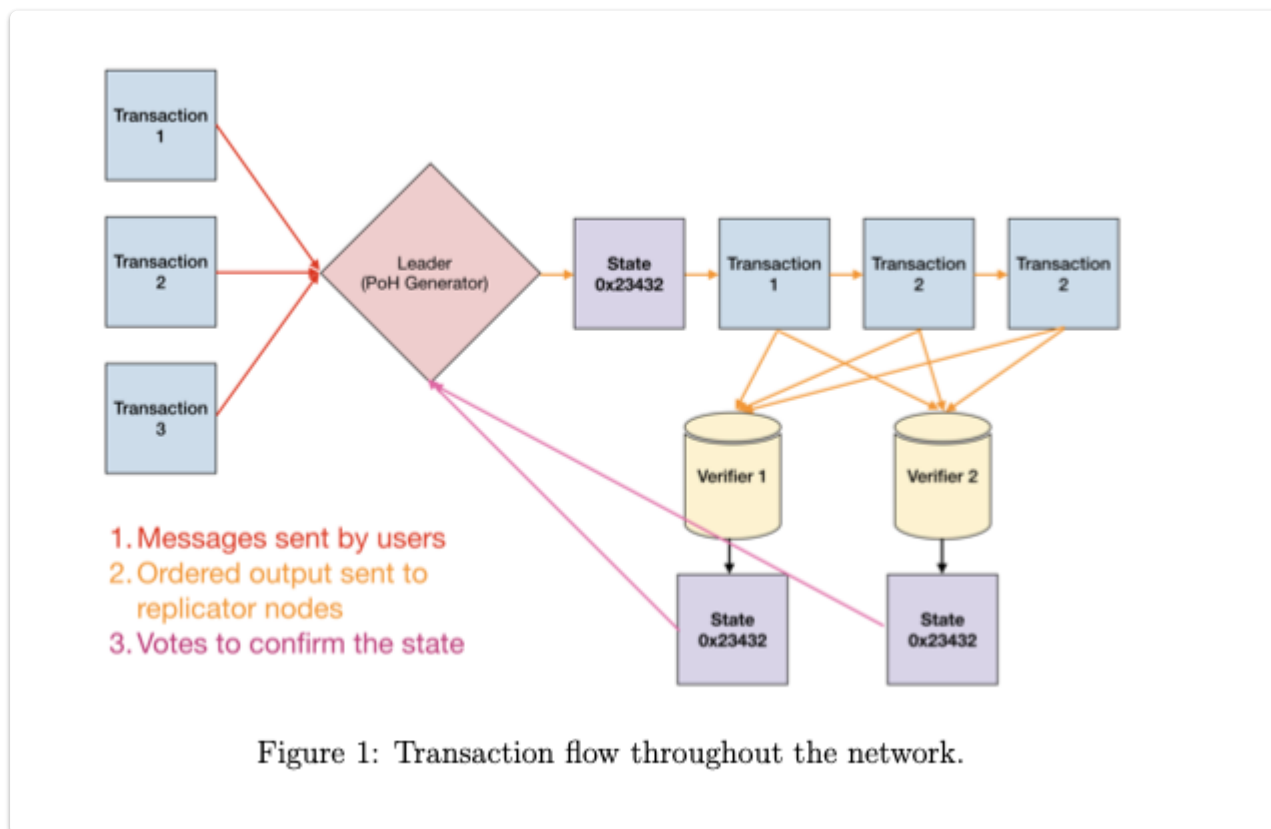
USEFUL ARTICLES

[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.



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 $\frac{2}{3}$ 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.

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 spec'd 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

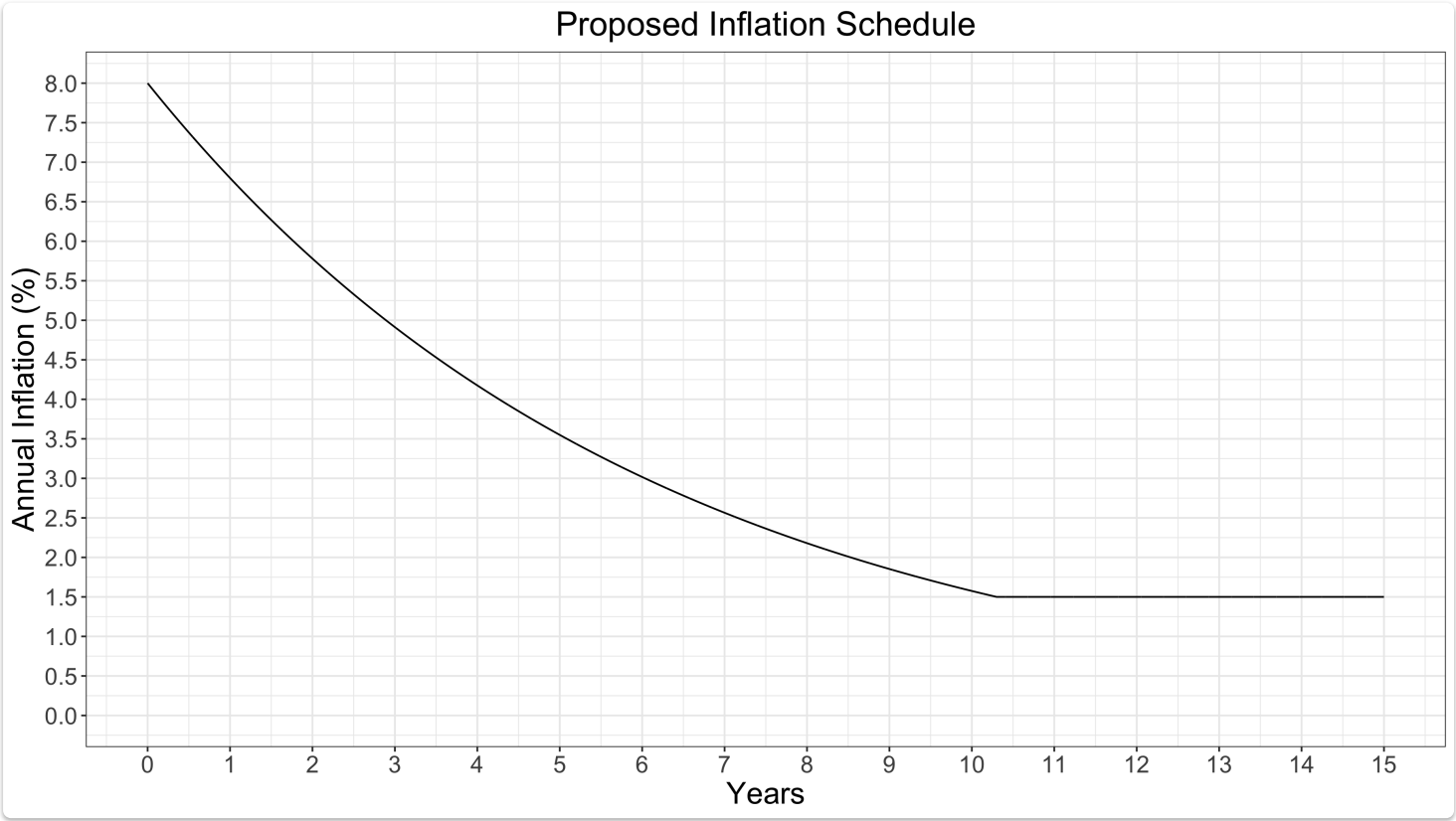
Validators earn SOL in two ways:

- 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.



Network Outages

Date	Duration	Reason
14/09/21	18h	Bots flooding raydium (DDoS)

Date	Duration	Reason
04/12/20	6h	Validator was transmitting multiple blocks for same slot
04/01/22	48h	Excessive duplicate transactions (DDoS)
21/01/22	20h	Bots flooding the network (DDoS)
01/06/22	4h	Network failed to reach consensus.
01/10/22	6h	Degraded Performance.

Solana Status (@SolanaStatus) [September 14, 2021](#)

With Solana's engineers unable to stabilize the network, its validator community opted to coordinate a restart of the network. Solana's community is currently preparing a new release, with further information expected to be released soon.



From [incident report](#)

One of the biggest benefits of blockchains is that, even in complete liveness failure for any reason, the validators are individually responsible for recovering the state and continuing the chain without relying on a trusted third party. On a decentralized network, each validator works to bring it back and has their work guaranteed and verified by everyone else. This was a coordinated effort by the community, not only in creating a patch, but in getting 80% of the network to come to consensus.

From [write up](#)

On 09.14 there was a spike in proposed transactions, on the order of 300k / second. This overloaded the "forwarders" (part of the Gulf Stream protocol that pushes transactions to validators), which resulted in validators crashing from memory overload. To mitigate this, block producers started to automatically propose a number of forks - what they are supposed to do. The challenge was that validators could not agree on a fork, which is a byproduct of the parallelization because validators are trying to reconcile varying states. With this overload, the automatic system of forking came to a halt with <80% consensus on a proposed fork. A bug related to transaction prioritization was found and addressed, and the network was "restarted" after 80% of validators agreed on a state of the chain and did a manual hard fork.

On a technical level, the network is still up -- it's just that nothing important is being processed.

Solution - The basic solution seems to just set a limit to how much can be forwarded when a node gets behind.

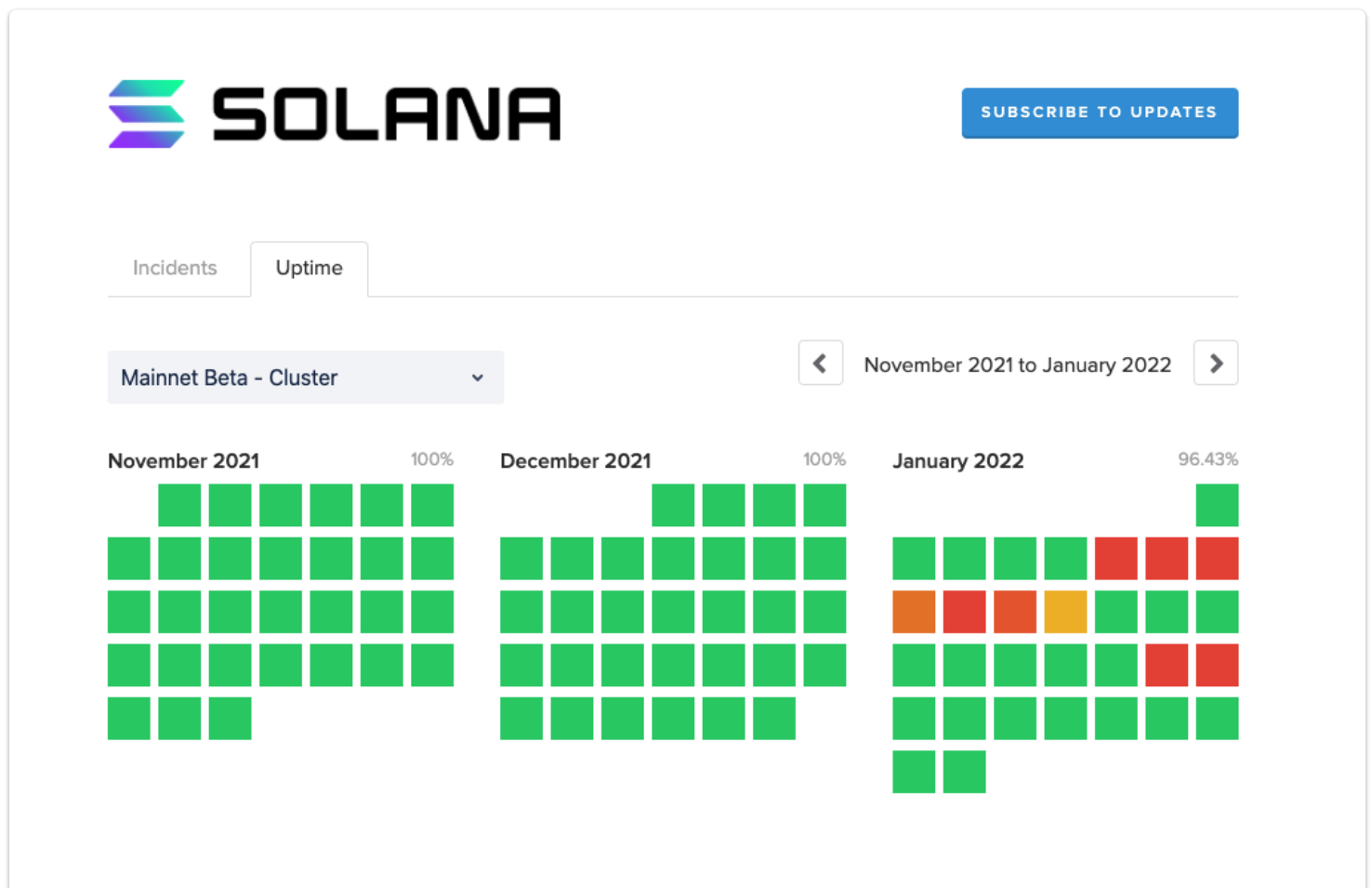
Dec 04 2021

A validator booted up two instances of their machine and it started transmitting multiple different blocks for the same slot, eventually creating 3 different unconfirmed minority partitions of the network. This very specific set of attempted simultaneous block propagations led the network to stall because the partitions could not download different suggested blocks from each other. The stall was due to a known issue in the block propagation/repair path where duplicate blocks for the same slot cannot be repaired between partitions.

Jan 21 2022

"The mainnet-beta cluster is experiencing some performance degradation, we are currently investigating the issue," the team wrote.

Solana Status [Page](#)



Incident [Report](#)

Solana Mainnet Beta encountered a large increase in transaction load which peaked at 400,000 TPS. The current issue experienced by validators is due to excessive duplicate transactions. Engineers have just released 1.8.14, which will attempt to mitigate the worst effects of this issue.

Cryptoeconomics of the attack

If SOL fees are \$0.000005 then 400k transactions ~ \$2

Essentially what it has shown is that it cost \$ ~\$7000 an hour to cause the outage.

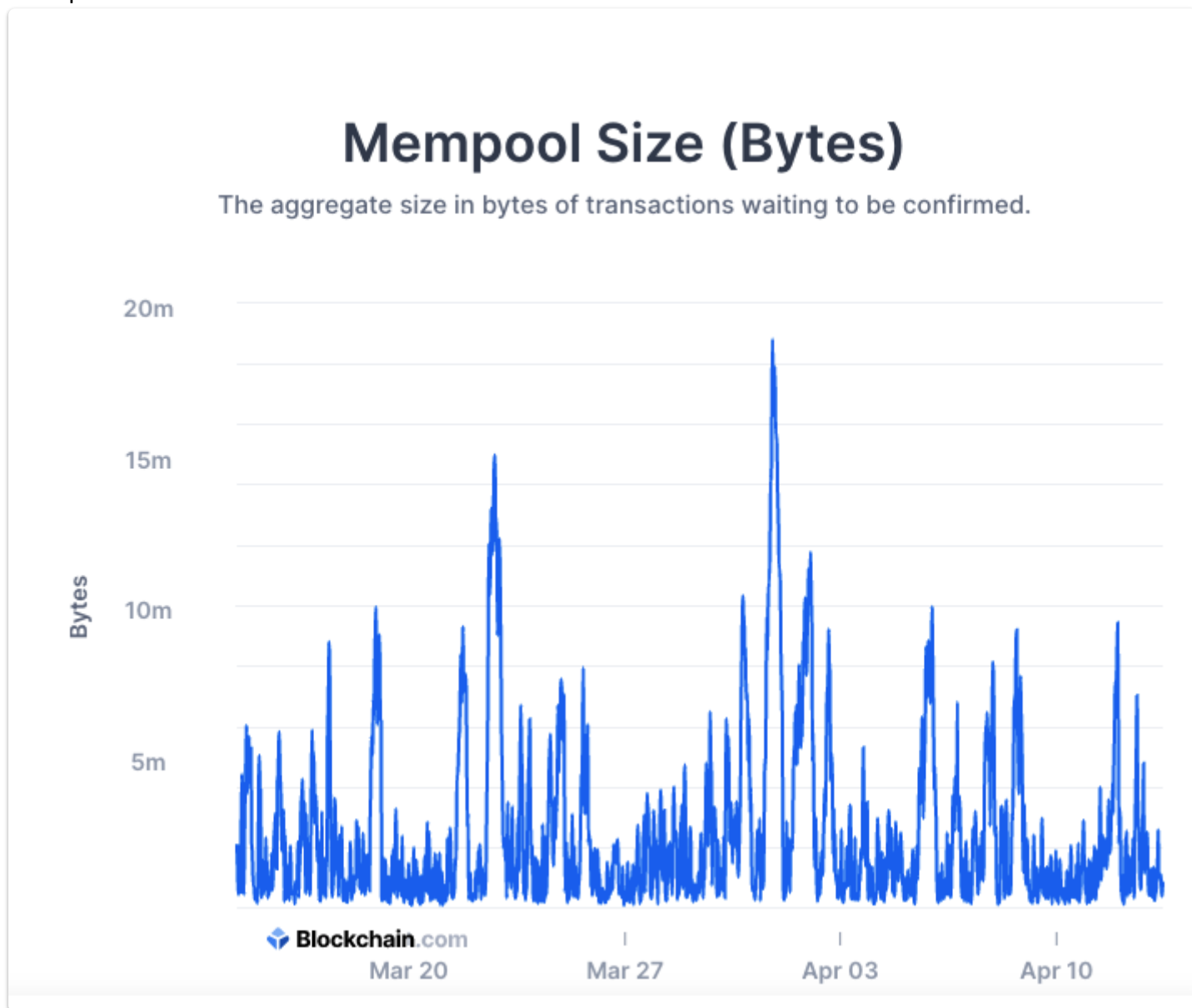
Compare this to the [cost of a 51% attack](#) on PoW chains

(This shows the role of transaction fees in preventing sybil / DOS attacks)

In an interview in Sep 2022 Anatoly Yakovenko said that the network outages had been Solana's "curse," but said the outages have resulted because of the network's low-cost transactions.

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](#)

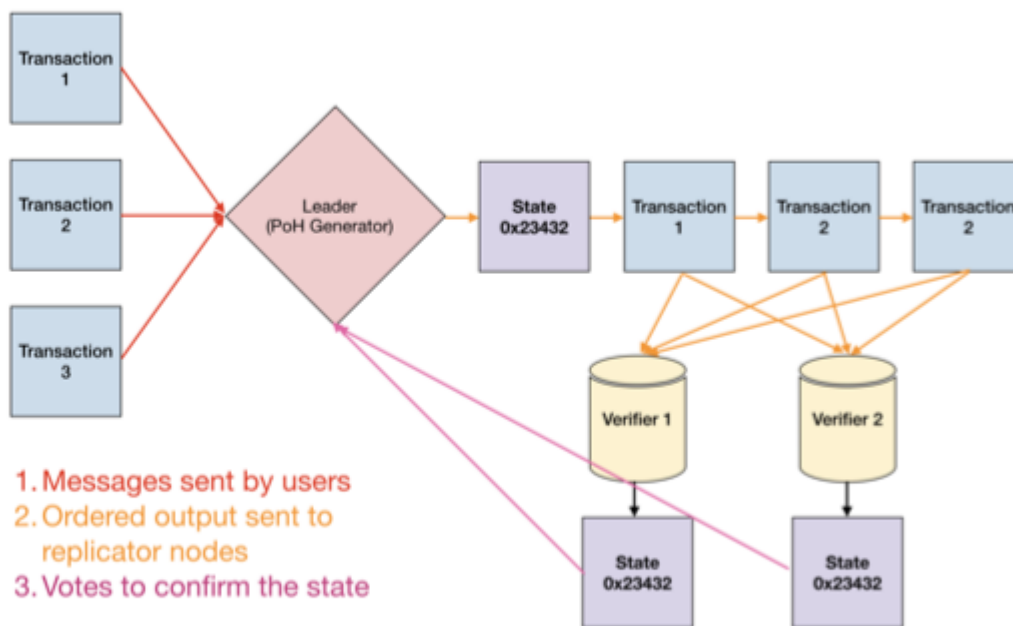


Figure 1: Transaction flow throughout the network.

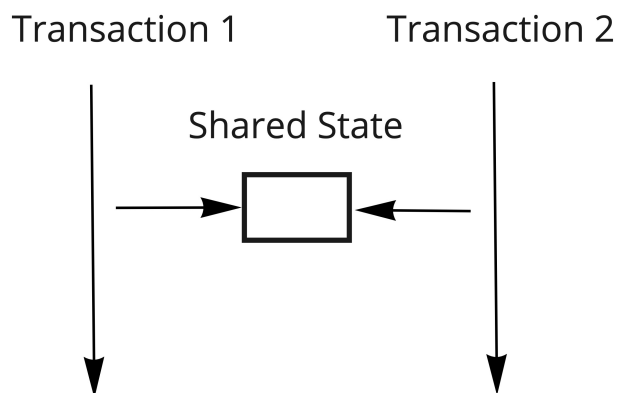
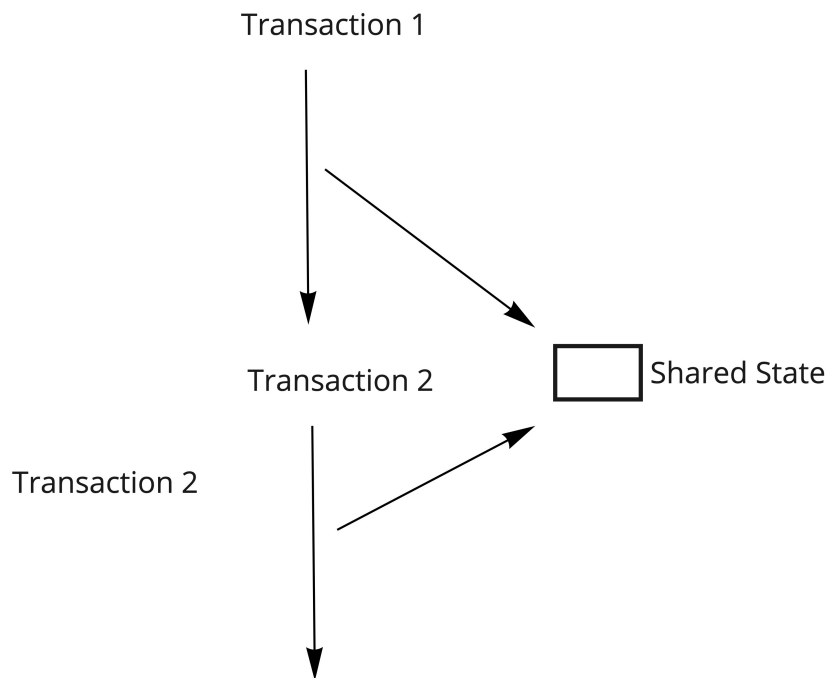
Since every validator knows the order of upcoming leaders, clients and validators forward transactions to the expected leader ahead of time. This allows validators to execute transactions ahead of time, reduce confirmation times, switch leaders faster, and reduce the memory pressure on validators from 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
1	Account A

Transaction	Updates
2	Account B
3	Account A
4	Account C

Cryptoeconomics

We have had decentralised and fault tolerant systems before, but what sets blockchains apart is cryptoeconomics

"Cryptoeconomic approaches combine cryptography and economics to create robust decentralized P2P networks that thrive over time despite adversaries attempting to disrupt the network."

From [Cryptoeconomics 101](#)

From Internet Policy Review : Cryptoeconomics

The term cryptoeconomics entered casual usage in the formative years of the Ethereum developer community in 2014-5. The phrase is typically attributed to Vitalik Buterin with the earliest public usage being in a 2015 talk by Vlad Zamfir entitled "What is Cryptoeconomics"

In a system what do we want to encourage ?

- Trusted execution
- Open Access
- Fast Finality
- Decentralized Control (there is no central authority which controls the protocol and the network)
- Inexpensiveness (encourages many transactions)

Things we want to avoid:

- Safety Failure (e.g. someone steals your tokens)
- Censorship (e.g. someone decides that a certain group of people should not be allowed to transact)
- Slow finality (opposite of fast finality)
- Centralized Control (opposite decentralized control)
- Expensiveness

Properties required for a cryptocurrency

- Eventual consensus. At any time, all compliant nodes agree upon a prefix of what will become the eventual "true" blockchain.
- Exponential convergence. Users should have high confidence that a simple "k confirmations" rule will ensure their transactions are settled permanently.
- Liveness. New blocks will continue to be added and valid transactions with appropriate fees will be included in the blockchain within a reasonable amount of time.
- Correctness. All blocks in the chain with the most cumulative proof of work will only include valid transactions.
- Fairness. A proposer with X% of the network's total computational power will produce approximately X% of blocks.

"Whatever your rules are for rewarding, penalizing inside of the mechanism, they have to be specified as a piece of Solidity code, Vyper code, whatever programming language you're using in that set. That's a much tighter constraint than policymakers writing laws have."

"Another one is, of course, that all of the actors are anonymous, and what that means in practice is that you cannot drag people's utility down below zero. If I have 70 ether, and I put that 70 ether into a mechanism, the worst thing you can do to me is you can take away that 70 ether.

You cannot throw me in jail. You cannot socially ostracize me so I can't earn any money again because I can always just switch identities. But to the extent that I'm willing to lock that ether up and make it vulnerable to a mechanism, then you have the ability to motivate me to that extent."

Incentives

"An incentive is any design element of a system that influences the behaviour of system participants by changing the relative costs and benefits of choices those participants may make."

From [Why incentives matter](#)

We can incentivise through

- Rewards
 - For example the block reward, or transaction fee
 - Privileges within the system
- Punishments
 - Direct : Loss of deposit (see proof of stake consensus mechanism)
 - Indirect : Loss of potential reward / privileges

Economics of Blockchain

From [Simple Economics of the blockchain](#)

"... two key costs affected by blockchain technology – the cost of verification of state, and the cost of networking – change the types of transactions that can be supported in the economy."

"Whereas the reduction in the cost of verification is what allows Bitcoin to settle transactions without an intermediary, the reduction in the cost of networking is what allowed its ecosystem to scale in the first place"

Blockchain Governance

"The greatest challenge that new blockchains must solve isn't speed or scaling, it's governance"

Kai Sedgwick - [Why Governance is the Greatest Problem for Blockchains To Solve](#)**

It is useful to think of governance in the following areas

- Consensus
 - Who is involved and how do they come to consensus ?

- Information
How does relevant information reach the participants ?
- Incentives
How are the incentives aligned to ensure
- Correct Behaviour
There is a sufficient level of participation
- Procedures
In a decentralised system how are
Proposals made
Votes submitted
Consensus reached

Types of Governance

1. Off chain

The mechanism to change the protocol are external to the system

The process is often

- ad hoc
- may be poorly specified
- communication and coordination can be problematic

Developers may have a key role in deciding and implementing changes to the protocol

2. On chain

The mechanism to change the protocol is part of the protocol

Typically participants can vote to accept or reject proposals to upgrade the protocol or some aspects of the system

Coordination and communication is usually more efficient than in off chain solutions

In reality, there is often a mixture of both

Governance in Solana

See [Docs](#)

The Feature Proposal Program provides a workflow for activation of Solana network features through community vote based on validator stake weight.

Community voting is accomplished using [SPL Tokens](#). Tokens are minted that represent the total active stake on the network, and distributed to all validators based on their stake. Validators vote for feature activation by transferring their vote tokens to a predetermined address. Once the vote threshold is met the feature is activated.

The Feature Proposal Program provides an additional mechanism over these runtime feature activation primitives to permit feature activation by community vote when appropriate.

Lifecycle

1. Implement the feature

The developers change the runtime to include a possible new feature

2. Initiate the voting

3. Tally the Votes - if the vote succeeds the feature is implemented.

Resources

Solana [Documentation](#)

Solana [White Paper](#)

[Article - Functional Programming](#)

[Why functional programming matters](#)