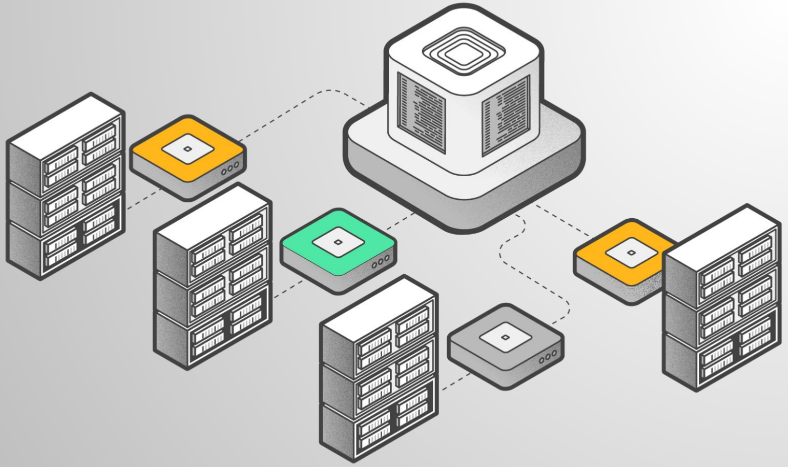
Scaling BlockChain by Sharding



Debdoot Roy Chowdhury (23M0675)

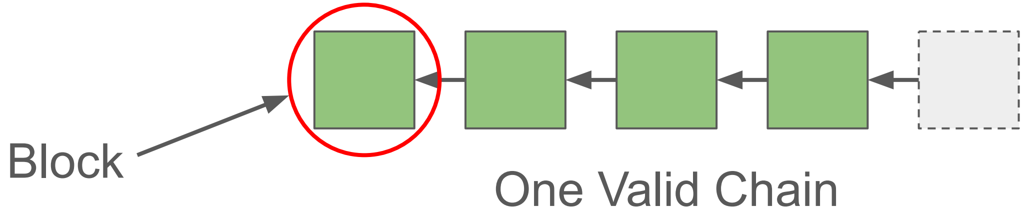
Guided By: Prof. Vinay J. Ribeiro

What is a BlockChain?

Blockchain is a decentralized, distributed, shared and immutable digital ledger that records data across many peers.

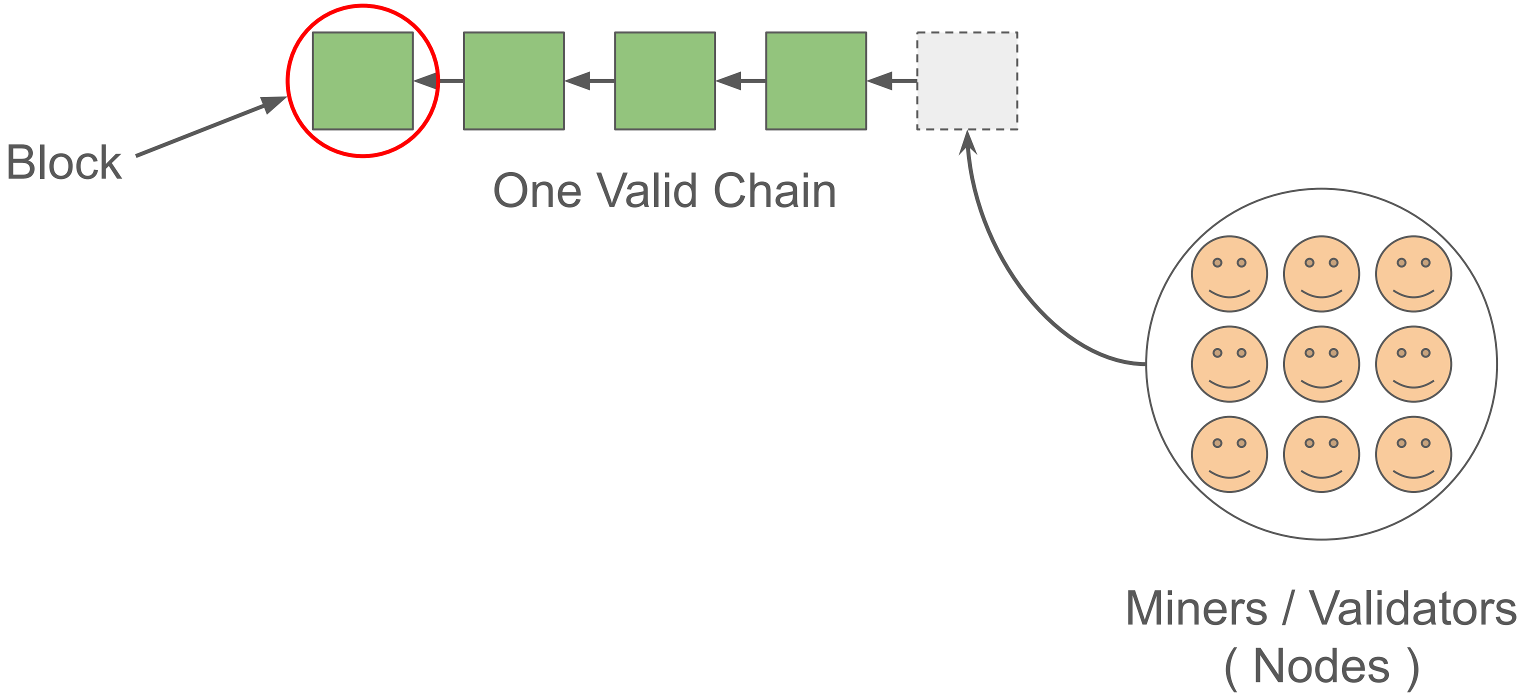
What is a BlockChain?

Blockchain is a decentralized, distributed, shared and immutable digital ledger that records data across many peers.



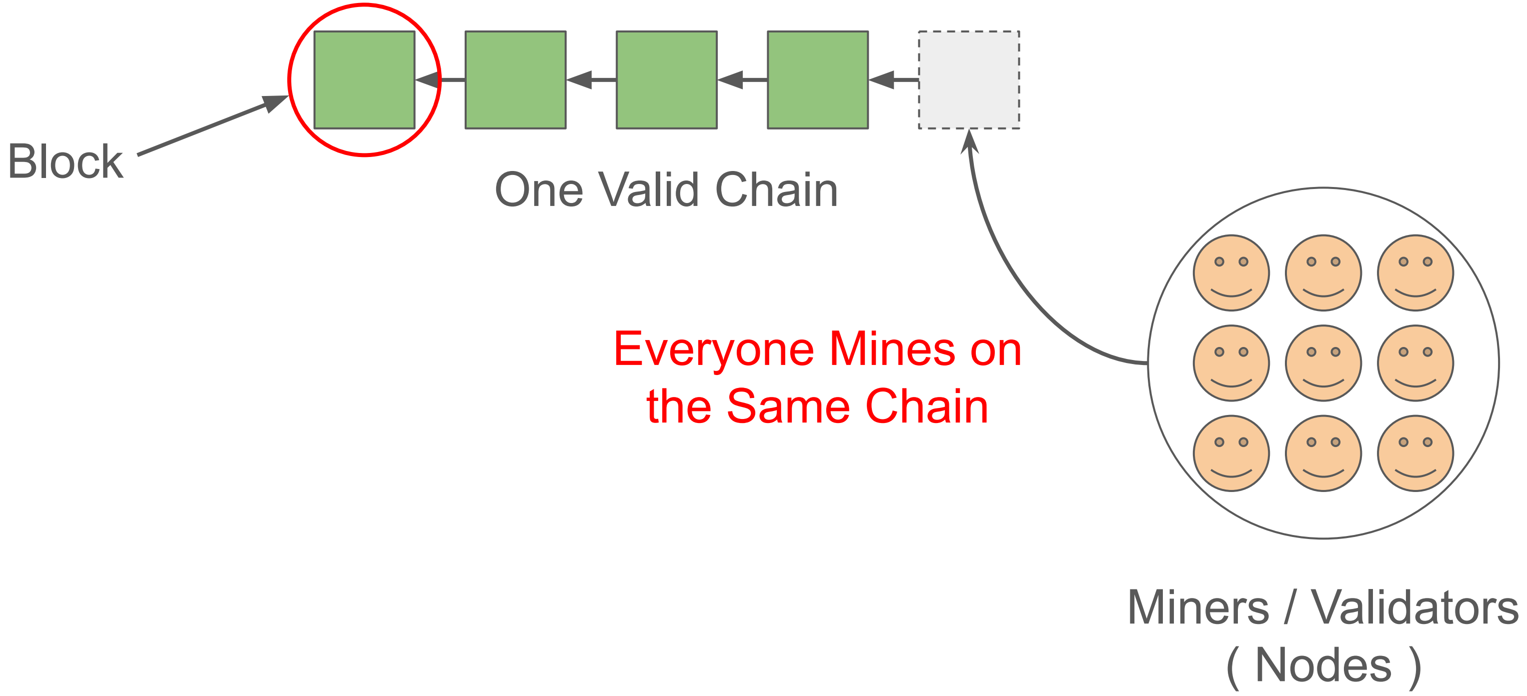
What is a BlockChain?

Blockchain is a decentralized, distributed, shared and immutable digital ledger that records data across many peers.



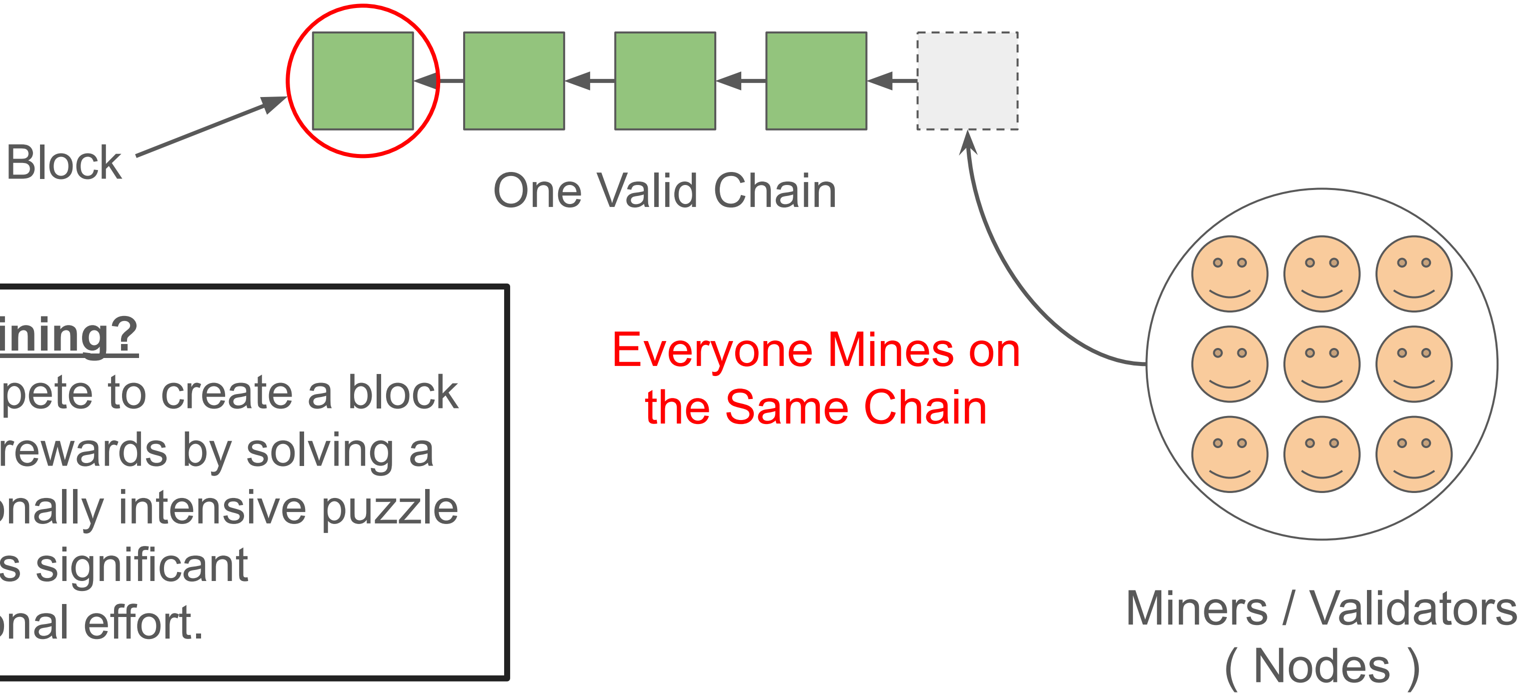
What is a BlockChain?

Blockchain is a decentralized, distributed, shared and immutable digital ledger that records data across many peers.



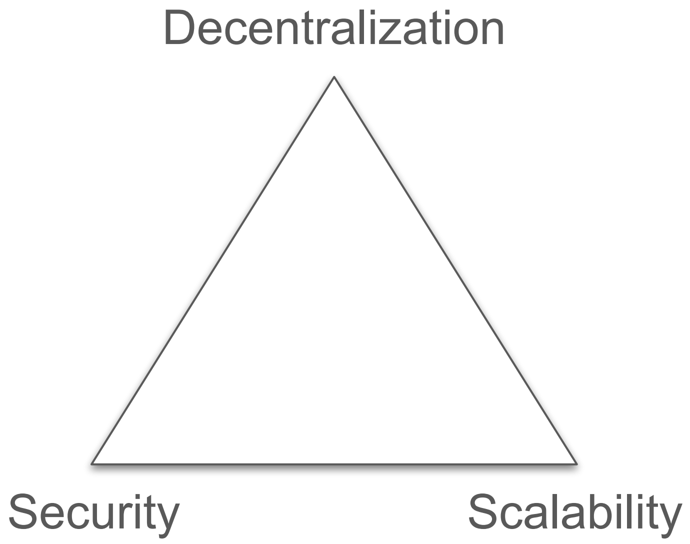
What is a BlockChain?

Blockchain is a decentralized, distributed, shared and immutable digital ledger that records data across many peers.

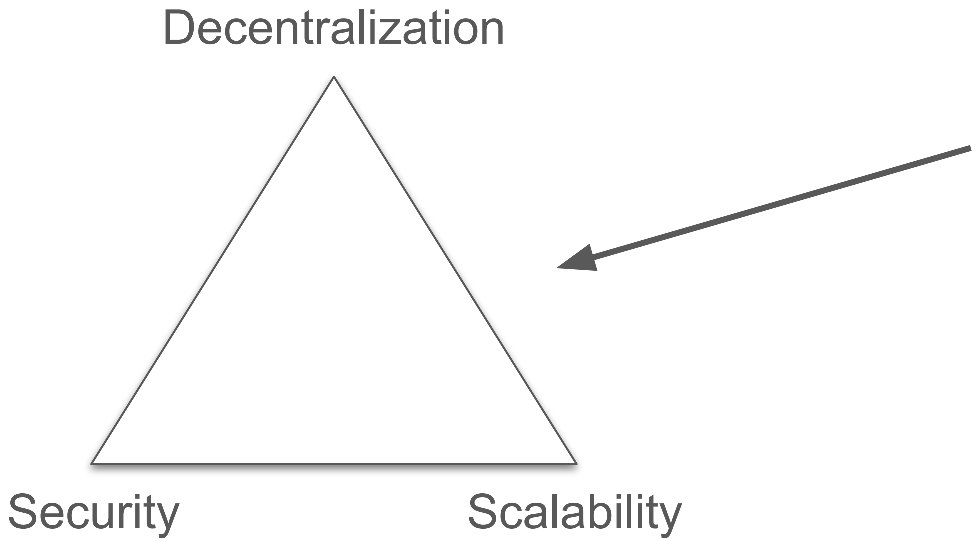

Block
One Valid Chain
Everyone Mines on the Same Chain
Miners / Validators (Nodes)

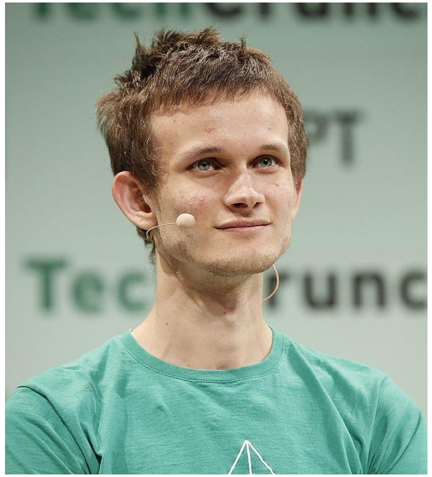
What is Mining?
Users compete to create a block for getting rewards by solving a computationally intensive puzzle which takes significant computational effort.

The Scalability Trilemma



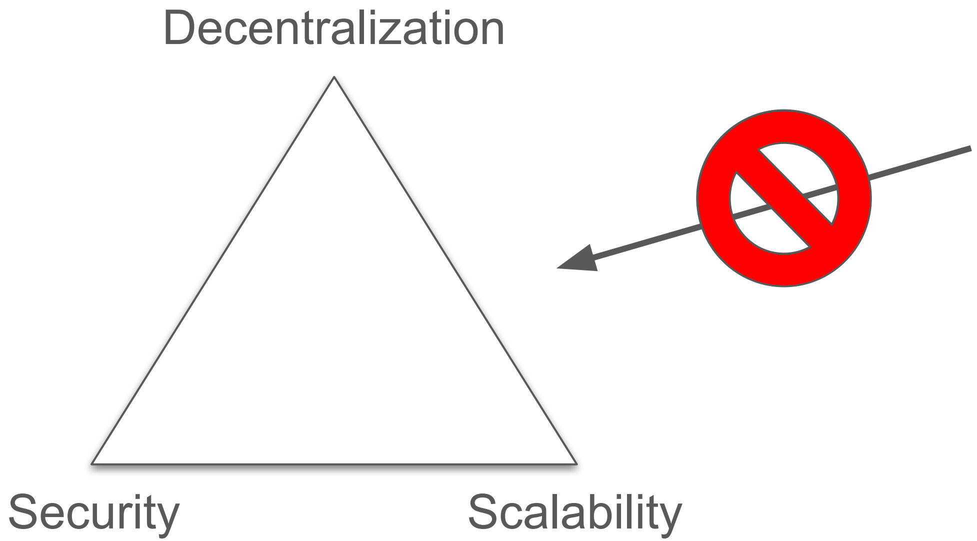
The Scalability Trilemma

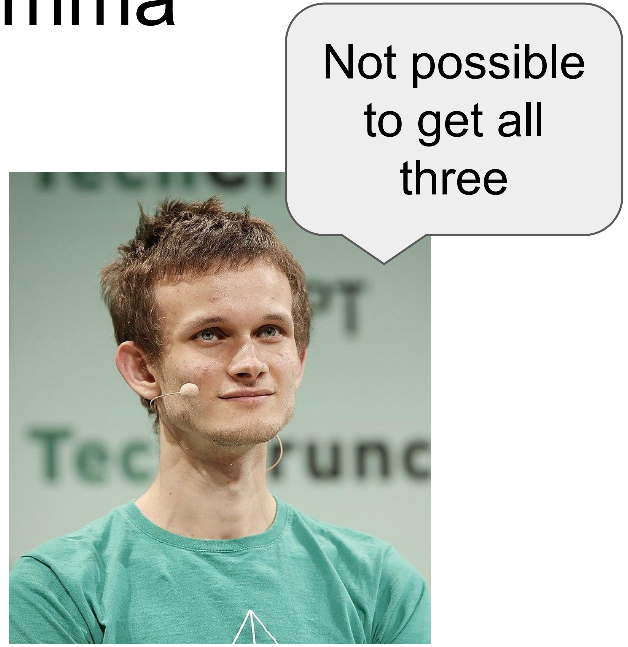




Vitalik Buterin
Founder of ETH

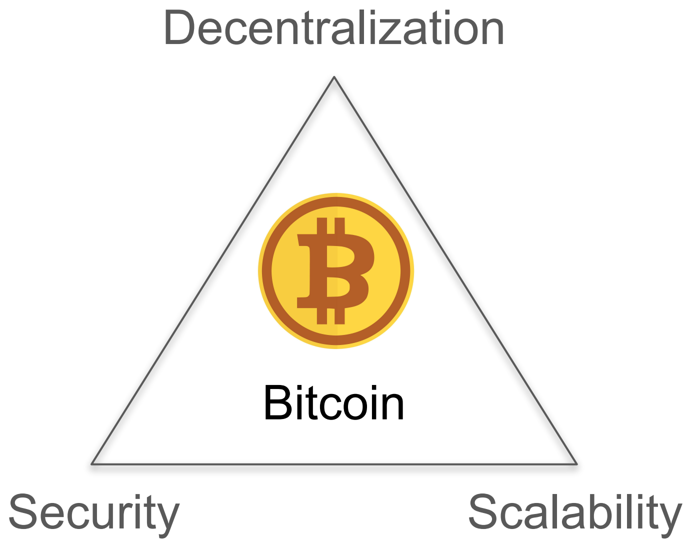
The Scalability Trilemma



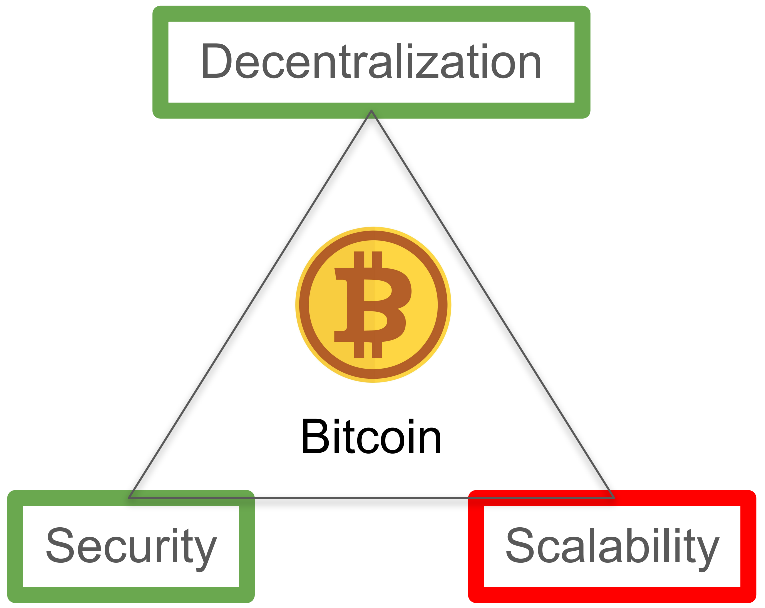


Vitalik Buterin
Founder of ETH

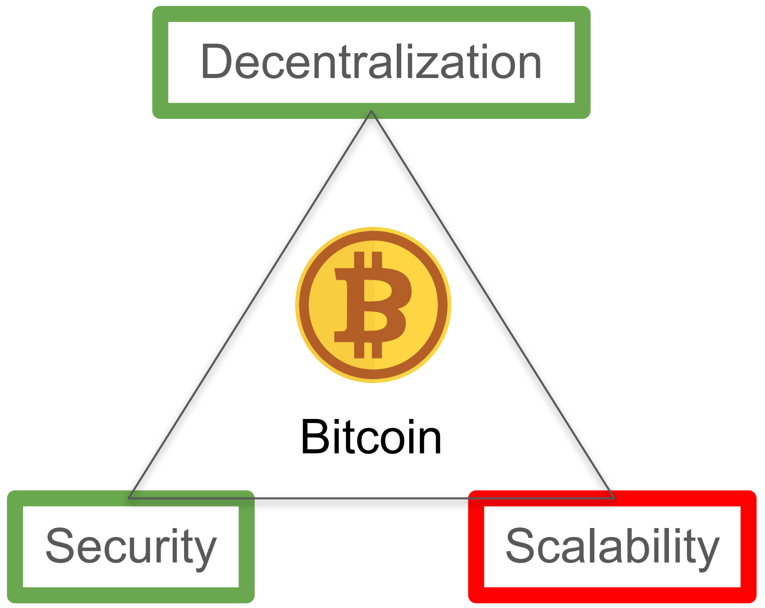
The Scalability Trilemma

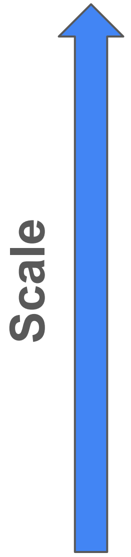


The Scalability Trilemma

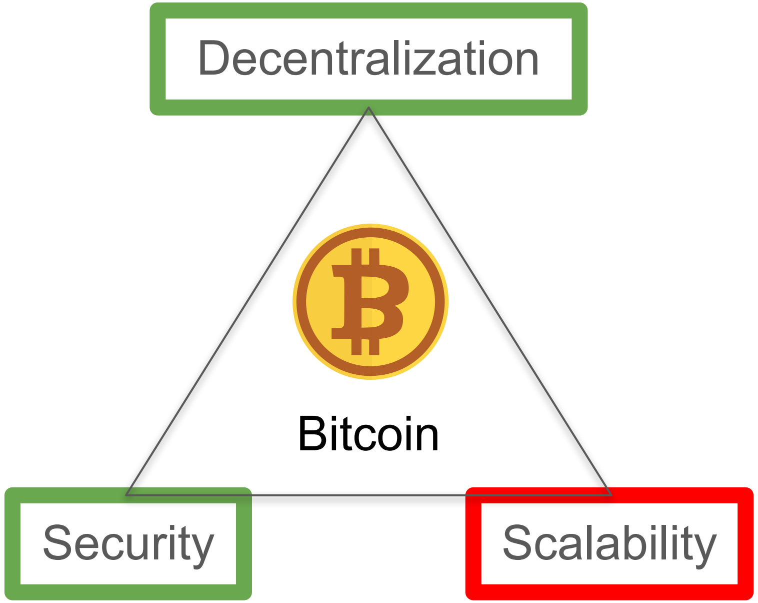


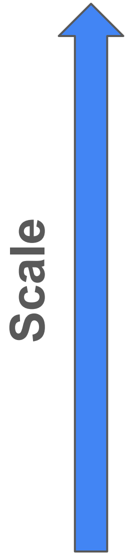
The Scalability Trilemma





The Scalability Trilemma





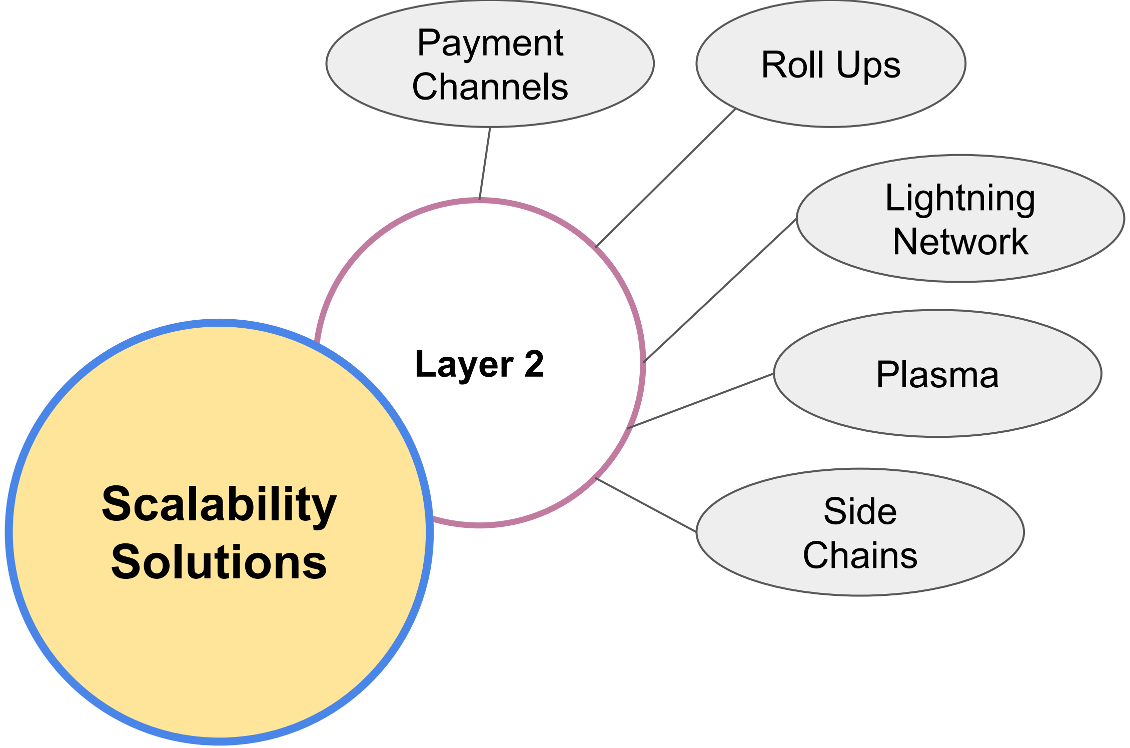
NEED

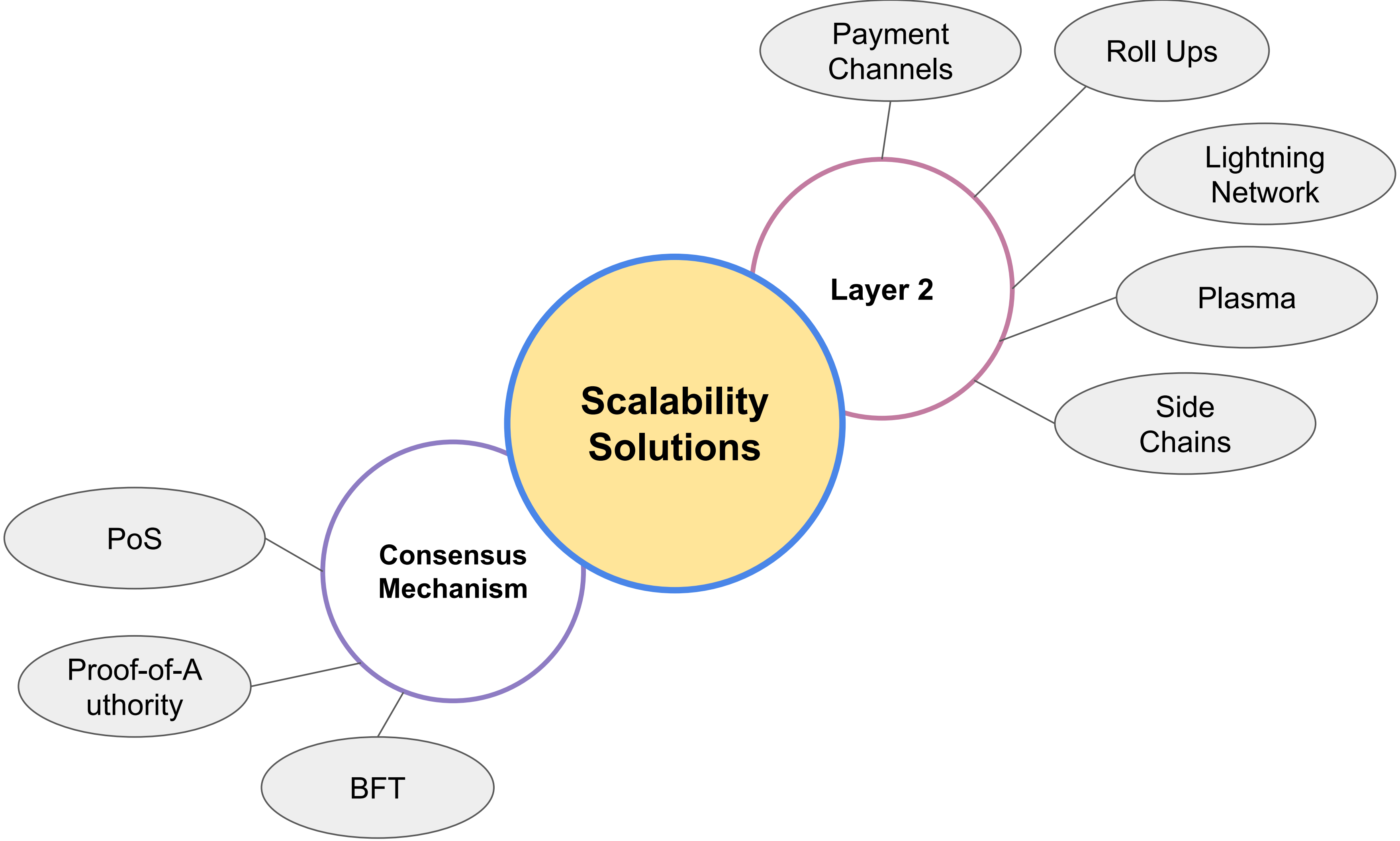
More Speed

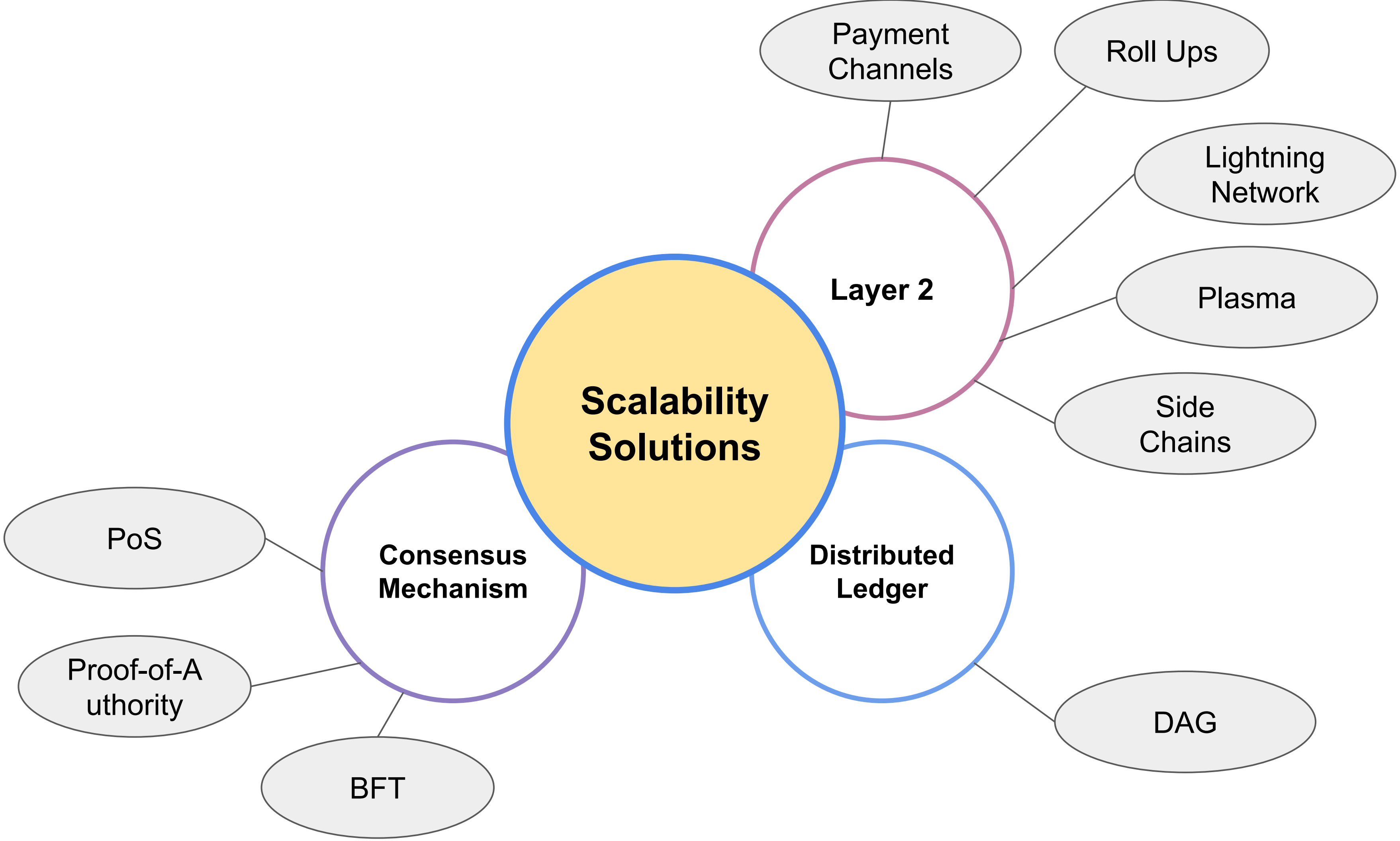
Less Storage Overhead

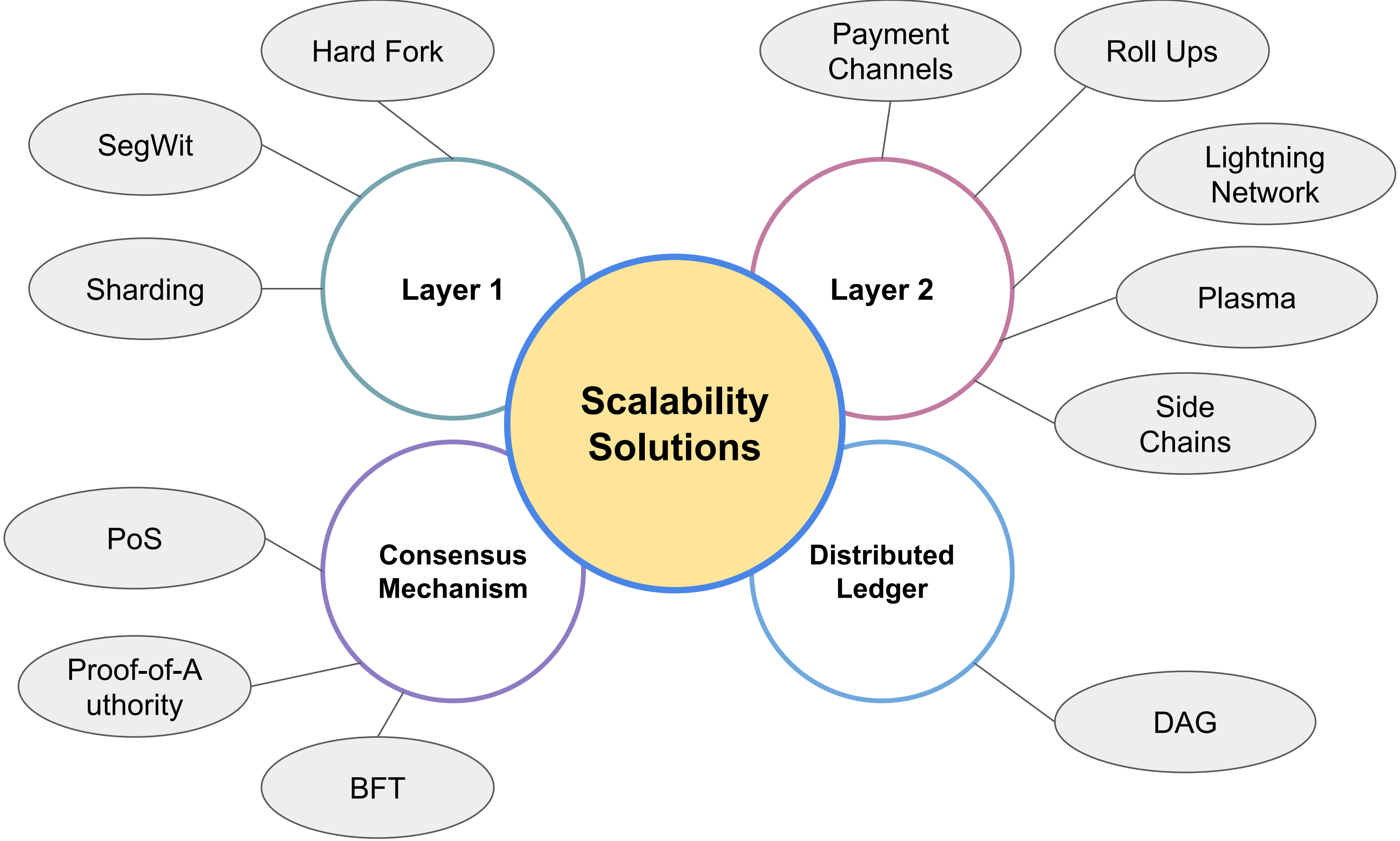
More Miners = More Speed

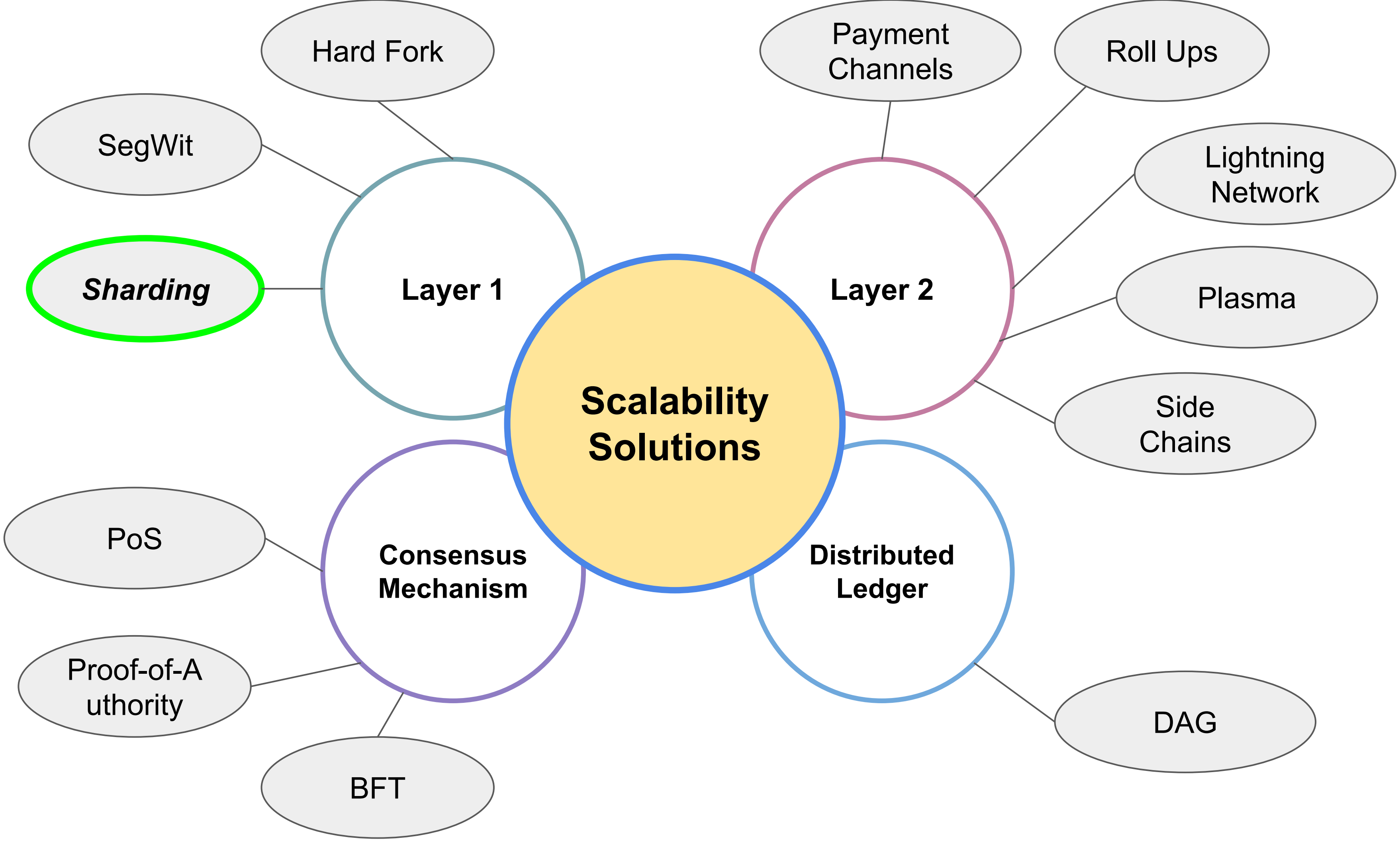
Scalability Solutions


Scalability Solutions
Layer 2
Payment Channels
Roll Ups
Lightning Network
Plasma
Side Chains


Scalability Solutions
Layer 2
Payment Channels
Roll Ups
Lightning Network
Plasma
Side Chains
Consensus Mechanism
PoS
Proof-of-Authority
BFT


Scalability Solutions
Layer 2
Payment Channels
Roll Ups
Lightning Network
Plasma
Side Chains
Consensus Mechanism
PoS
Proof-of-Authority
BFT
Distributed Ledger
DAG


Scalability Solutions
Layer 1
Hard Fork
SegWit
Sharding
Layer 2
Payment Channels
Roll Ups
Lightning Network
Plasma
Side Chains
Consensus Mechanism
PoS
Proof-of-Authority
BFT
Distributed Ledger
DAG

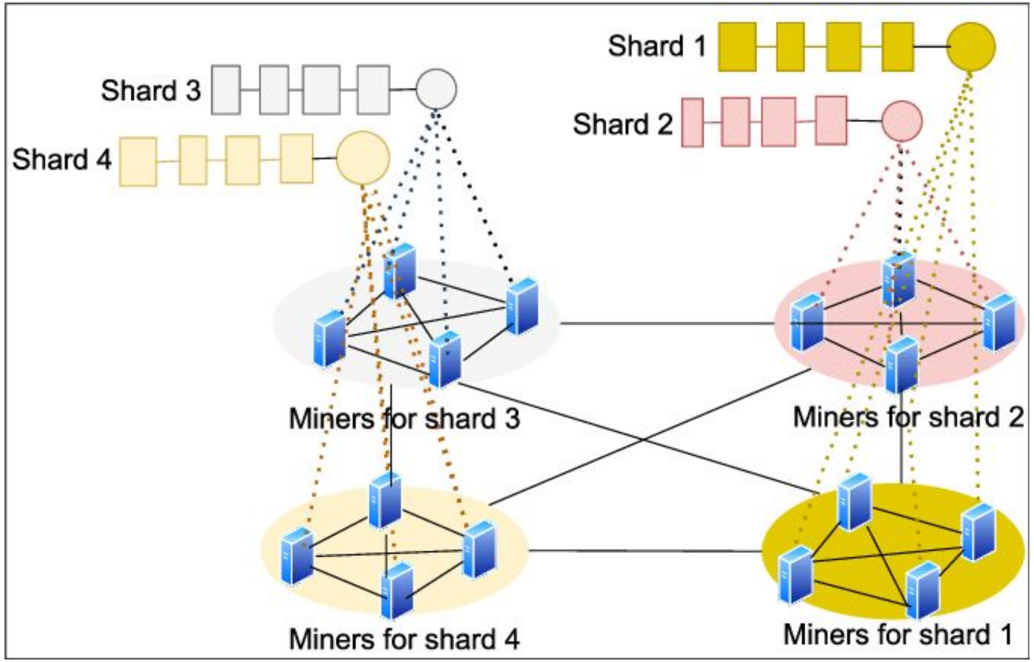

Hard Fork
SegWit
Sharding
Layer 1
Payment Channels
Roll Ups
Lightning Network
Plasma
Layer 2
Side Chains
Scalability Solutions
PoS
Consensus Mechanism
Proof-of-Authority
BFT
Distributed Ledger
DAG

What is Sharding?

Partitions the network into smaller groups called ***shards***, allowing *parallel processing* of transactions to improve overall throughput.

What is Sharding?

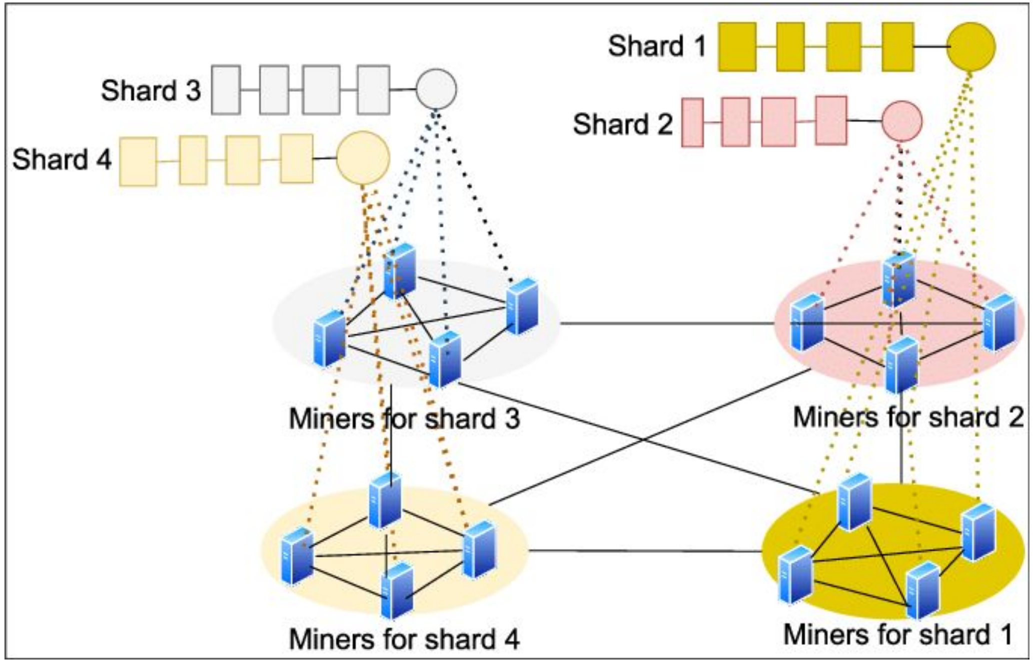
Partitions the network into smaller groups called ***shards***, allowing *parallel processing* of transactions to improve overall throughput.

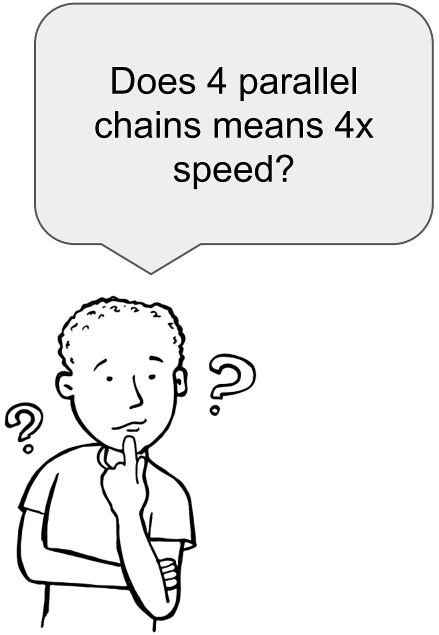


Uddin, Md Ashraf, et al. "A survey on the adoption of blockchain in iot: Challenges and solutions."

What is Sharding?

Partitions the network into smaller groups called ***shards***, allowing *parallel processing* of transactions to improve overall throughput.

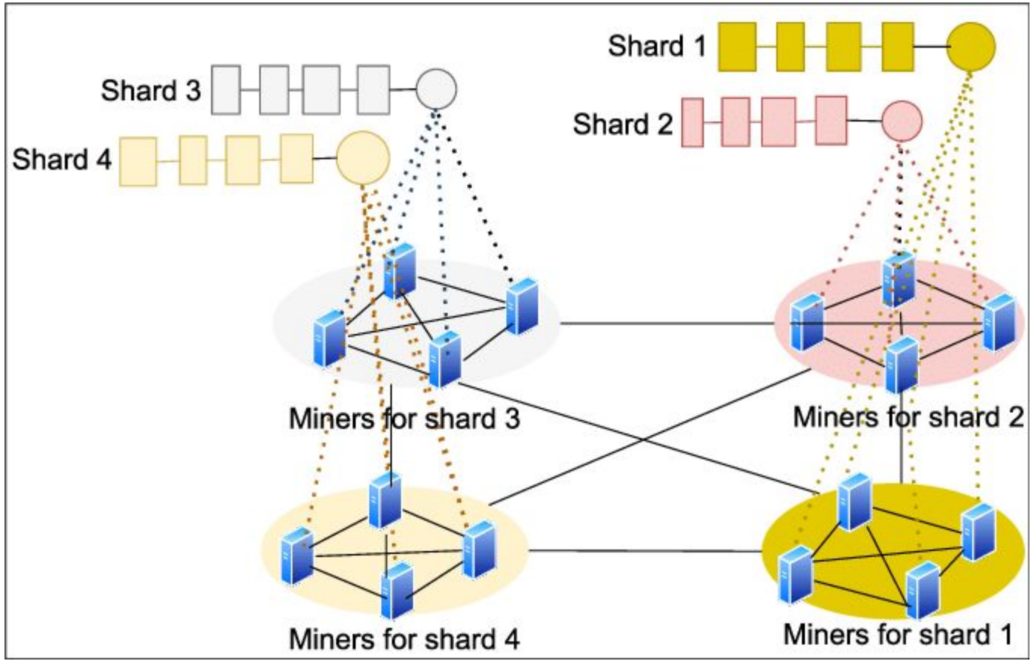




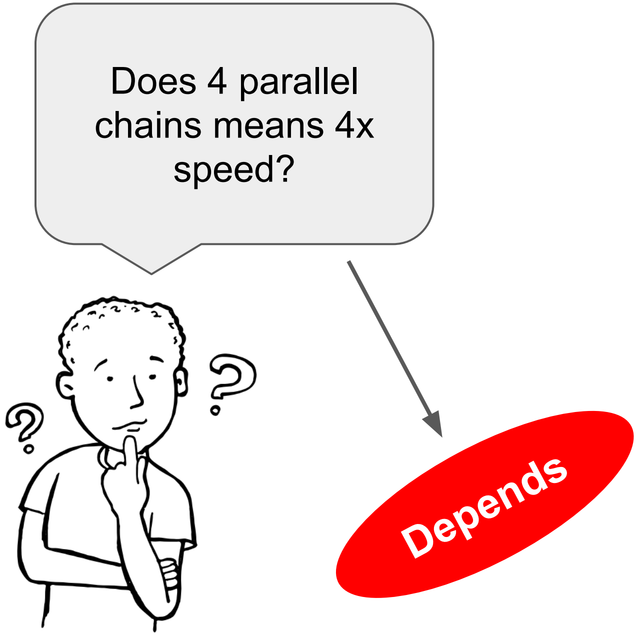
Uddin, Md Ashraf, et al. "A survey on the adoption of blockchain in iot: Challenges and solutions."

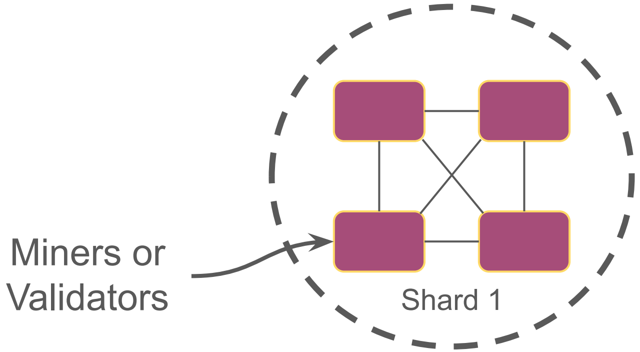
What is Sharding?

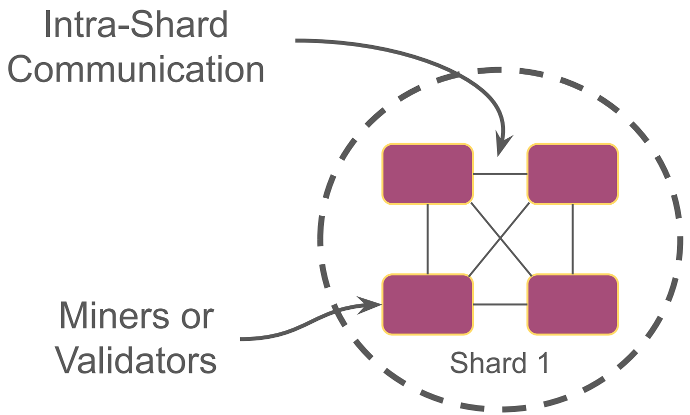
Partitions the network into smaller groups called ***shards***, allowing *parallel processing* of transactions to improve overall throughput.

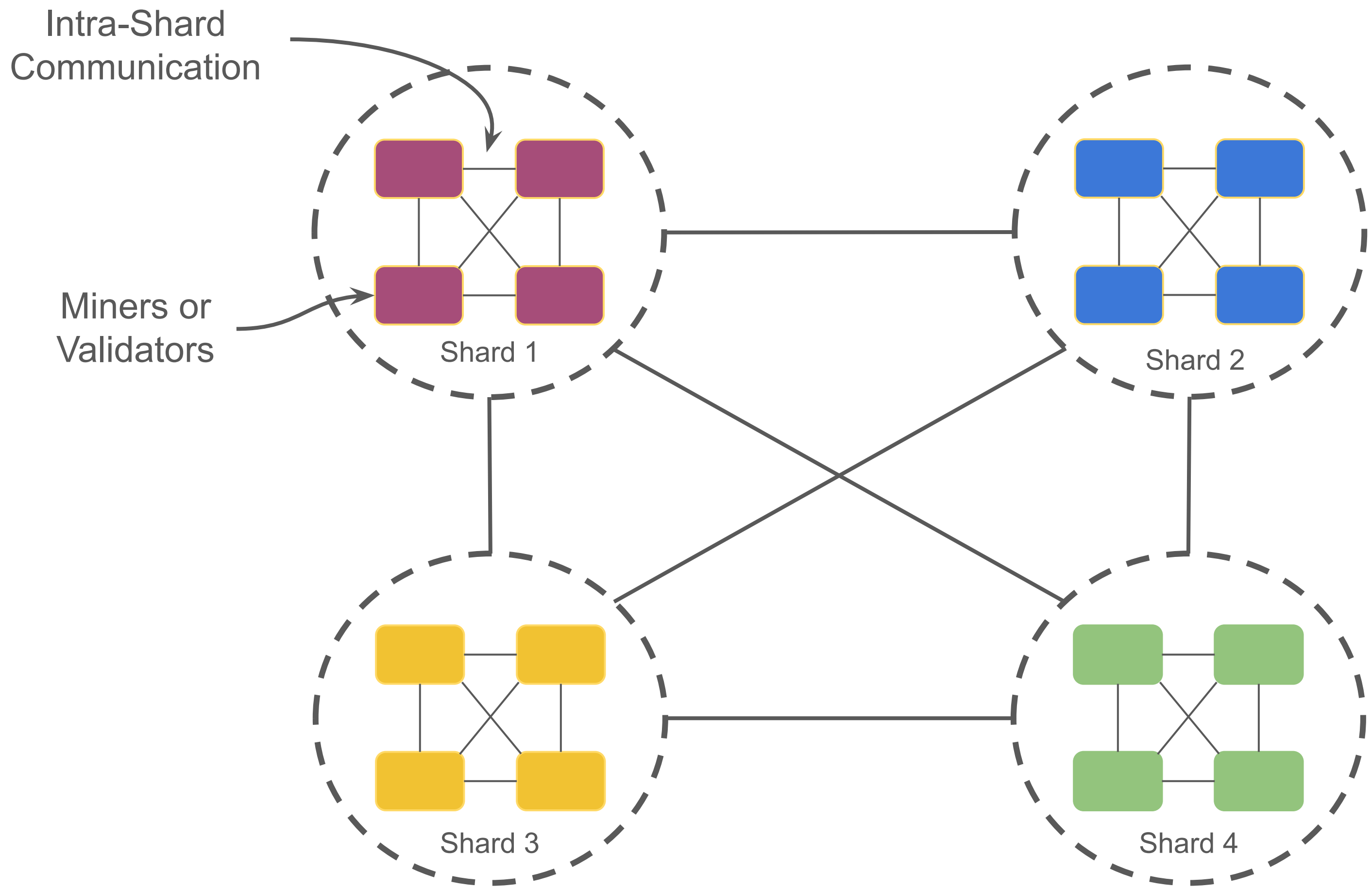


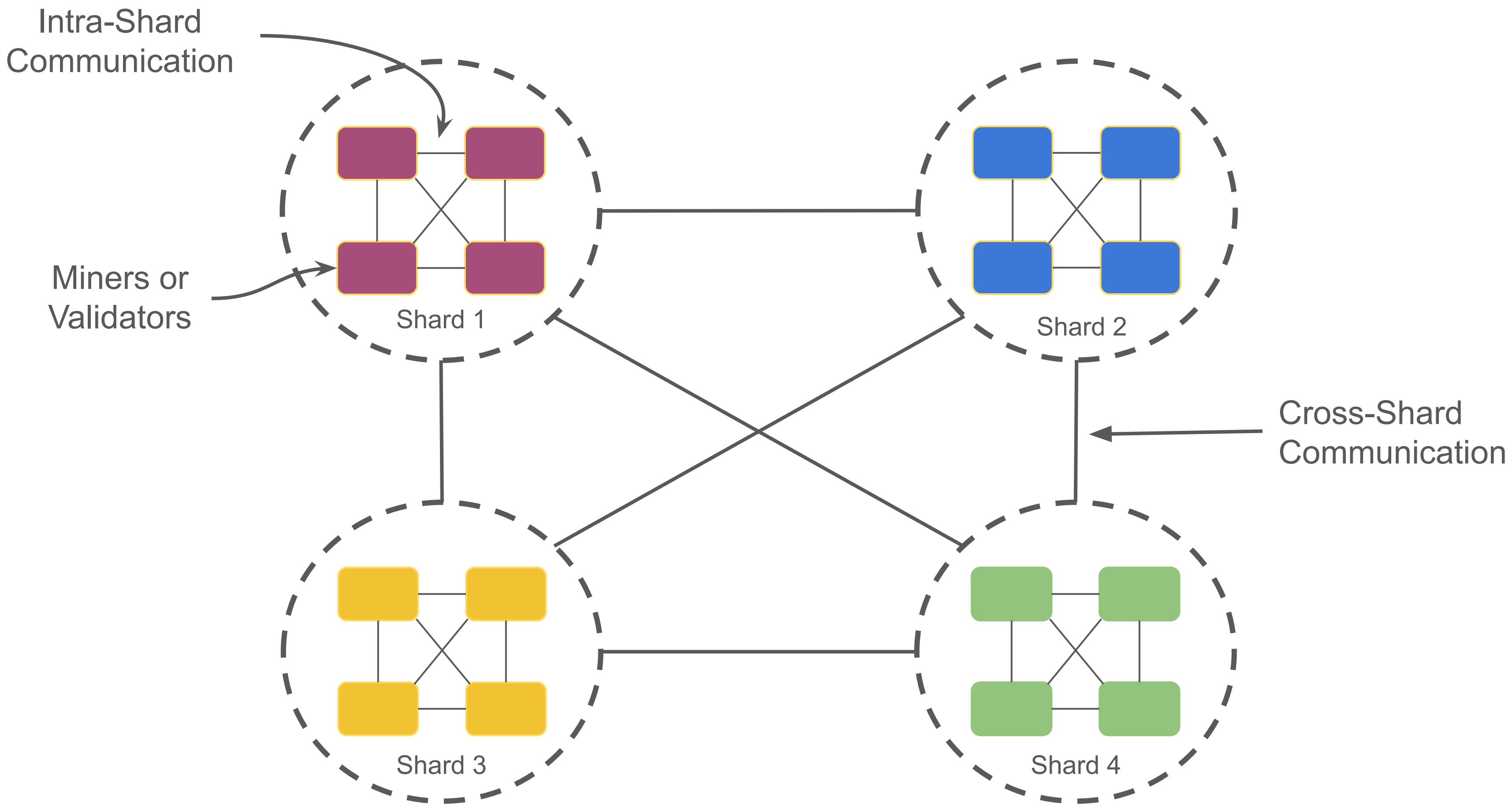
Uddin, Md Ashraf, et al. "A survey on the adoption of blockchain in iot: Challenges and solutions."



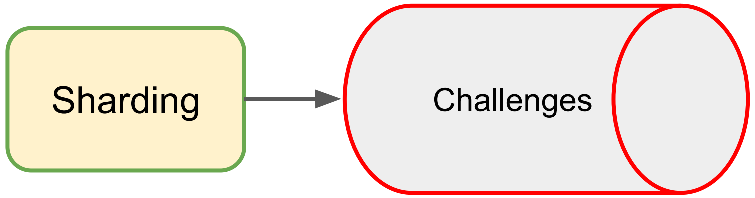

Miners or Validators
Shard 1


Intra-Shard Communication
Miners or Validators
Shard 1

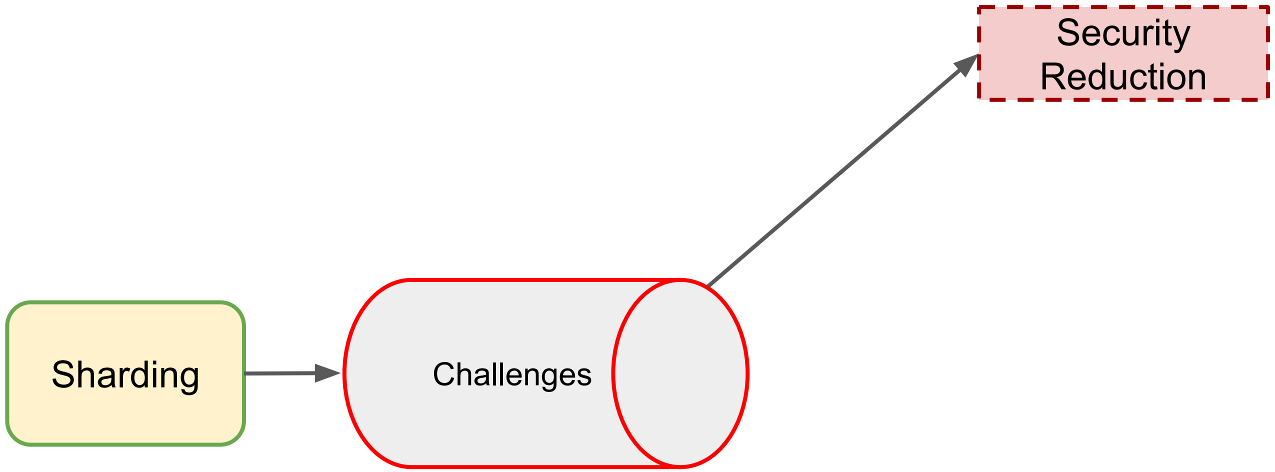

Intra-Shard Communication
Miners or Validators
Shard 1
Shard 2
Shard 3
Shard 4


Intra-Shard Communication
Miners or Validators
Shard 1
Shard 2
Cross-Shard Communication
Shard 3
Shard 4

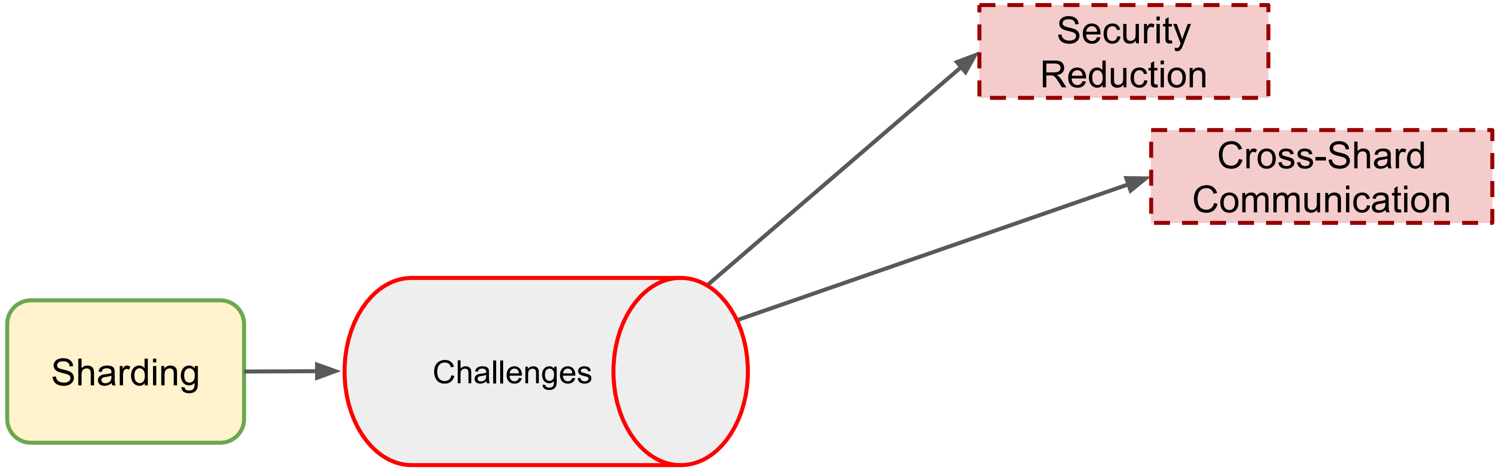
Challenges in Sharding



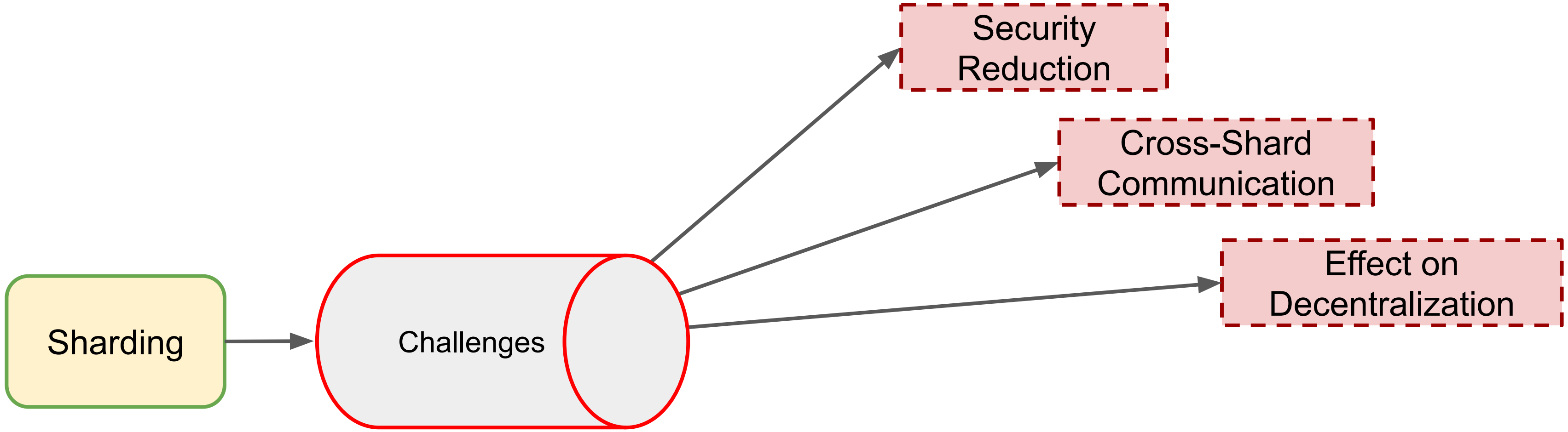
Challenges in Sharding



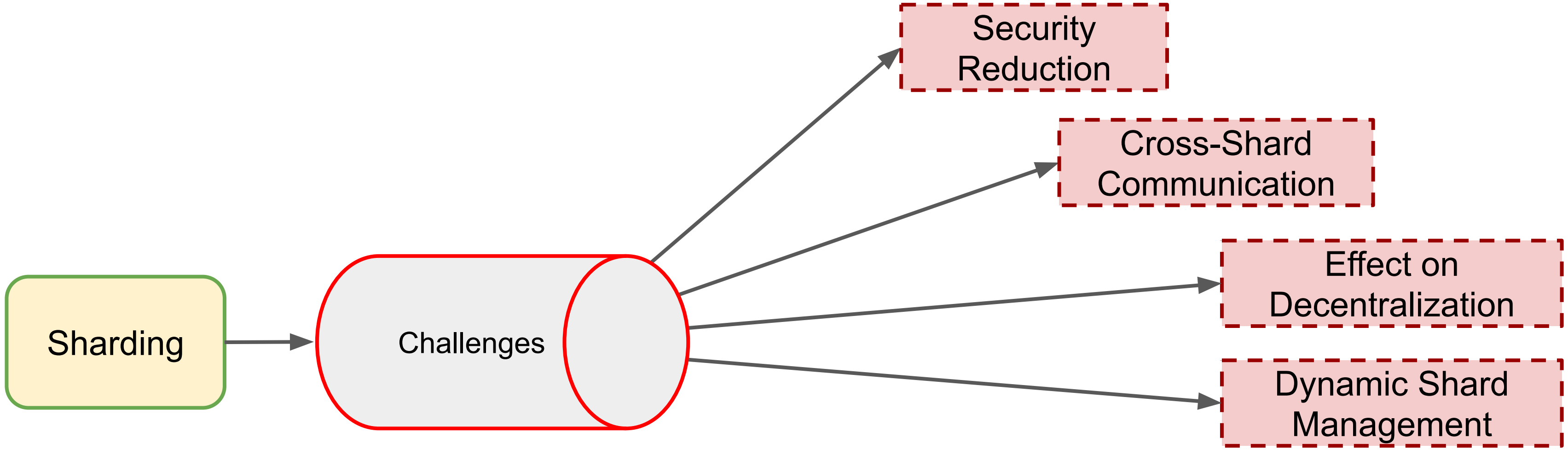
Challenges in Sharding



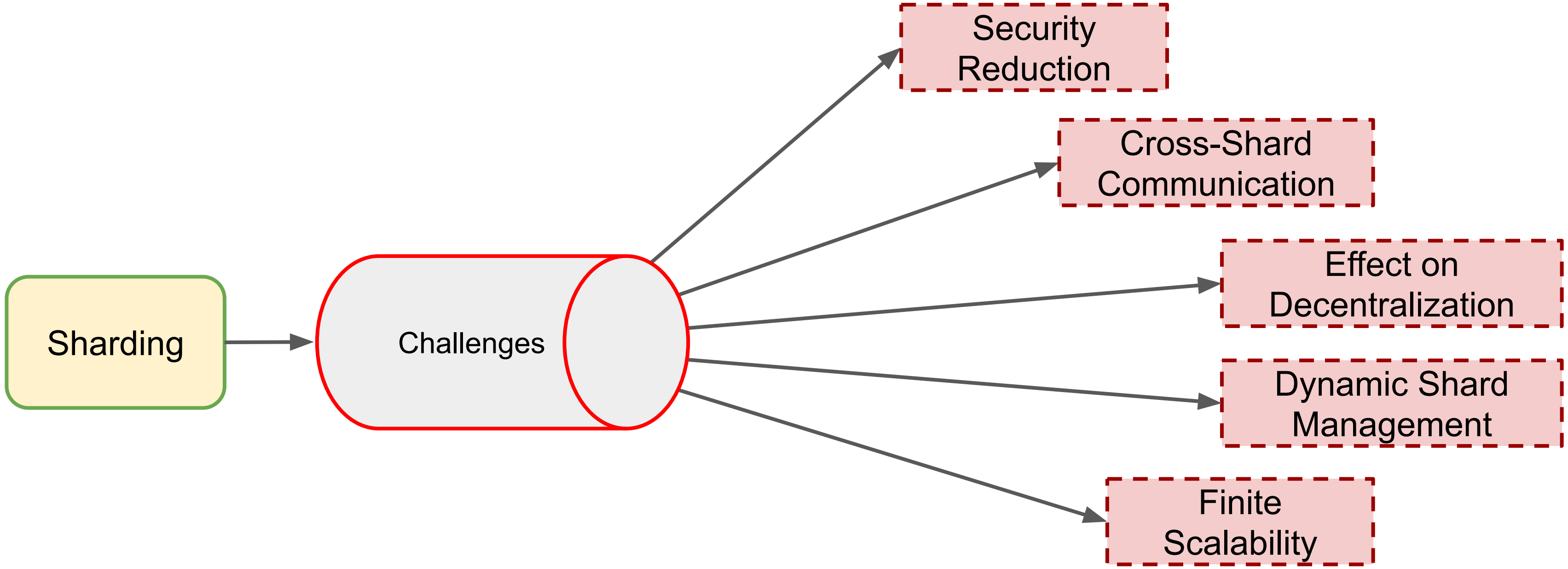
Challenges in Sharding



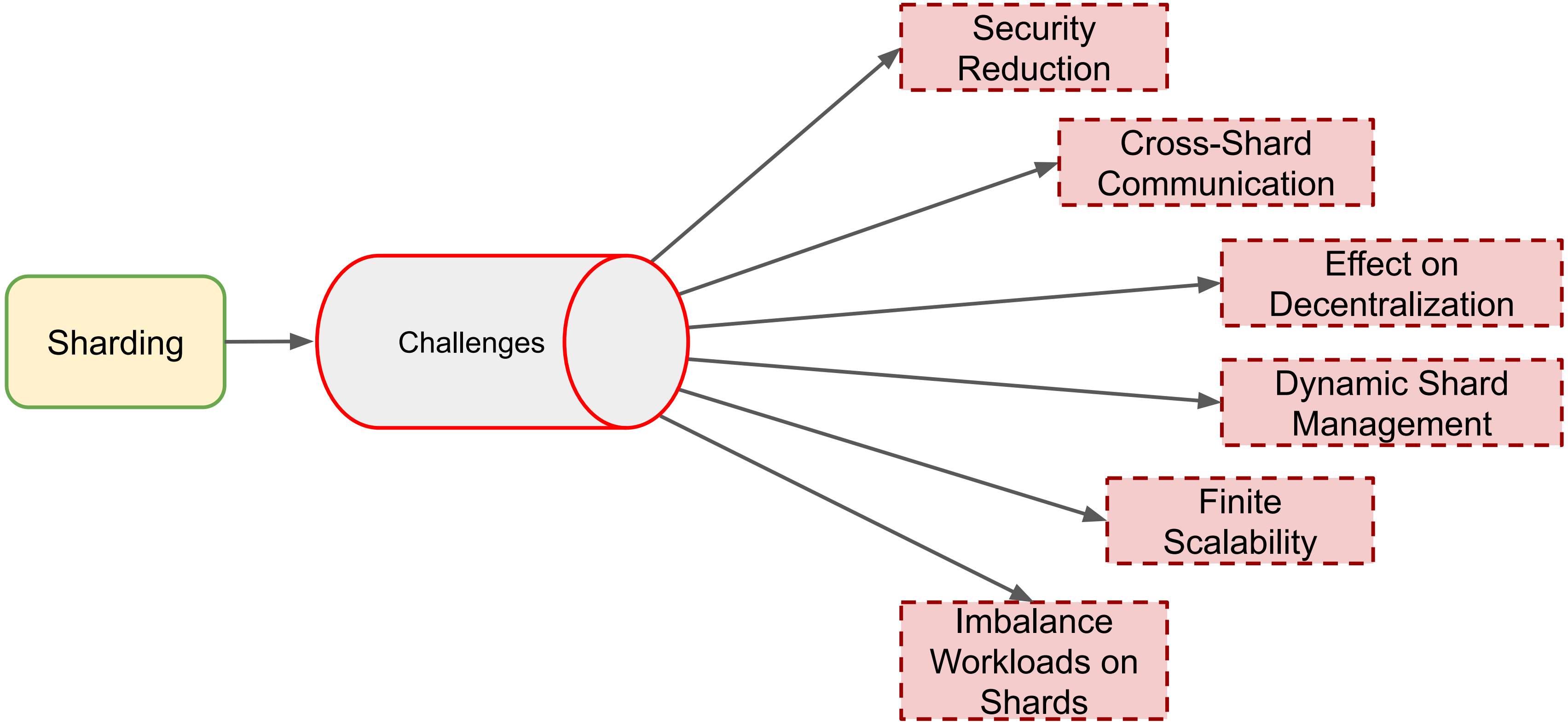
Challenges in Sharding



Challenges in Sharding



Challenges in Sharding


Sharding
Challenges
Security Reduction
Cross-Shard Communication
Effect on Decentralization
Dynamic Shard Management
Finite Scalability
Imbalance Workloads on Shards

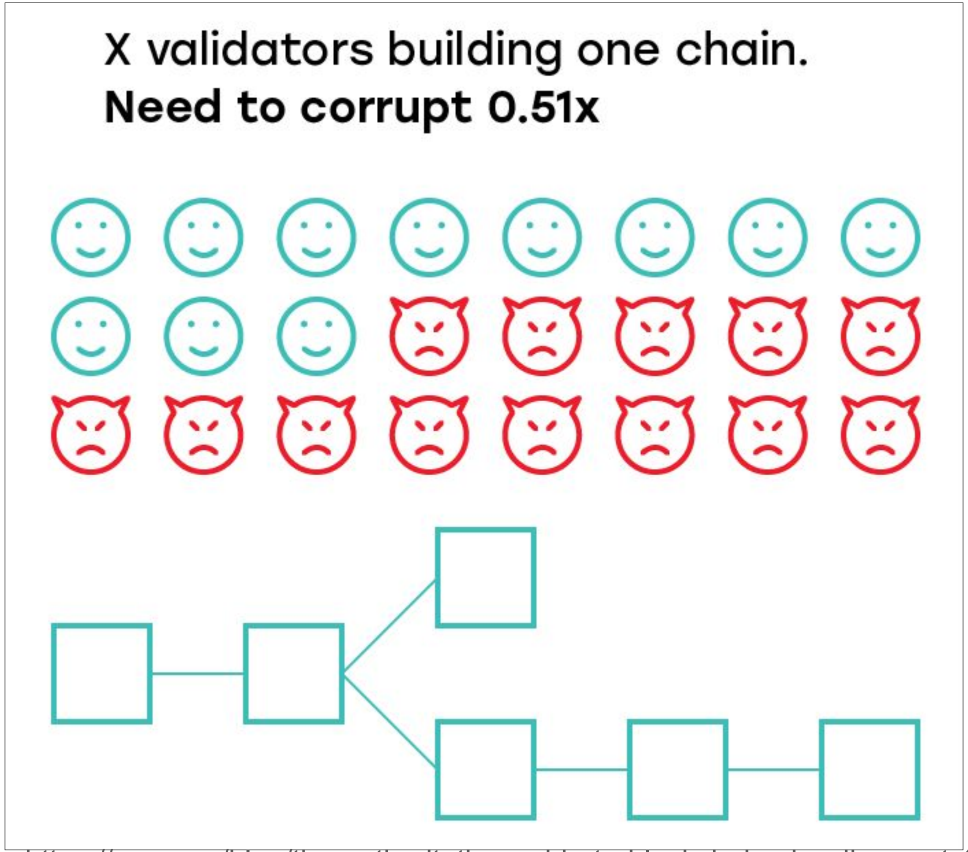
Security Reduction

In a protocol where majority votes are considered as final decision, over 50% of corrupted nodes will always yield a corrupted block.

Security Reduction

In a protocol where majority votes are considered as final decision, over 50% of corrupted nodes will always yield a corrupted block.

But, getting over 50% of the entire network is HARD!



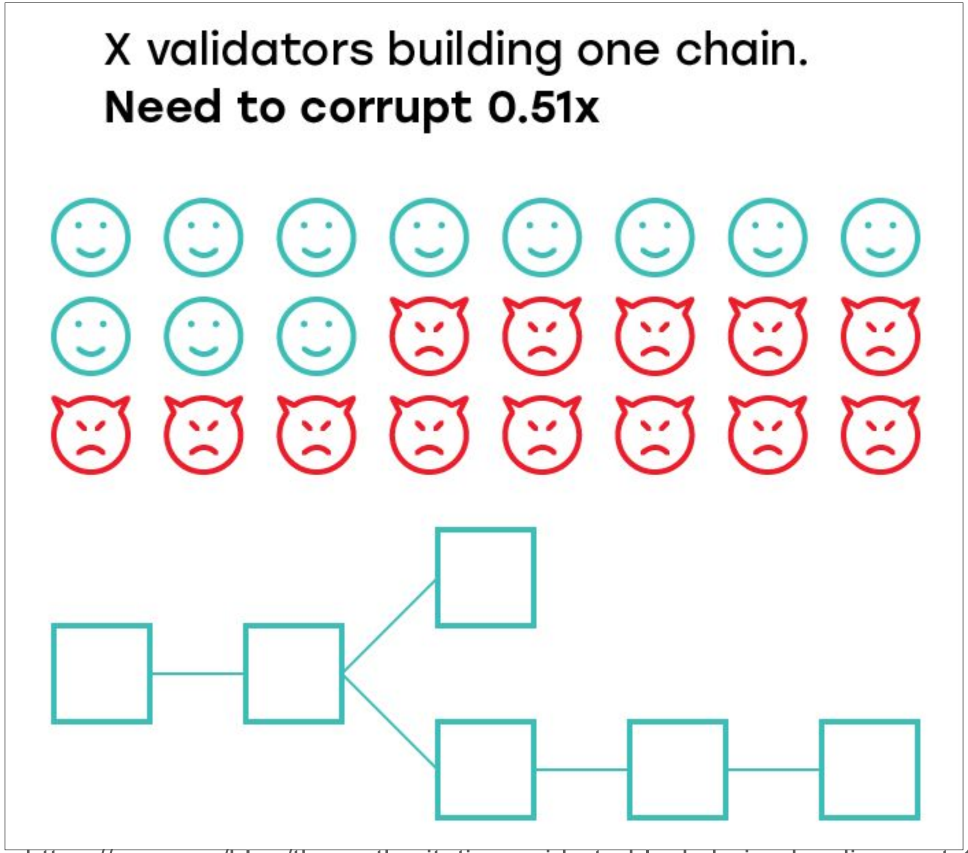
https://near.org/blog/the-authoritative-guide-to-blockchain-sharding-part-1

Security Reduction

In a protocol where majority votes are considered as final decision, over 50% of corrupted nodes will always yield a corrupted block.

But, getting over 50% of the entire network is HARD!

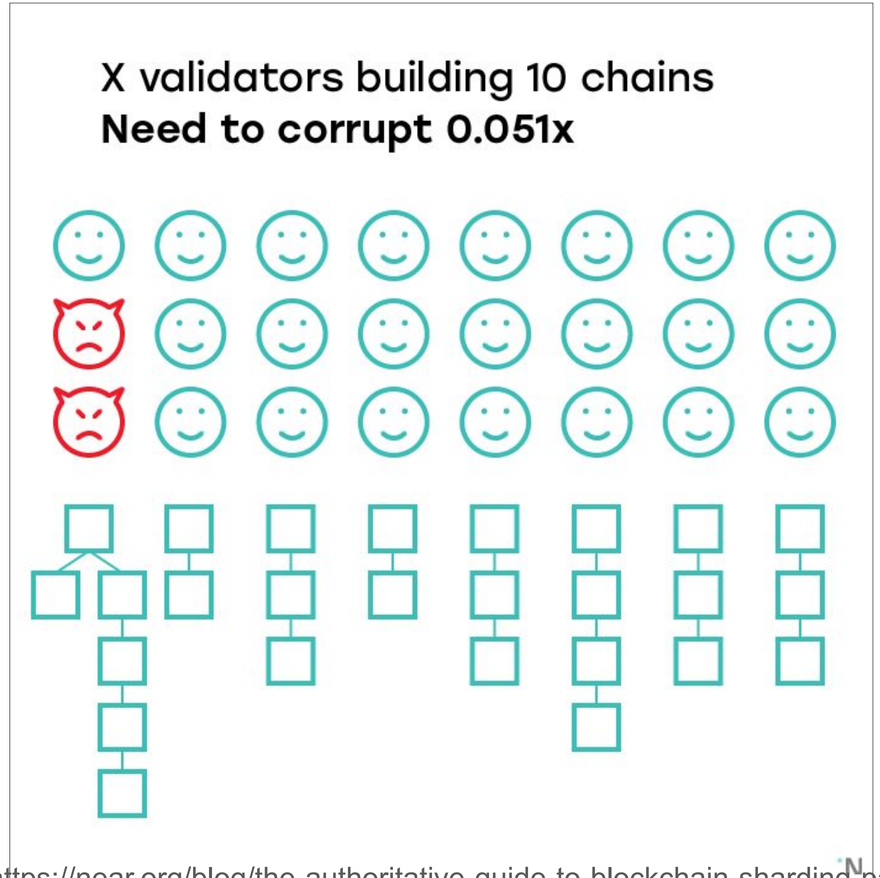
Famously known as 51% attack.



https://near.org/blog/the-authoritative-guide-to-blockchain-sharding-part-1

Security Reduction

But, getting over 50% of a single shard is EASY!

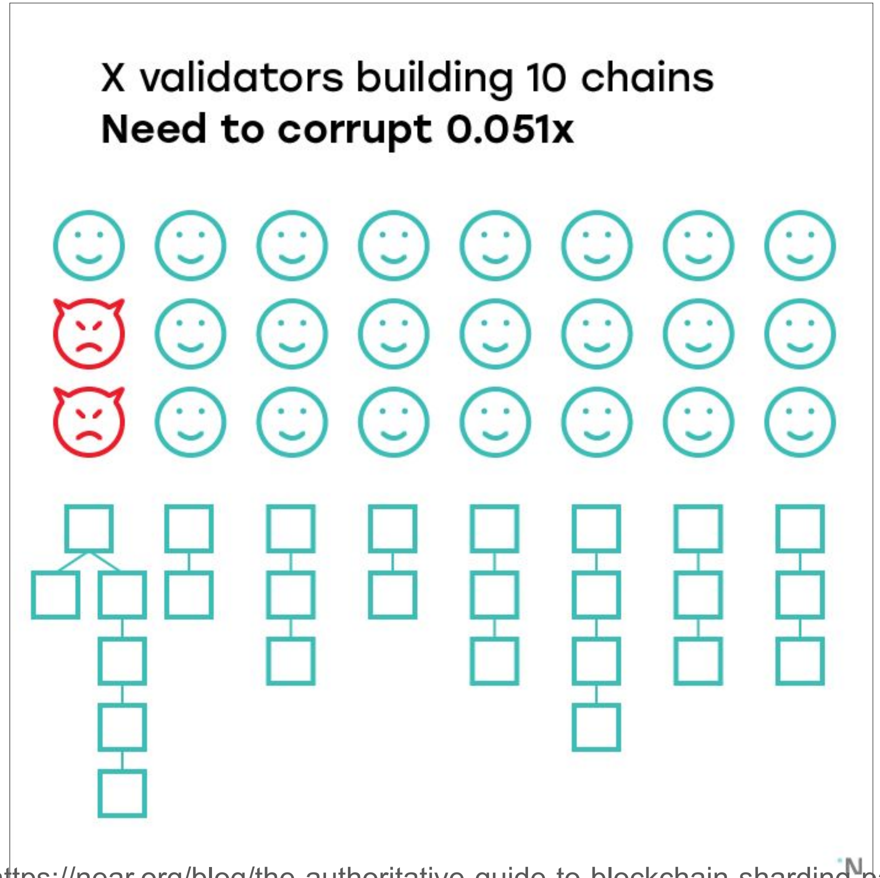


https://near.org/blog/the-authoritative-guide-to-blockchain-sharding-part-1

Security Reduction

But, getting over 50% of a single shard is EASY!

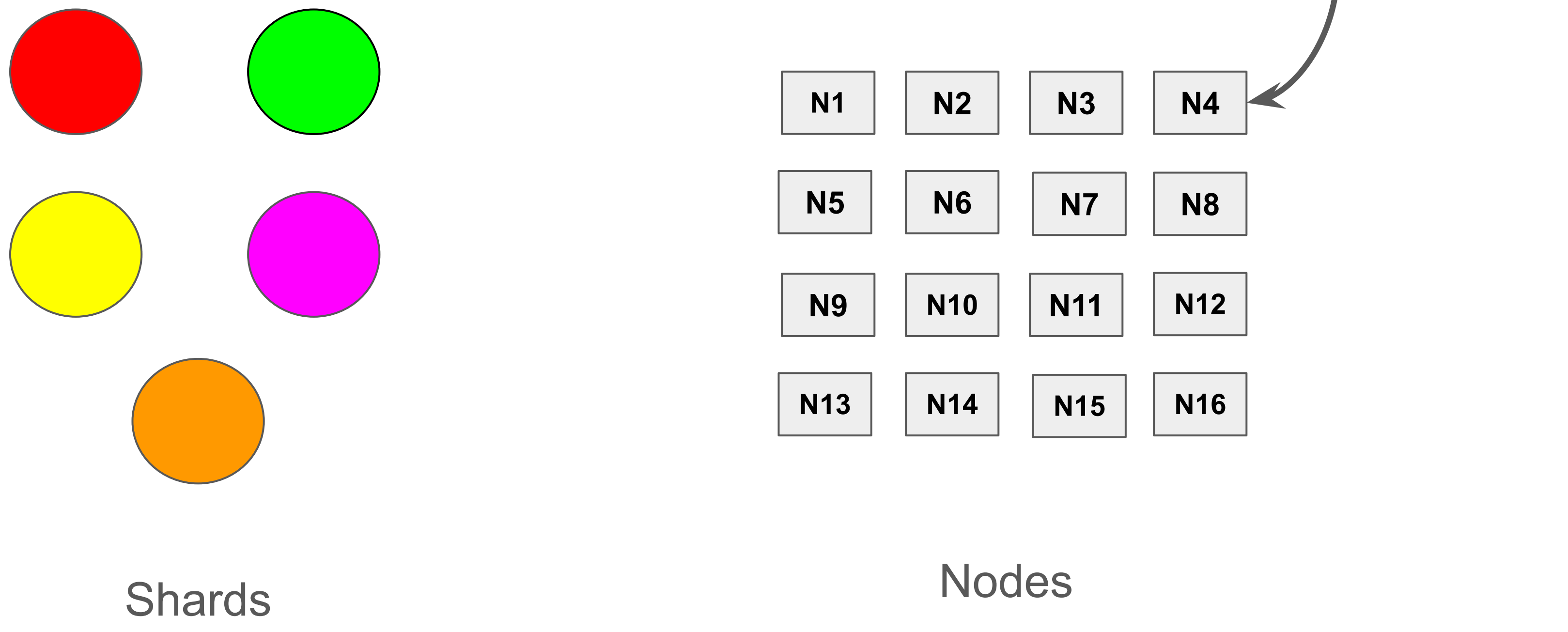
Famously known as 1% attack.



https://near.org/blog/the-authoritative-guide-to-blockchain-sharding-part-1

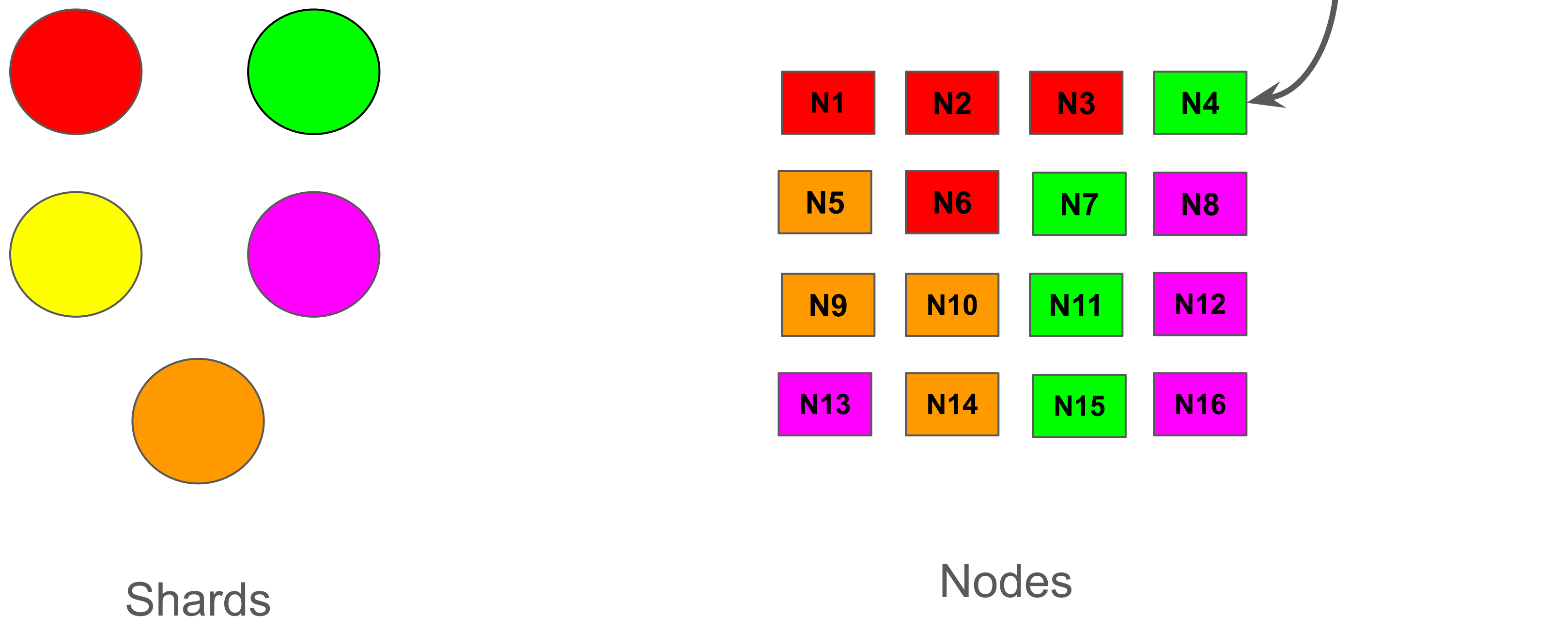
Techniques of Sharding

1) Network Sharding



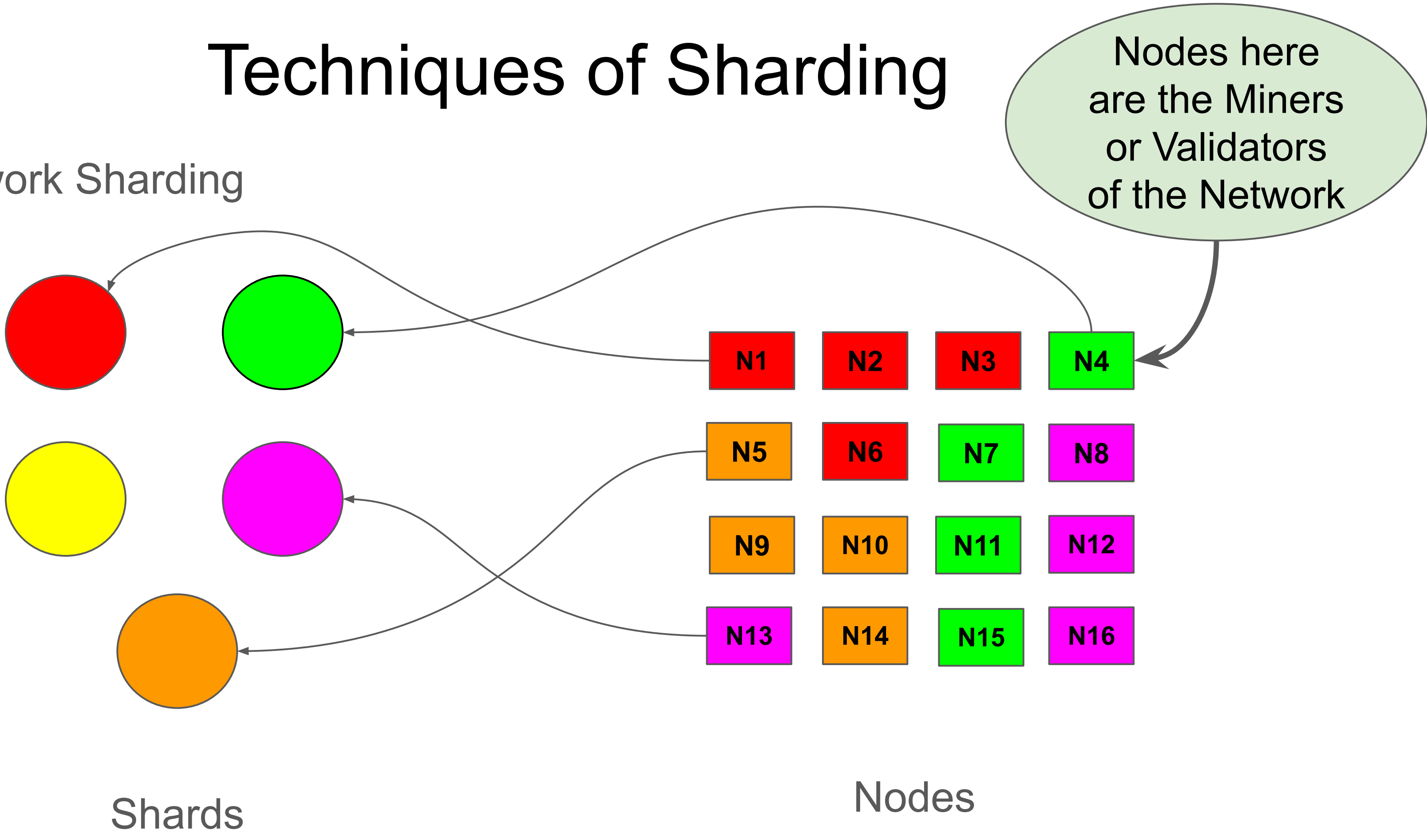
Techniques of Sharding

1) Network Sharding



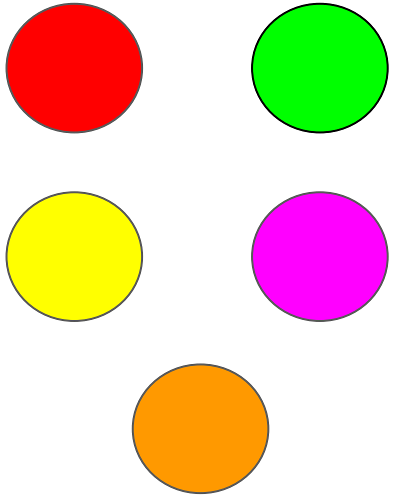
Techniques of Sharding

1) Network Sharding

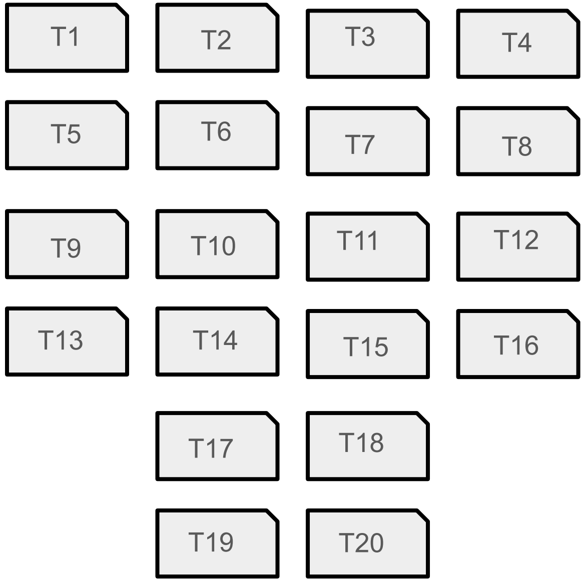


Techniques of Sharding

2) Transaction Sharding



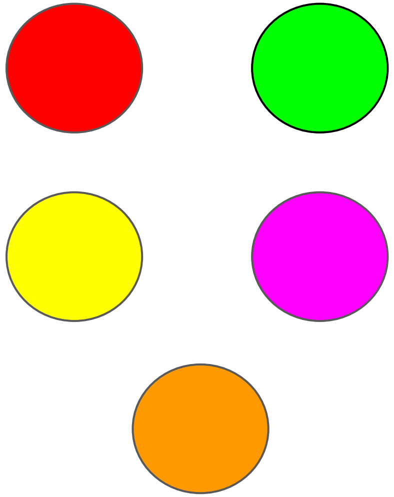
Shards

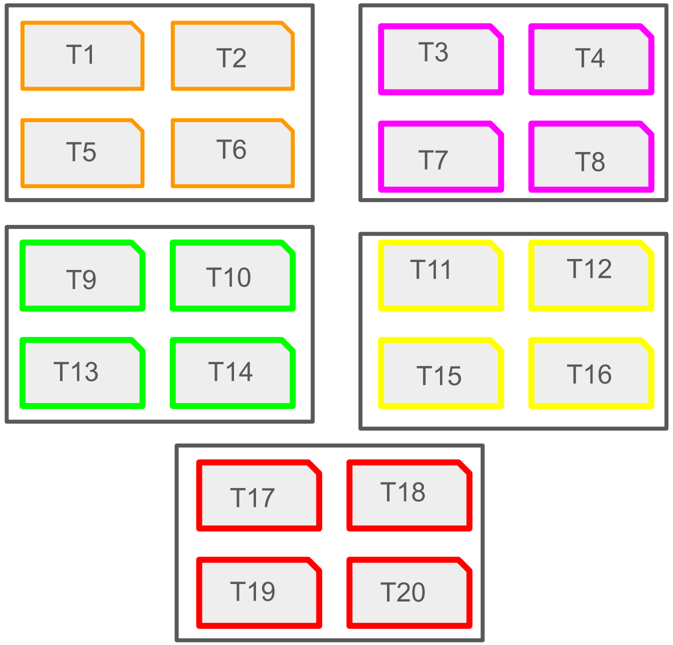


Transactions

Techniques of Sharding

2) Transaction Sharding

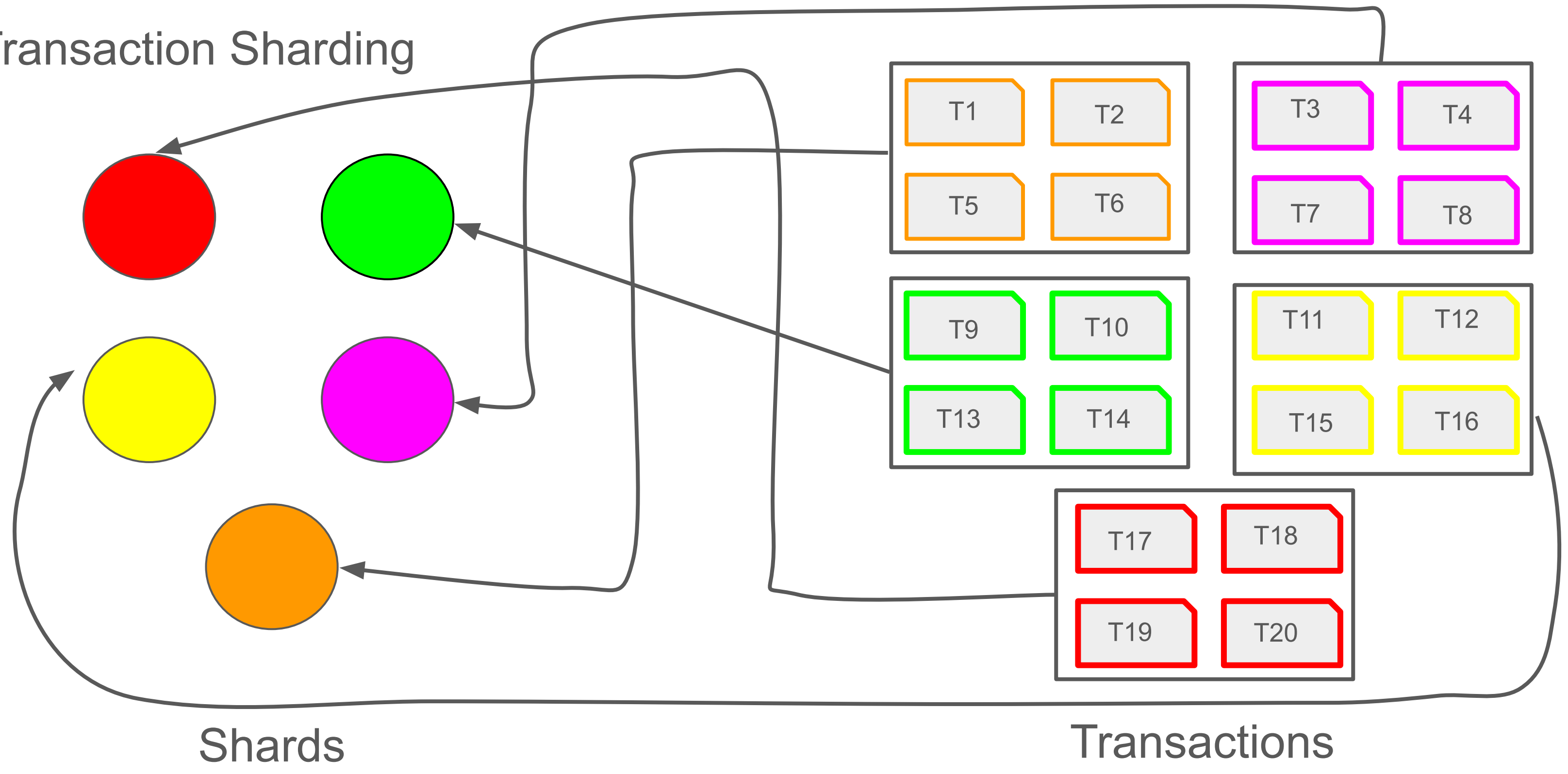

Shards



Transactions

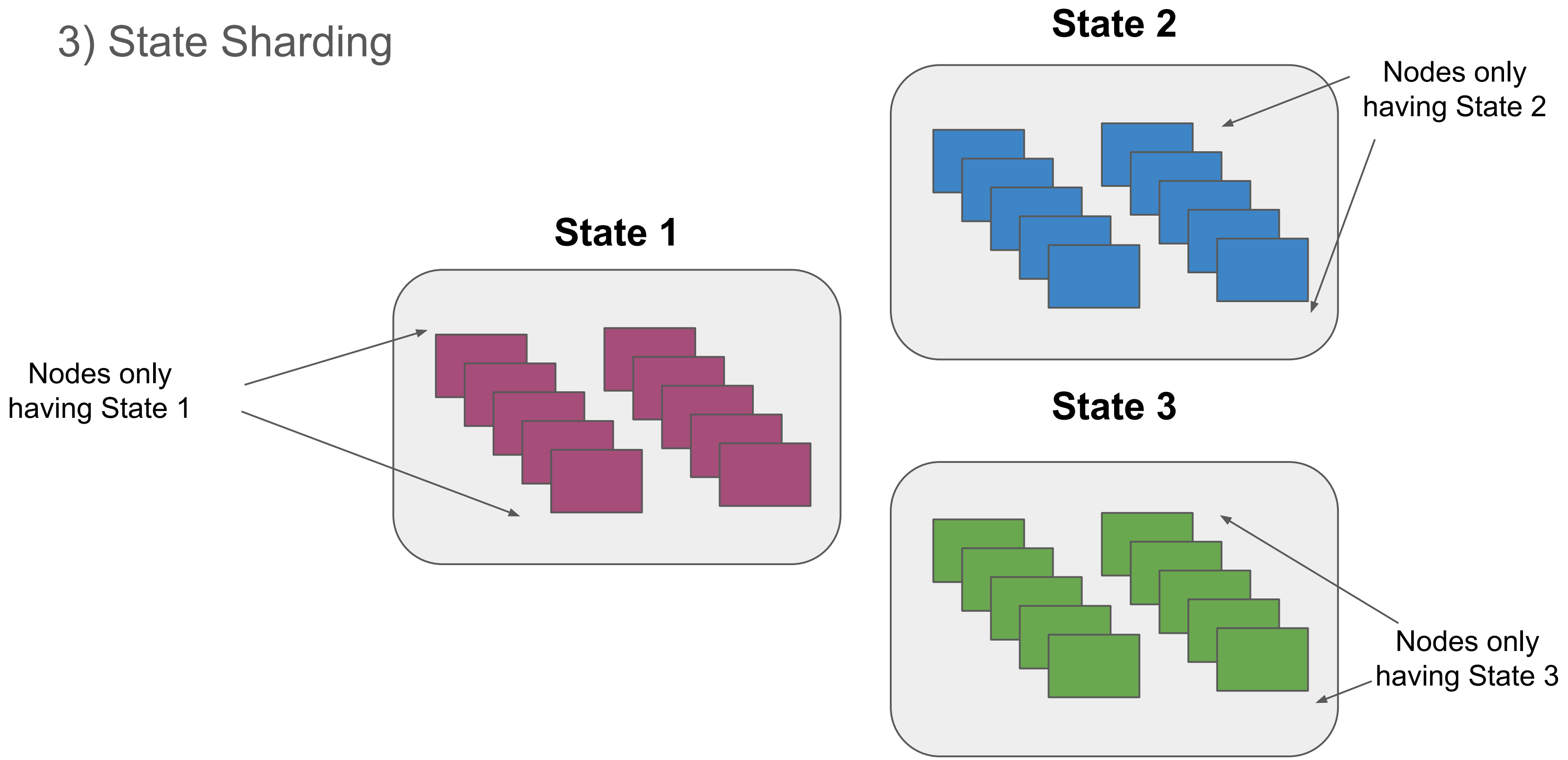
Techniques of Sharding

2) Transaction Sharding



Techniques of Sharding

3) State Sharding



State 2
Nodes only having State 2
State 1
Nodes only having State 1
State 3
Nodes only having State 3

Metrics for evaluating a Sharding Solution

Scalability

Security

Fault Tolerance

Latency

Load Balancing

Throughput

Adaptability and Flexibility

Atomicity and Overhead of Cross-Shard Transactions

Metrics for evaluating a Sharding Solution

- Scalability
- Security
- Fault Tolerance
- Latency
- Load Balancing
- Throughput
- Adaptability and Flexibility
- Atomicity and Overhead of Cross-Shard Transactions

Steps in Sharding

1) Identity Establishment

Steps in Sharding

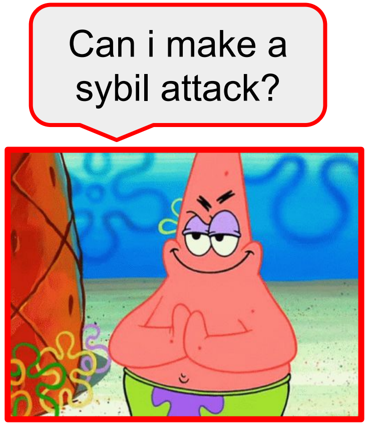
1) Identity Establishment

Nodes need to establish their identity using some method like PoX.

Steps in Sharding

1) Identity Establishment

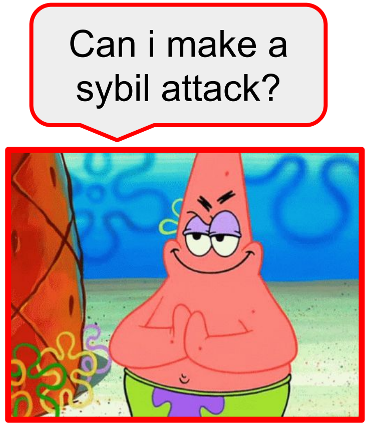
Nodes need to establish their identity using some method like PoX.

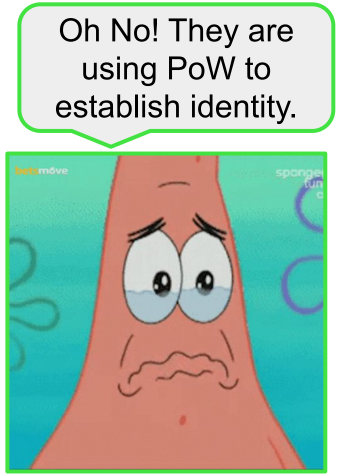


Steps in Sharding

1) Identity Establishment

Nodes need to establish their identity using some method like PoX.



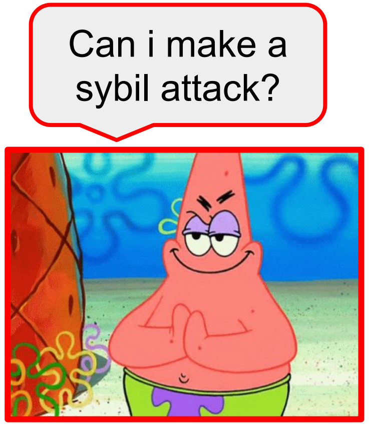


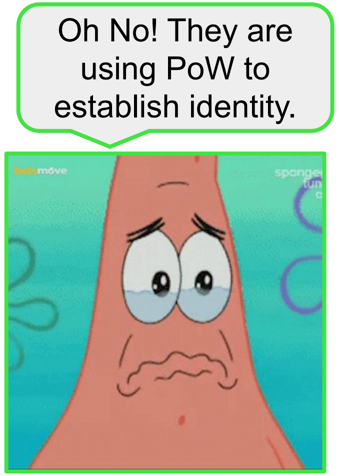
Steps in Sharding

1) Identity Establishment

2) Shard Assignment and Setup

Nodes need to establish their identity using some method like PoX.

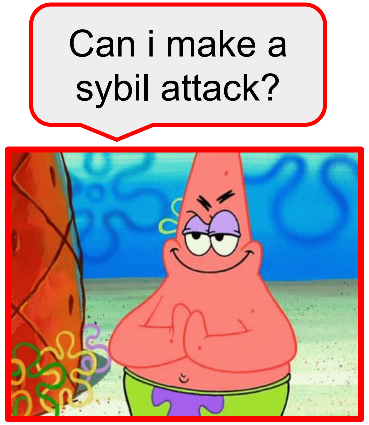


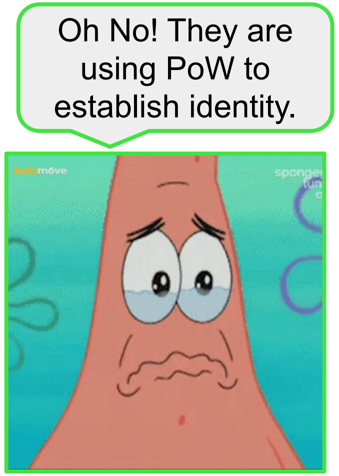


Steps in Sharding

1) Identity Establishment

Nodes need to establish their identity using some method like PoX.





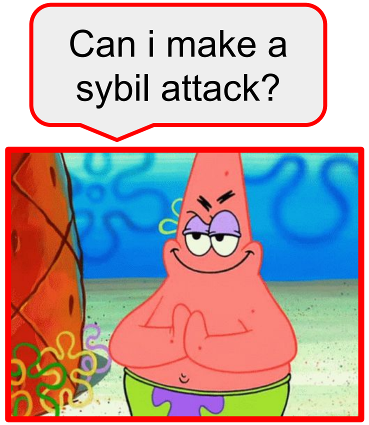
2) Shard Assignment and Setup

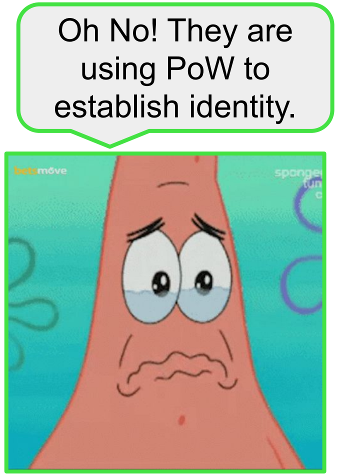
Nodes and transaction randomly distributed among shard based on their identity.

Steps in Sharding

1) Identity Establishment

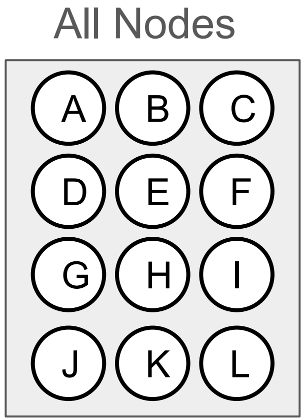
Nodes need to establish their identity using some method like PoX.





2) Shard Assignment and Setup

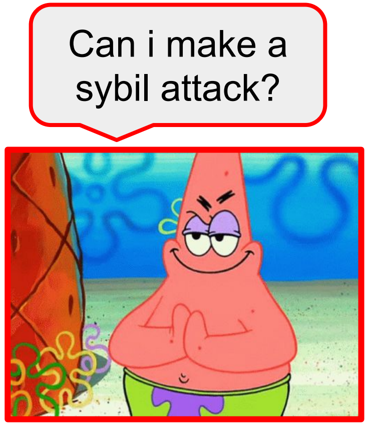
Nodes and transaction randomly distributed among shard based on their identity.

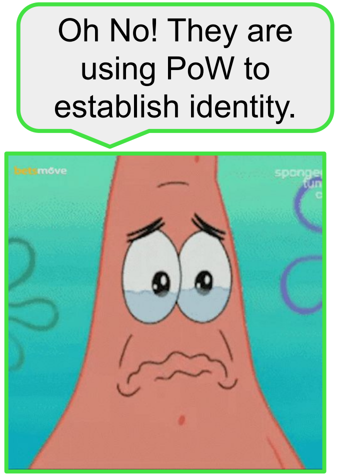


Steps in Sharding

1) Identity Establishment

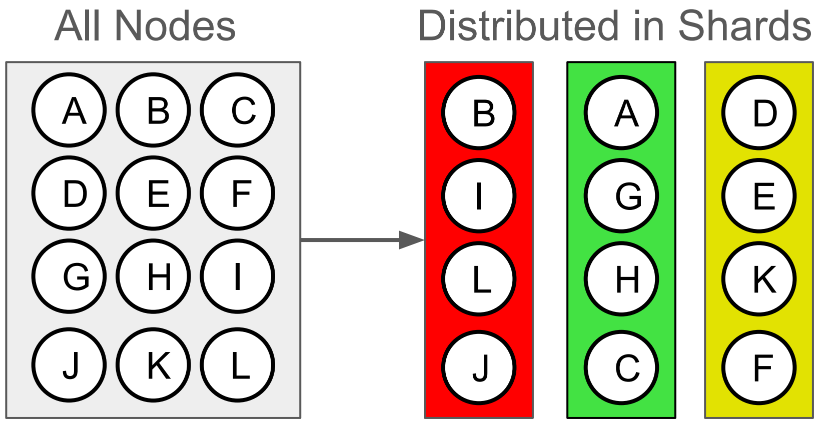
Nodes need to establish their identity using some method like PoX.





2) Shard Assignment and Setup

Nodes and transaction randomly distributed (non-deterministically) among shard based on their identity.



Steps in Sharding

3) Consensus Determination

Steps in Sharding

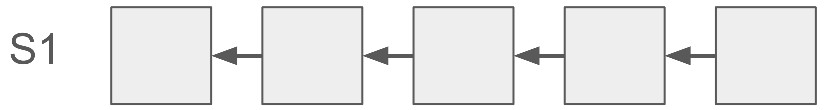
3) Consensus Determination

Consensus needs to be reached for both intra-shard and cross-shard communication.

Steps in Sharding

3) Consensus Determination

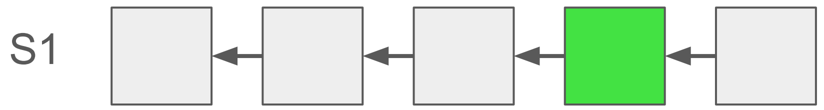
Consensus needs to be reached for both intra-shard and cross-shard communication.



Steps in Sharding

3) Consensus Determination

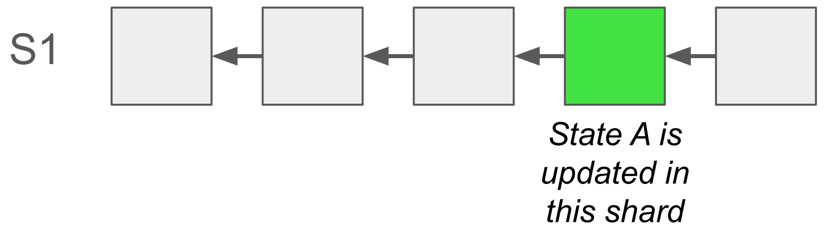
Consensus needs to be reached for both intra-shard and cross-shard communication.



Steps in Sharding

3) Consensus Determination

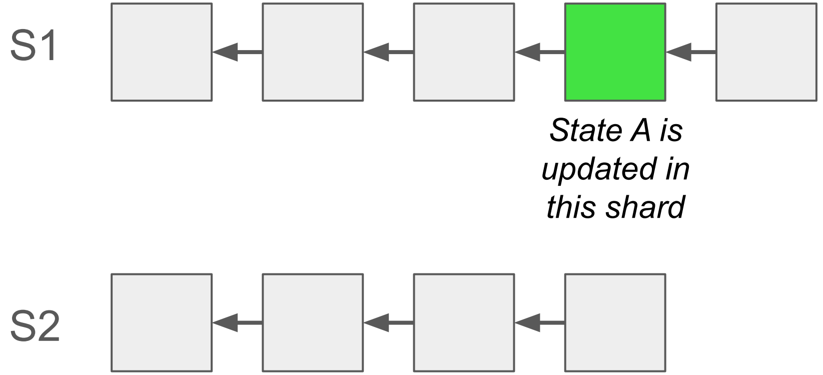
Consensus needs to be reached for both intra-shard and cross-shard communication.



Steps in Sharding

3) Consensus Determination

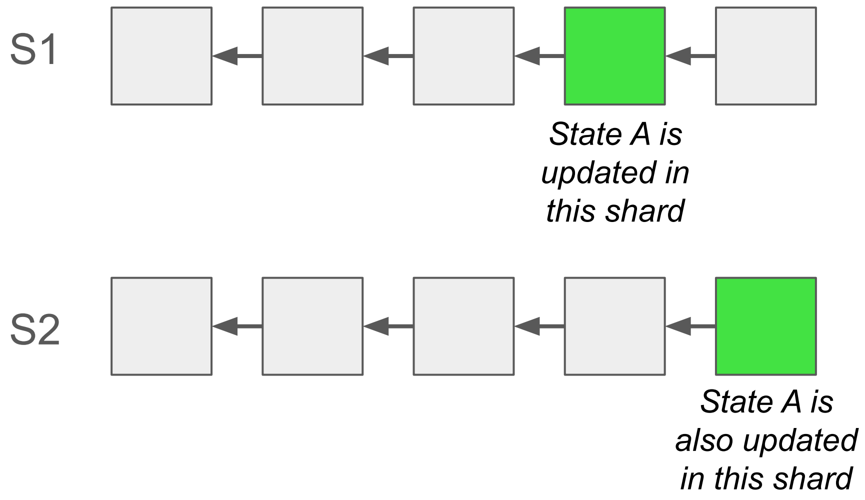
Consensus needs to be reached for both intra-shard and cross-shard communication.



Steps in Sharding

3) Consensus Determination

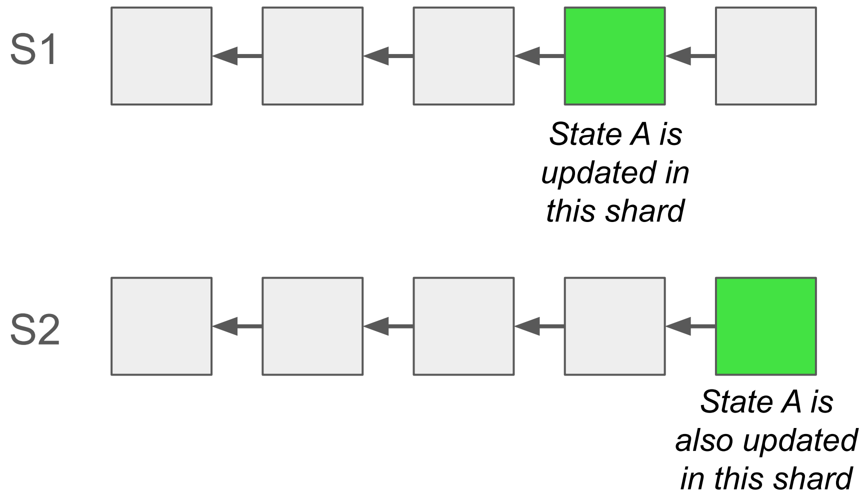
Consensus needs to be reached for both intra-shard and cross-shard communication.


S1
State A is updated in this shard
S2
State A is also updated in this shard

Steps in Sharding

3) Consensus Determination

Consensus needs to be reached for both intra-shard and cross-shard communication.

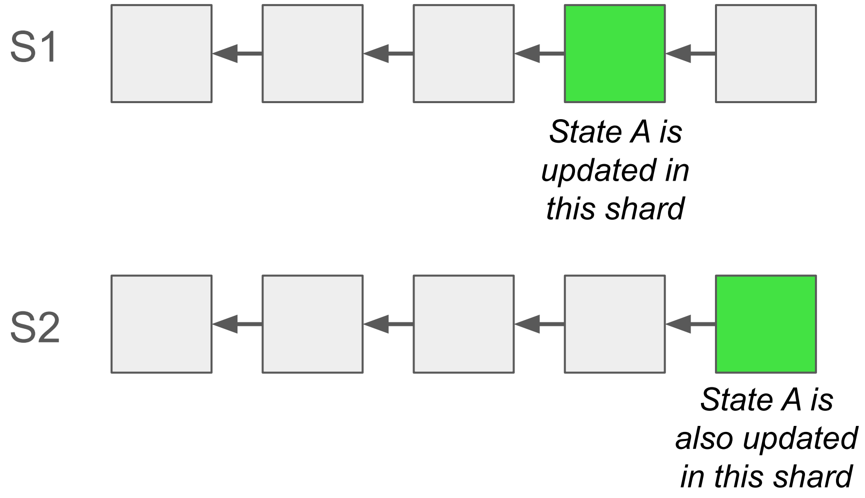

S1
State A is updated in this shard
S2
State A is also updated in this shard

4) Shard Reconfiguration

Steps in Sharding

3) Consensus Determination

Consensus needs to be reached for both intra-shard and cross-shard communication.


S1
State A is updated in this shard
S2
State A is also updated in this shard

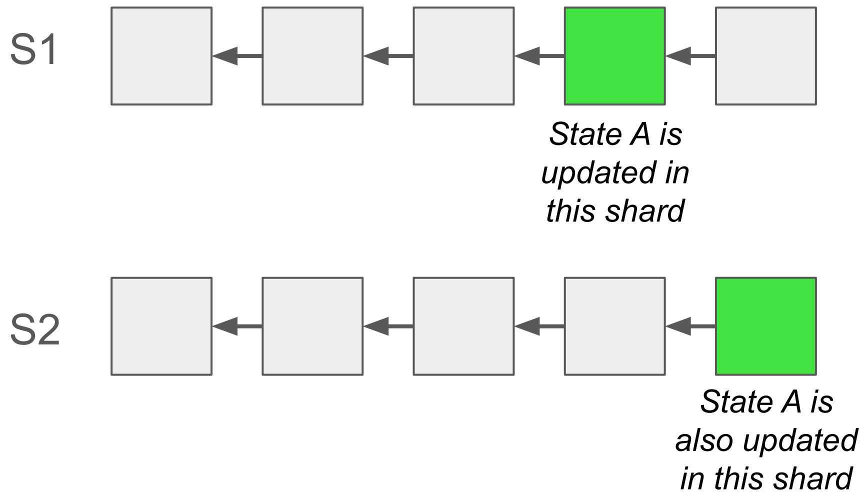
4) Shard Reconfiguration

Reconfiguration with redistribution of nodes among shard after each epoch to bolster security.

Steps in Sharding

3) Consensus Determination

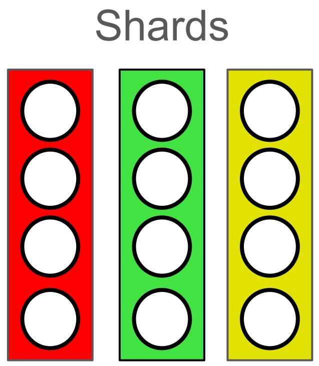
Consensus needs to be reached for both intra-shard and cross-shard communication.



S1
State A is updated in this shard
S2
State A is also updated in this shard

4) Shard Reconfiguration

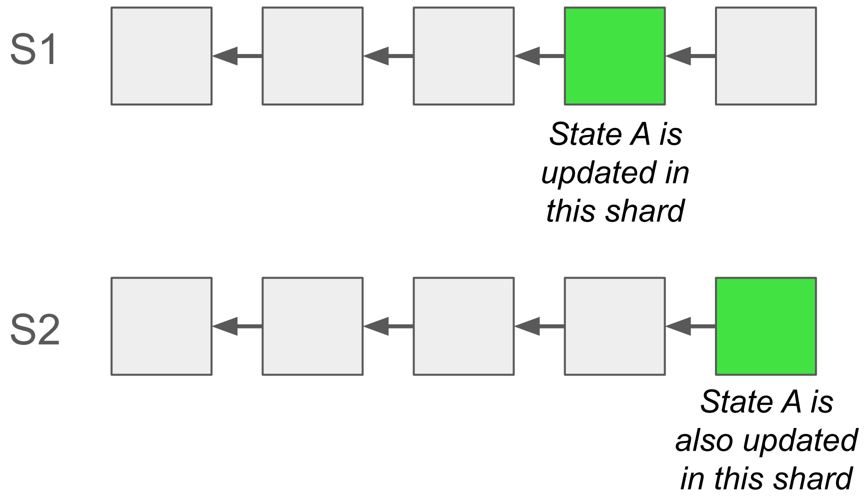
Reconfiguration with redistribution of nodes among shard after each epoch to bolster security.



Steps in Sharding

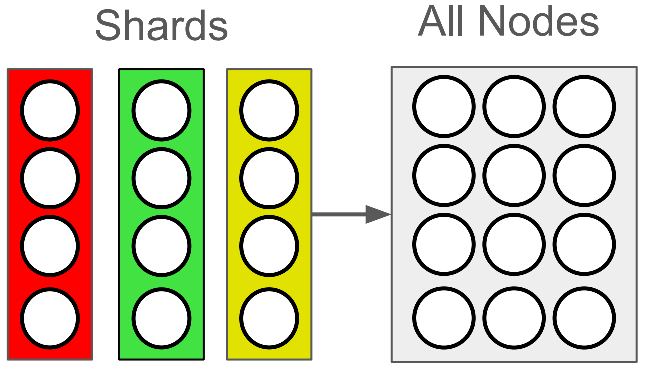
3) Consensus Determination

Consensus needs to be reached for both intra-shard and cross-shard communication.


S1
State A is updated in this shard
S2
State A is also updated in this shard

4) Shard Reconfiguration

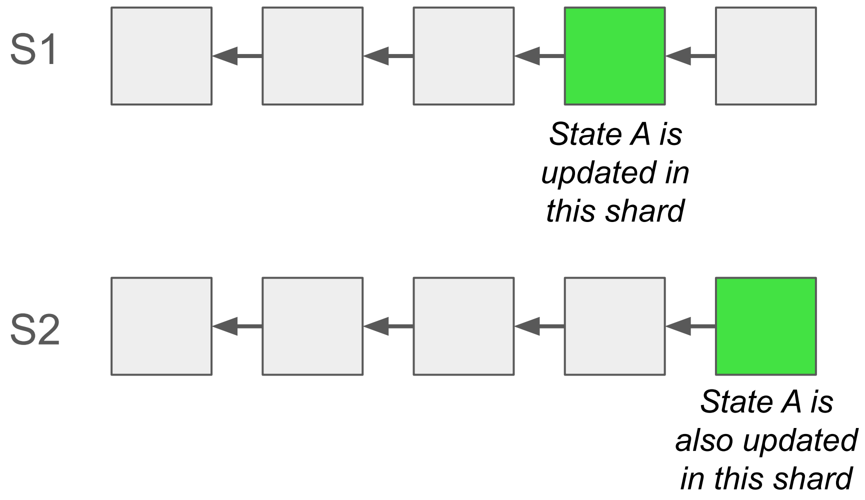
Reconfiguration with redistribution of nodes among shard after each epoch to bolster security.



Steps in Sharding

3) Consensus Determination

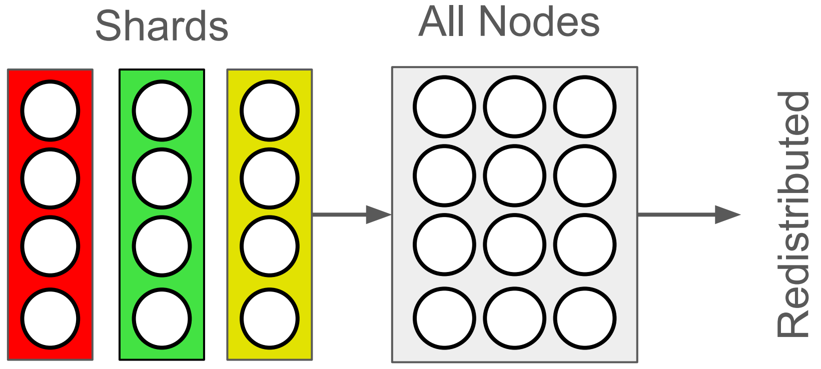
Consensus needs to be reached for both intra-shard and cross-shard communication.



S1
State A is updated in this shard
S2
State A is also updated in this shard

4) Shard Reconfiguration

Reconfiguration with redistribution of nodes among shard after each epoch to bolster security.



Elastico [CCS'16]

First secure sharding protocol for open blockchain.

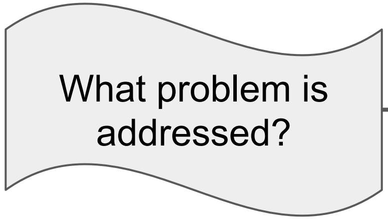
Elastico [CCS'16]

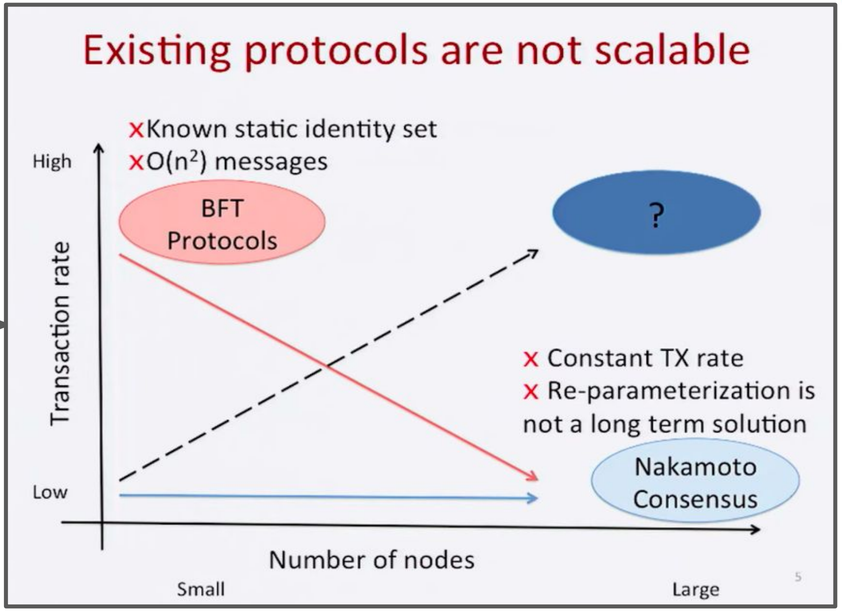
First secure sharding protocol for open blockchain.

What problem is addressed?

Elastico [CCS'16]

First secure sharding protocol for open blockchain.



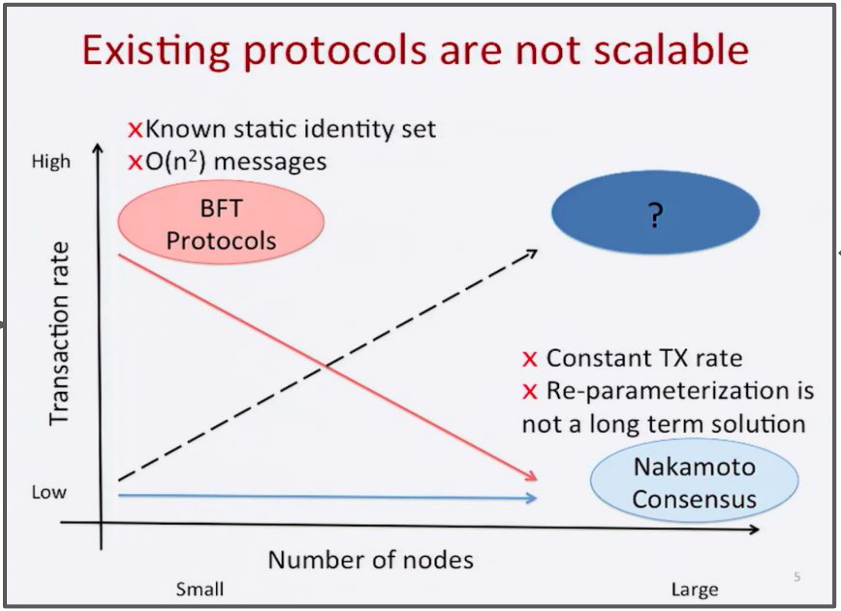


Luu, Loi, et al. "A secure sharding protocol for open blockchains."

Elastico [CCS'16]

First secure sharding protocol for open blockchain.

What problem is addressed?



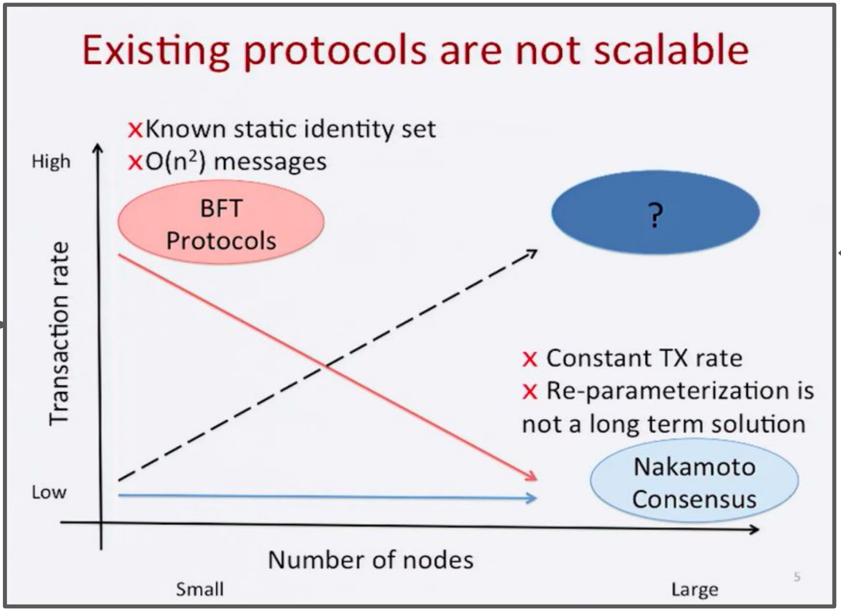
Luu, Loi, et al. "A secure sharding protocol for open blockchains."

Can we make "n" smaller, so BFT protocols can run faster?

Elastico [CCS'16]

First secure sharding protocol for open blockchain.

What problem is addressed?



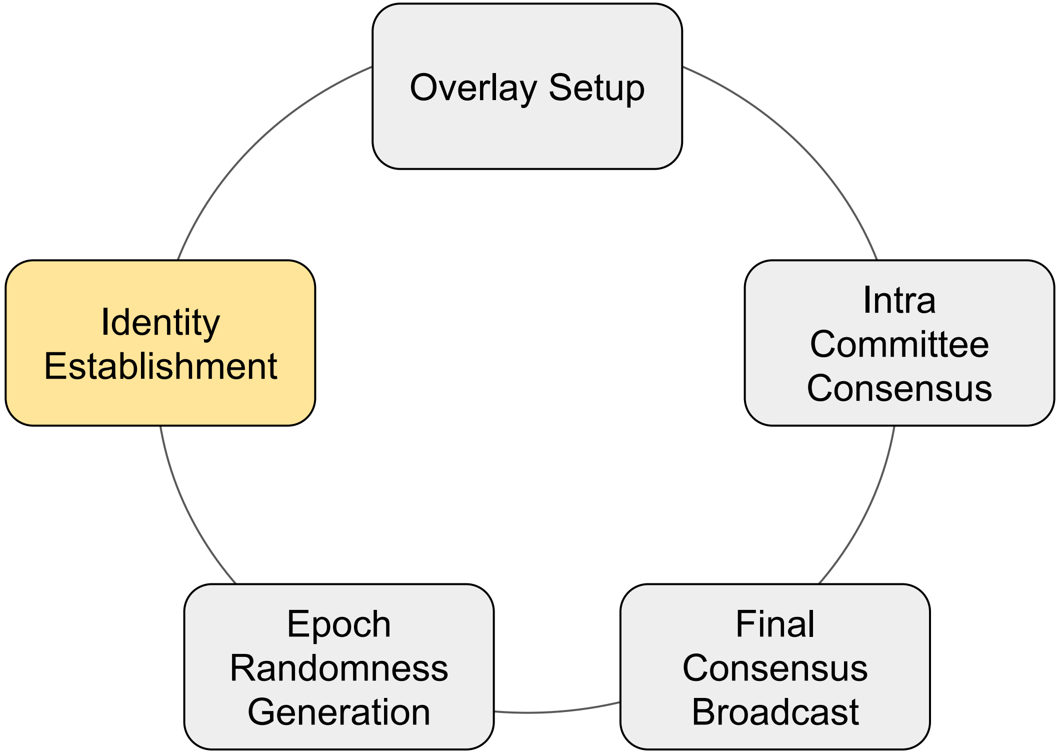
Luu, Loi, et al. "A secure sharding protocol for open blockchains."

Can we make "n" smaller, so BFT protocols can run faster?

Sharding

Elastico

The protocol operates in *Epochs*.


Overlay Setup
Intra Committee Consensus
Final Consensus Broadcast
Epoch Randomness Generation
Identity Establishment

Elastico

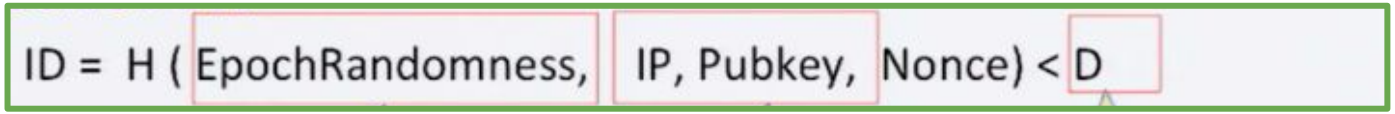
Identity Establishment

Nodes perform PoW to establish their identity.

Elastico

Identity Establishment

Nodes perform PoW to establish their identity.

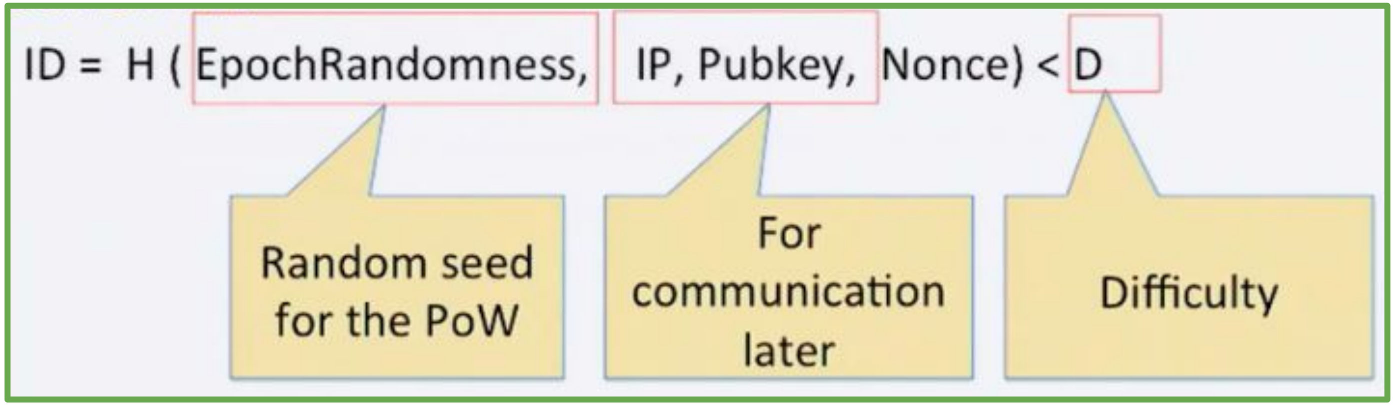


$ID = H(\boxed{EpochRandomness,}\ \boxed{IP, Pubkey,}\ Nonce) < \boxed{D}$

Elastico

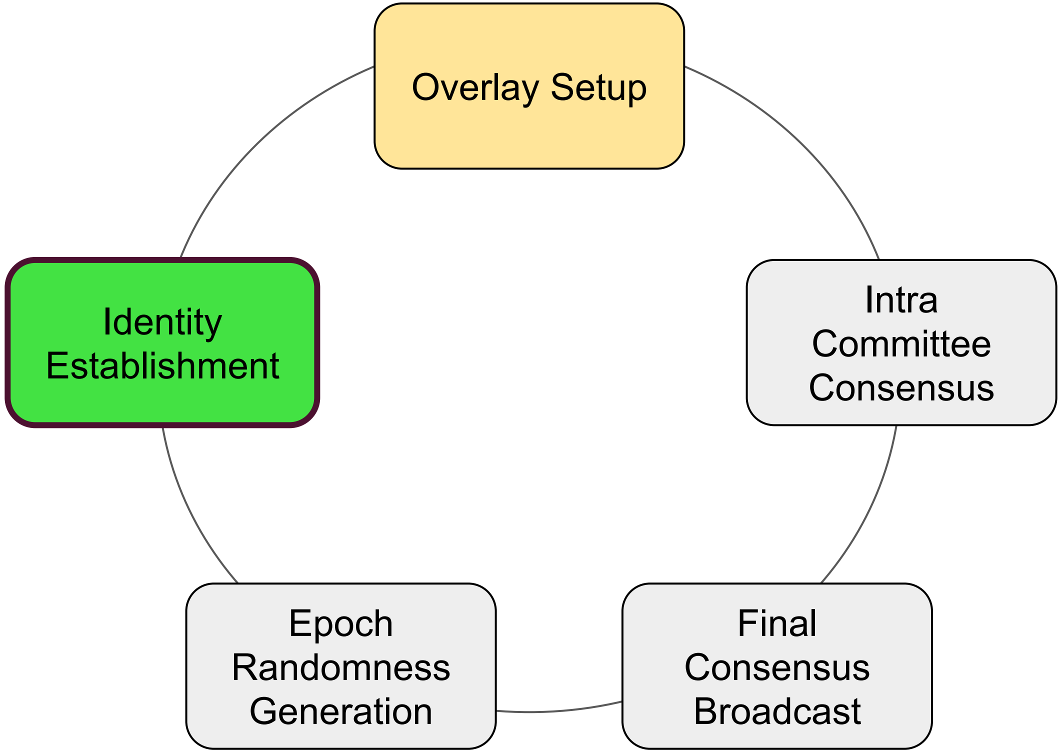
Identity Establishment

Nodes perform PoW to establish their identity.



Luu, Loi, et al. "A secure sharding protocol for open blockchains."

Elastico


Overlay Setup
Intra Committee Consensus
Final Consensus Broadcast
Epoch Randomness Generation
Identity Establishment

Elastico

Overlay Setup

- Nodes are assigned to shard according to their ID.

Elastico

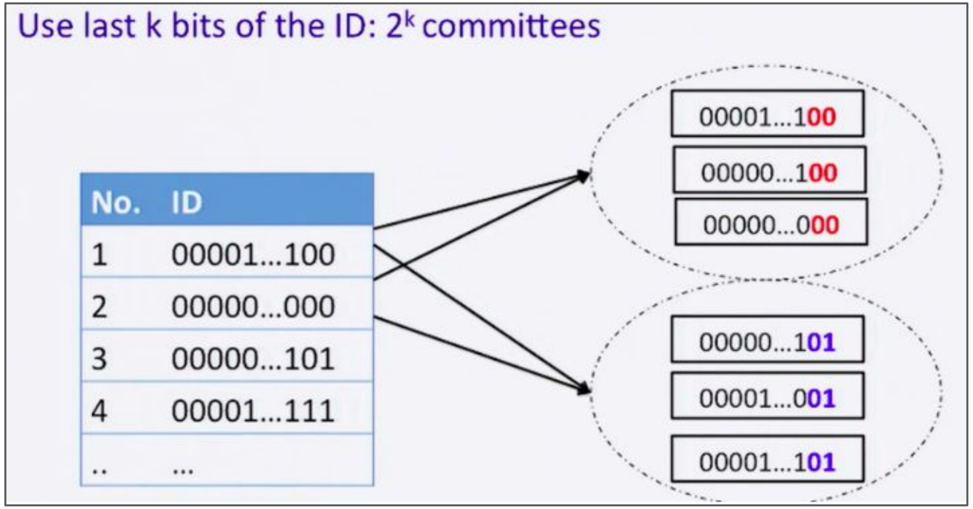
Overlay Setup

- Nodes are assigned to shard according to their ID.
- Elastico equitably allocates identities to 2^k committees, utilizing the last k bits of their identities (ID).

Elastico

Overlay Setup

- Nodes are assigned to shard according to their ID.
- Elastico equitably allocates identities to 2^k committees, utilizing the last k bits of their identities (ID).



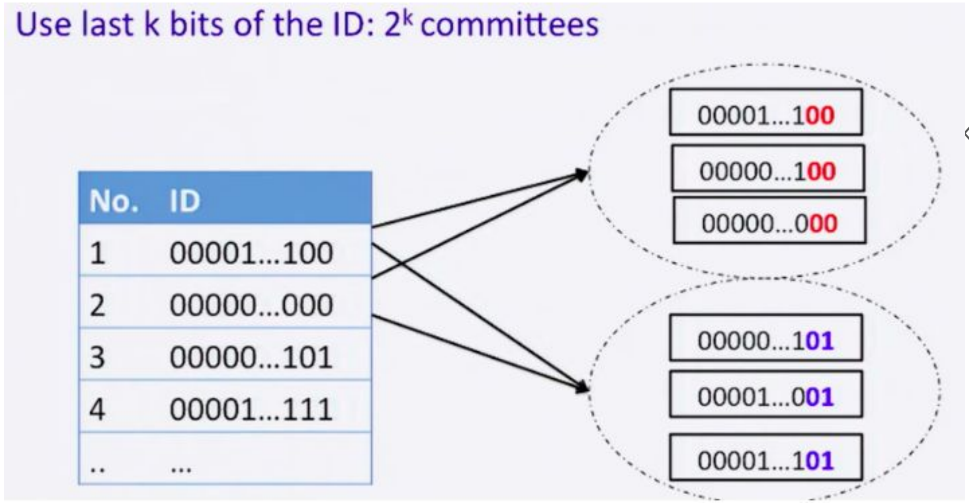
No.	ID
1	00001...100
2	00000...000
3	00000...101
4	00001...111
..	...

Luu, Loi, et al. "A secure sharding protocol for open blockchains."

Elastico

Overlay Setup

- Nodes are assigned to shard according to their ID.
- Elastico equitably allocates identities to 2^k committees, utilizing the last k bits of their identities (ID).



No.	ID
1	00001...100
2	00000...000
3	00000...101
4	00001...111
..	...

Luu, Loi, et al. "A secure sharding protocol for open blockchains."

This is done to limit at most ⅓ malicious for each shard (for PBFT)

How to get identities from other nodes?

Elastico

Overlay Setup

Elastico

Overlay Setup

How to get identities from other nodes?

Broadcast N identity to N nodes.
Resulting in $O(N^2)$ messages.

Naive Method

Elastico

Overlay Setup

How to get identities from other nodes?

Broadcast N identity to N nodes.
Resulting in $O(N^2)$ messages.

Naive Method

1. First C identities becomes Directory Servers.

Elastico

Overlay Setup

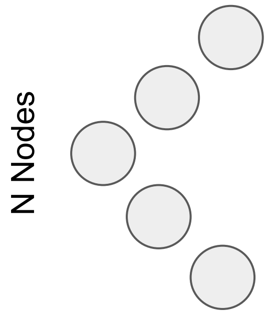
How to get identities from other nodes?

Broadcast N identity to N nodes.
Resulting in $O(N^2)$ messages.

Naive Method

1. First C identities becomes Directory Servers.

2. Each node after establishing identity sends identity to nearest Directory Server.



Elastico

Overlay Setup

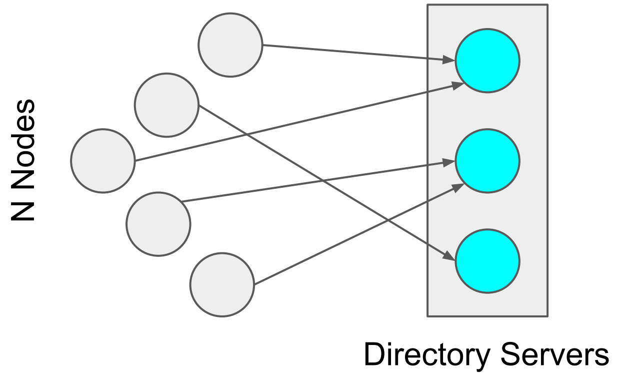
How to get identities from other nodes?

Broadcast N identity to N nodes.
Resulting in $O(N^2)$ messages.

Naive Method

1. First C identities becomes Directory Servers.

2. Each node after establishing identity sends identity to nearest Directory Server.



Elastico

Overlay Setup

How to get identities from other nodes?

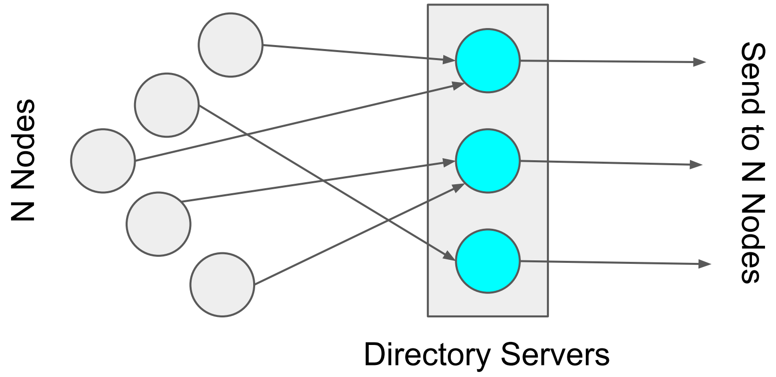
Broadcast N identity to N nodes.
Resulting in $O(N^2)$ messages.

Naive Method

1. First C identities becomes Directory Servers.

2. Each node after establishing identity sends identity to nearest Directory Server.

3. Each Directory Server sends the committee/shard lists to N nodes, once each committee attains C or more nodes.



Elastico

Overlay Setup

How to get identities from other nodes?

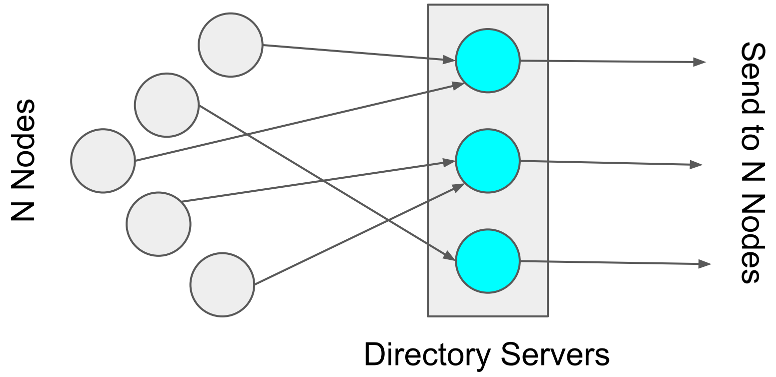
Broadcast N identity to N nodes.
Resulting in $O(N^2)$ messages.

Naive Method

1. First C identities becomes Directory Servers.

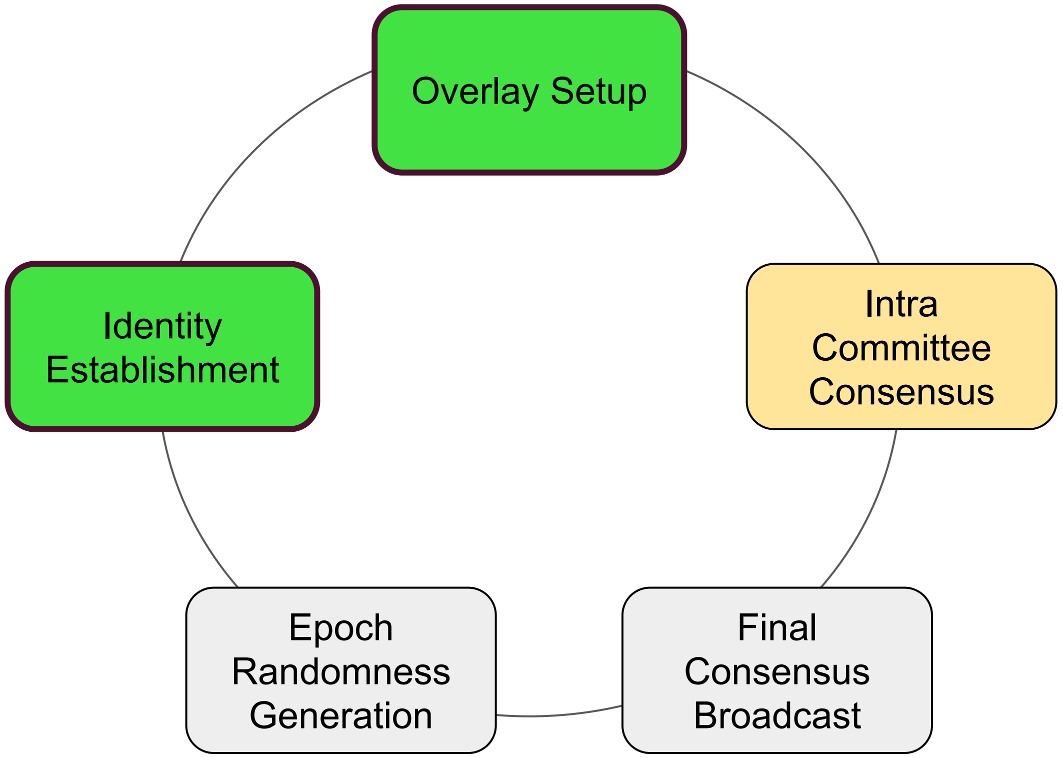
2. Each node after establishing identity sends identity to nearest Directory Server.

3. Each Directory Server sends the committee/shard lists to N nodes, once each committee attains C or more nodes.



Total Message = O(NC)

Elastico


Overlay Setup
Intra Committee Consensus
Final Consensus Broadcast
Epoch Randomness Generation
Identity Establishment

Elastico

Intra-Committee Consensus

- Assumption of ⅓ malicious nodes in each committee.

- Run a Byzantine agreement protocol (PBFT or PolyByz).

- Number of messages in each committee = $O(C^2)$

[C is the Committee Size]

Elastico

Intra-Committee Consensus

Better when C is more

- Assumption of ⅓ malicious nodes in each committee.

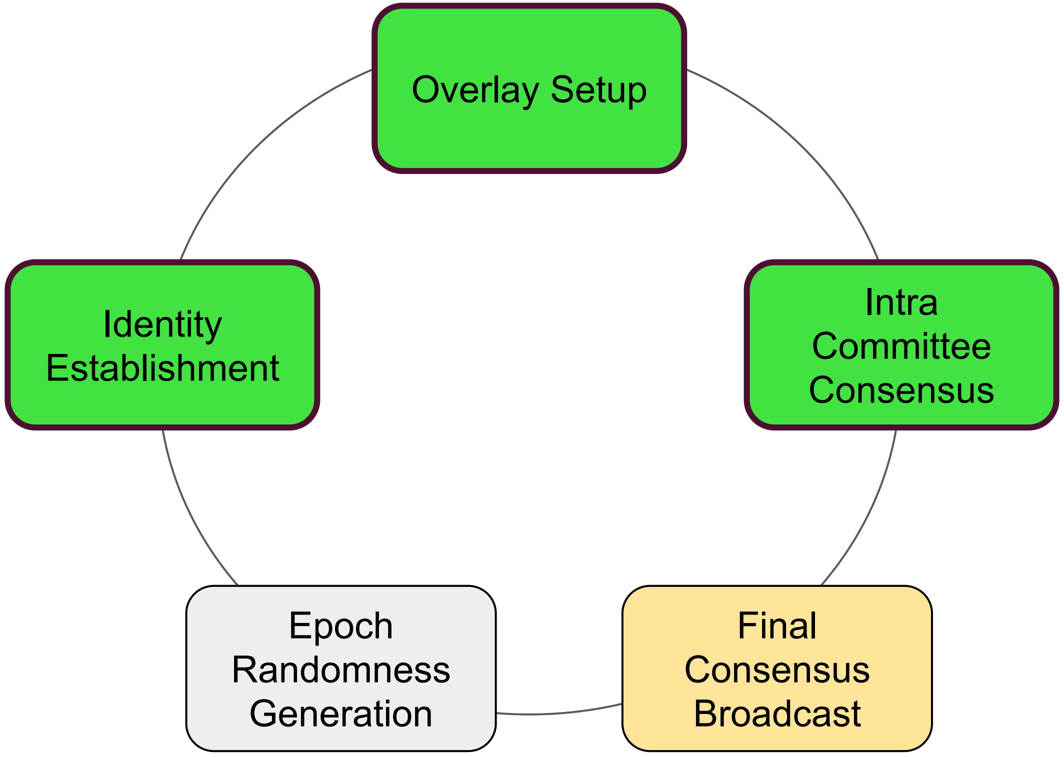
- Run a Byzantine agreement protocol (PBFT or PolyByz).

- Number of messages in each committee = $O(C^2)$

[C is the Committee Size]

Better when C is less

Elastico


Overlay Setup
Intra Committee Consensus
Final Consensus Broadcast
Epoch Randomness Generation
Identity Establishment

Elastico

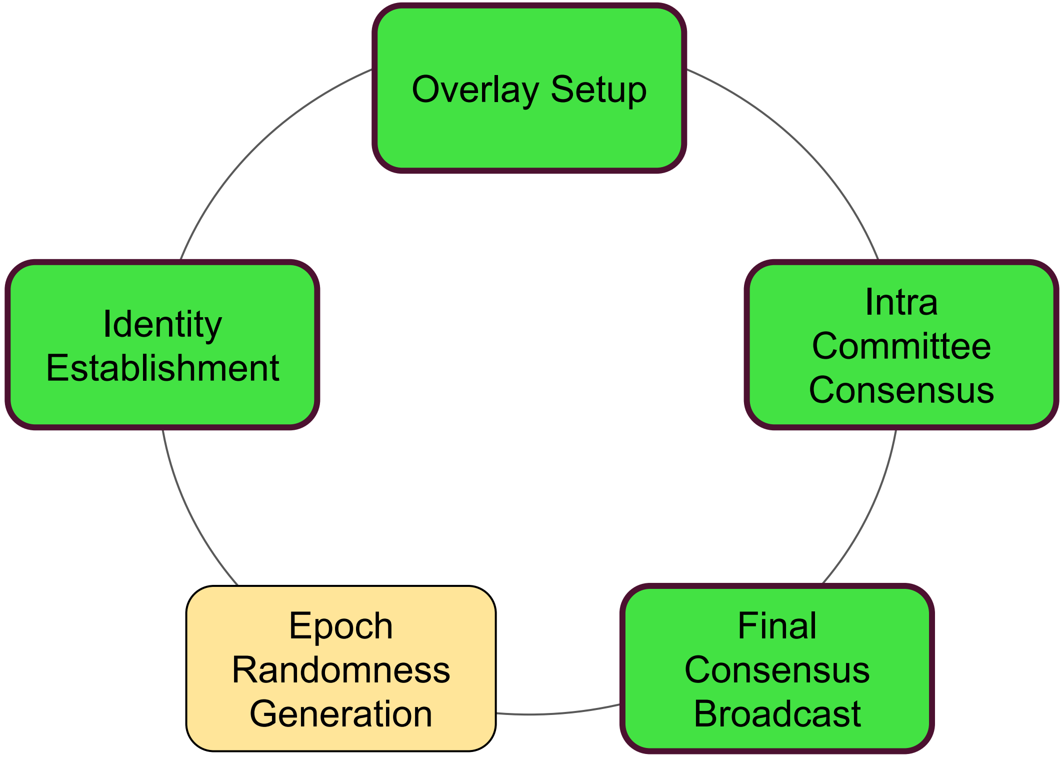
Final Consensus Broadcast

- Form a final committee (consensus committee) to reach final consensus.

- The nodes constituting the final committee comprise all members possessing a fixed *s-bit* committee ID.

- Run BFT, and Broadcast the final block to everyone.

Elastico


Overlay Setup
Intra Committee Consensus
Final Consensus Broadcast
Epoch Randomness Generation
Identity Establishment

Elastico

Epoch Randomness Generation (Commit-And-XOR)

- Each member of the final committee pick a random “R_i”

- Include $H(R_i)$ in final block.

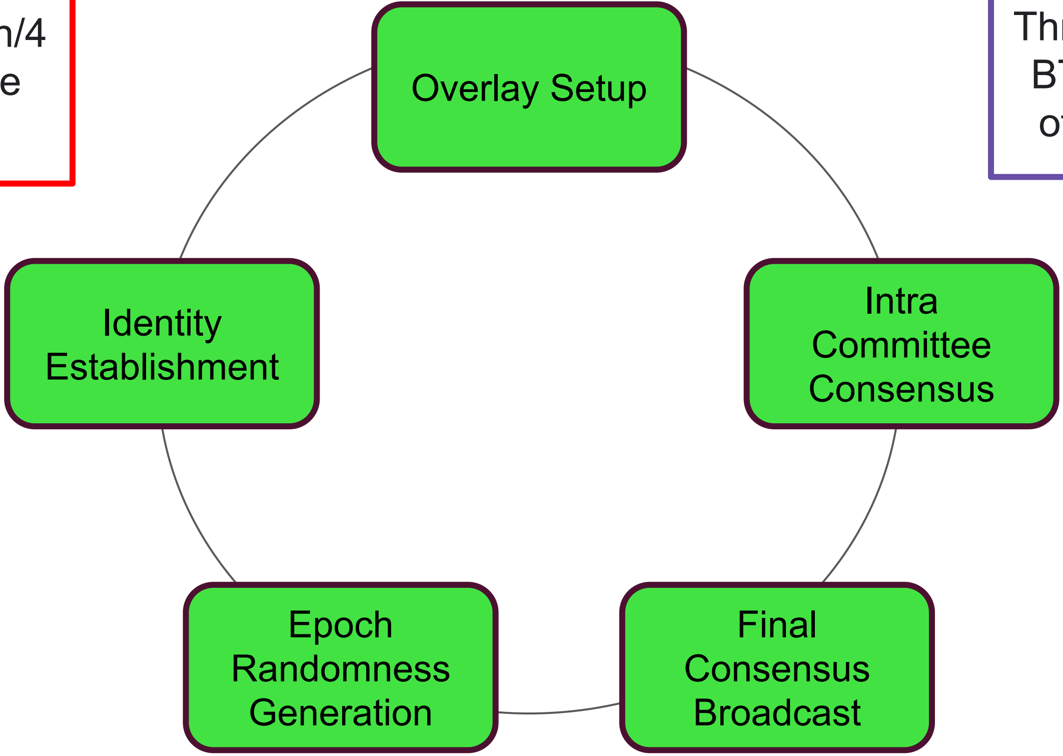
- Broadcast R_i with final block.

- Nodes XOR any set of 2C/3 “Ri” to get randomness.

Elastico

Tolerates up to $f < n/4$ adaptive byzantine adversaries

Throughput of 6x than BTC with 16 Shards of 100 nodes each.


Overlay Setup
Intra Committee Consensus
Final Consensus Broadcast
Epoch Randomness Generation
Identity Establishment

Elastico

Limitations

- ➔ PoW method used for identification is susceptible to bias since miners has the ability to drop blocks, thus influencing randomness.
- ➔ The distributed commit-and-xor scheme is not a perfectly unbiased randomness generator.
- ➔ High failure rate, with total fault tolerance of 25%. Can be mitigated by increasing the consensus group but at the expense of communication overhead.
- ➔ No Cross-Shard transaction. Global state is maintained in each shard. High storage overhead.

RapidChain [CCS '18]

Improved Security and Throughput over Elastico

- Tolerates up to $f < n/3$ in total network and $f < n/2$ in intra-shard, where f is byzantine adversaries.

- Each node exclusively stores a fraction of *1/k* of the ledger [*k signifies the number of shards*].

- Cross-Shard Protocol for efficient cross-shard transaction.

- Decentralized Bootstrapping - Does not assume the existence of an initial randomness.

RapidChain

Decentralized Bootstrapping

RapidChain proposes a decentralized bootstrapping in the form of sampler-graph election network, with only a hardcoded seed and some network settings.

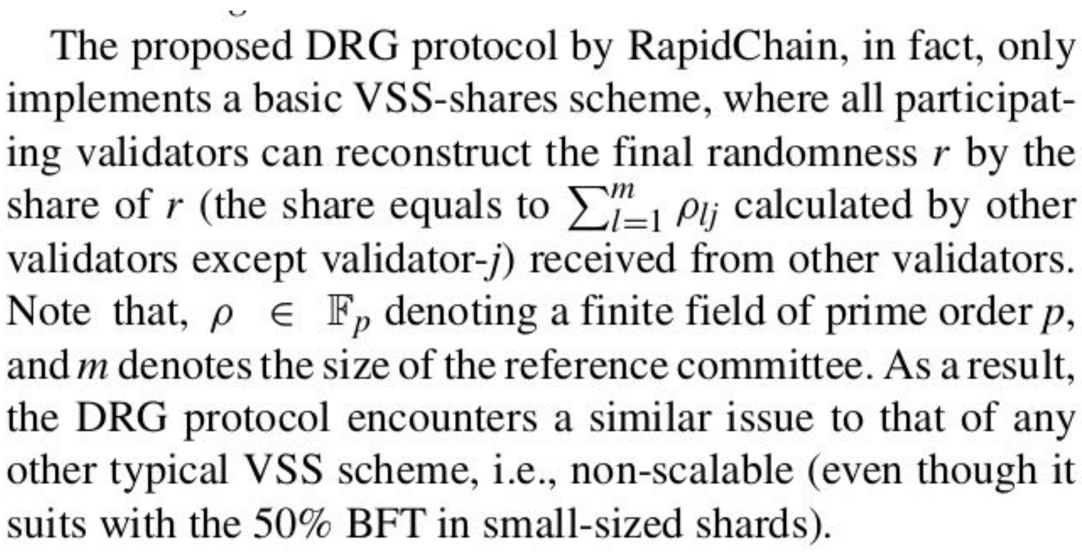
In such an election network, participating validators are uniformly distributed into a few groups, within each of which a PoW-based result is computed by each member based on the randomness generated by the VSS-based DRG protocol and its identification ID.

Based on the result, a subgroup can be obtained for each group. Finally, a unique root group (it randomly selects the members of the reference committee) can be obtained with 50% honest majority (high probability), when this process is iterated. Consequently, the communication overhead can be improved from □(n2) to O(nroot(n)) with n denoting the total number of participating validators.

RapidChain

Epoch Reconfiguration

RapidChain implements a VSS-based distributed random generation (DRG) protocol to agree on an unbiased randomness.



The proposed DRG protocol by RapidChain, in fact, only implements a basic VSS-shares scheme, where all participating validators can reconstruct the final randomness r by the share of r (the share equals to $\sum_{l=1}^{m} \rho_{lj}$ calculated by other validators except validator-j) received from other validators. Note that, $\rho \in \mathbb{F}_p$ denoting a finite field of prime order p, and m denotes the size of the reference committee. As a result, the DRG protocol encounters a similar issue to that of any other typical VSS scheme, i.e., non-scalable (even though it suits with the 50% BFT in small-sized shards).

RapidChain

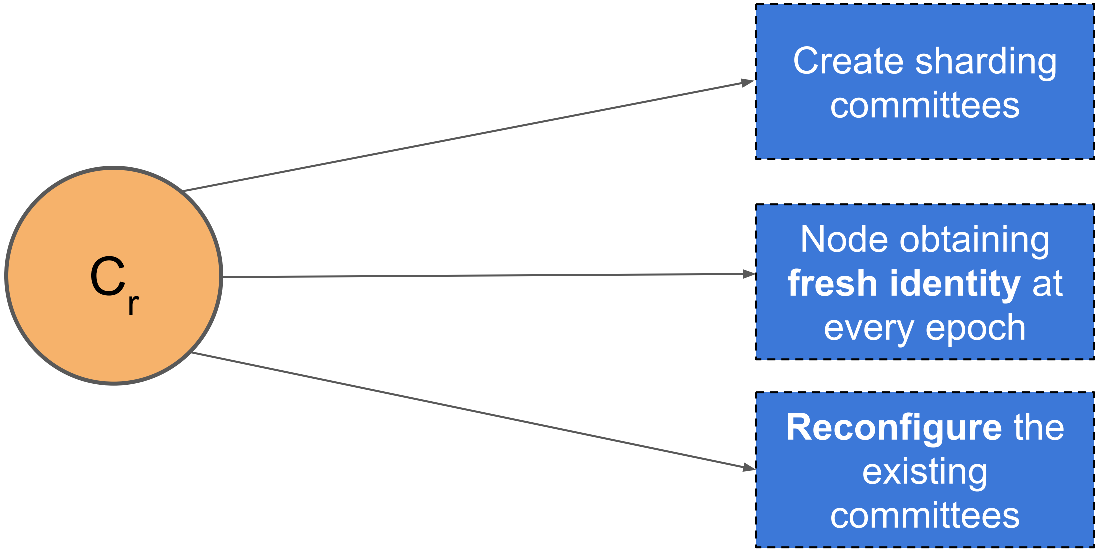
Intra-Consensus

- Runs an autonomous pre-scheduled scheme within a shard to agree on a timeout Δ.
- Based on the timeout, the consensus speed can be adjusted by the system to prevent the asynchronization.
- A non-responsive synchronous (with constant rounds) BFT-based consensus protocol with FT of 50% can be used.

RapidChain

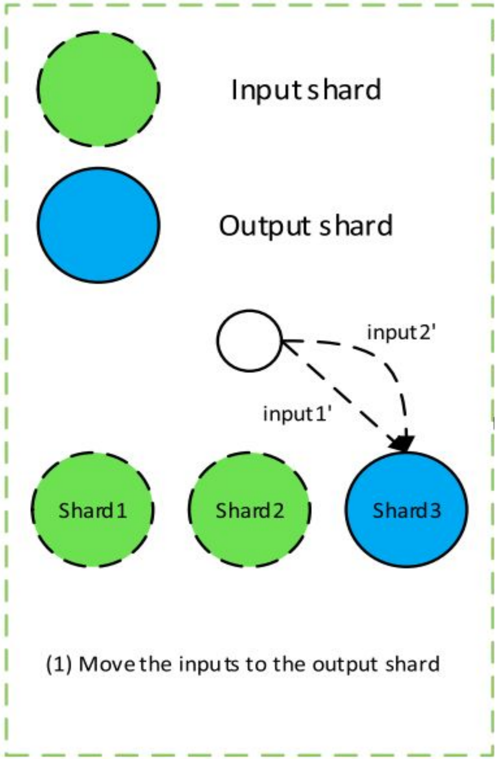
Reference Committee (C_r)

At the conclusion of each epoch ‘i’,
C_r generates a fresh randomness denoted as r_{i+1},
serving as the epoch randomness for the subsequent epoch i + 1.


C_r
Create sharding committees
Node obtaining **fresh identity** at every epoch
Reconfigure the existing committees

RapidChain

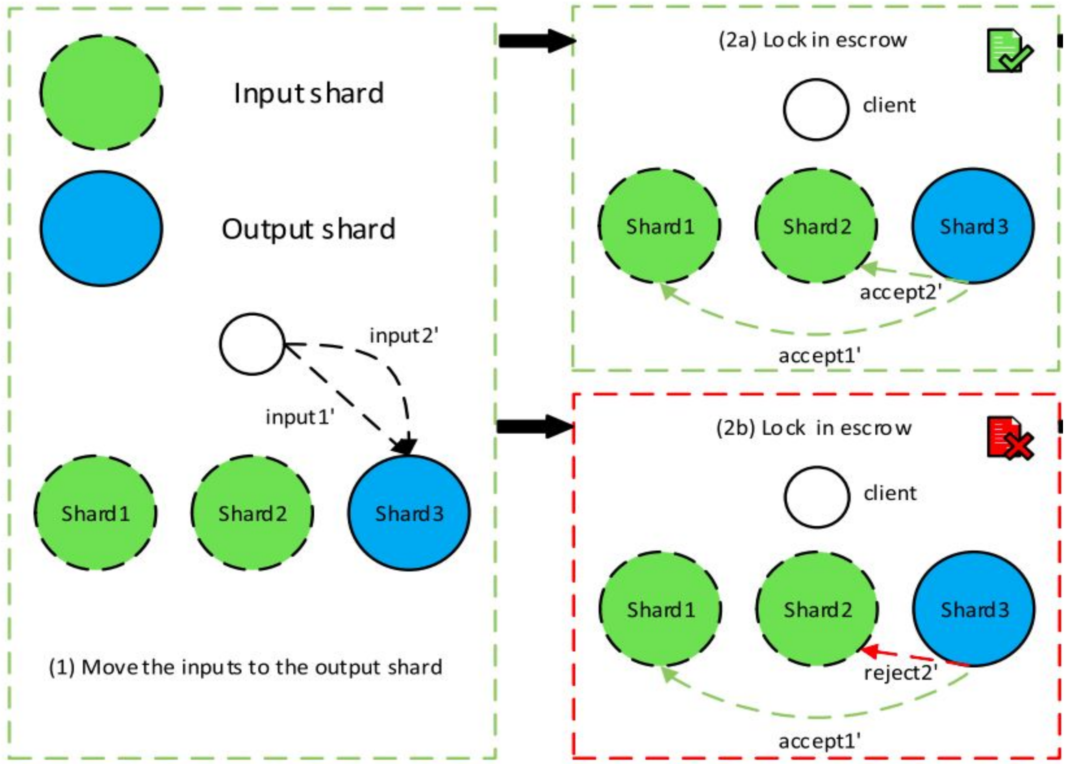
Cross-Shard Transaction



Zamani, Mahdi, Mahnush Movahedi, and Mariana Raykova. "Rapidchain: Scaling blockchain via full sharding."

RapidChain

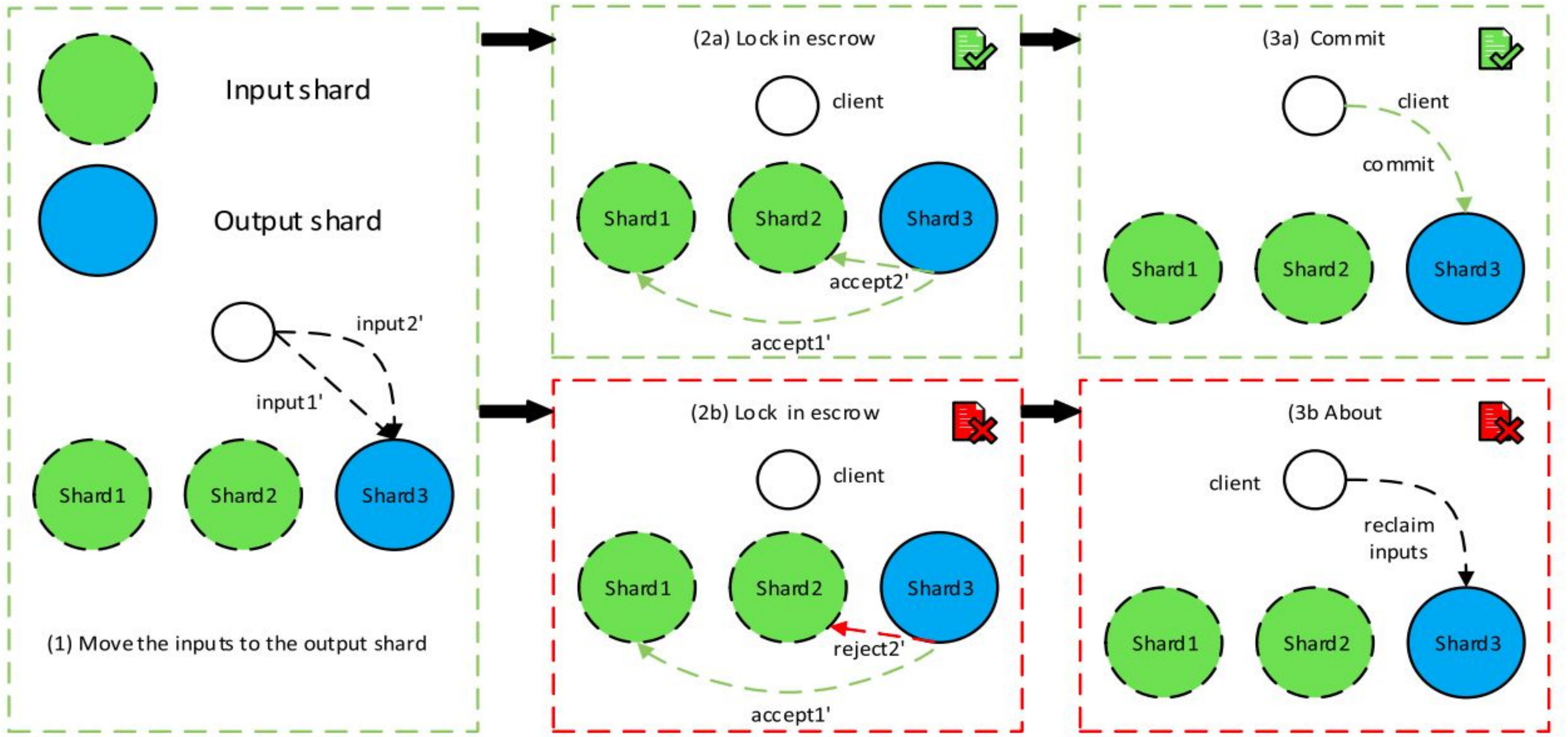
Cross-Shard Transaction



Zamani, Mahdi, Mahnush Movahedi, and Mariana Raykova. "Rapidchain: Scaling blockchain via full sharding."

RapidChain

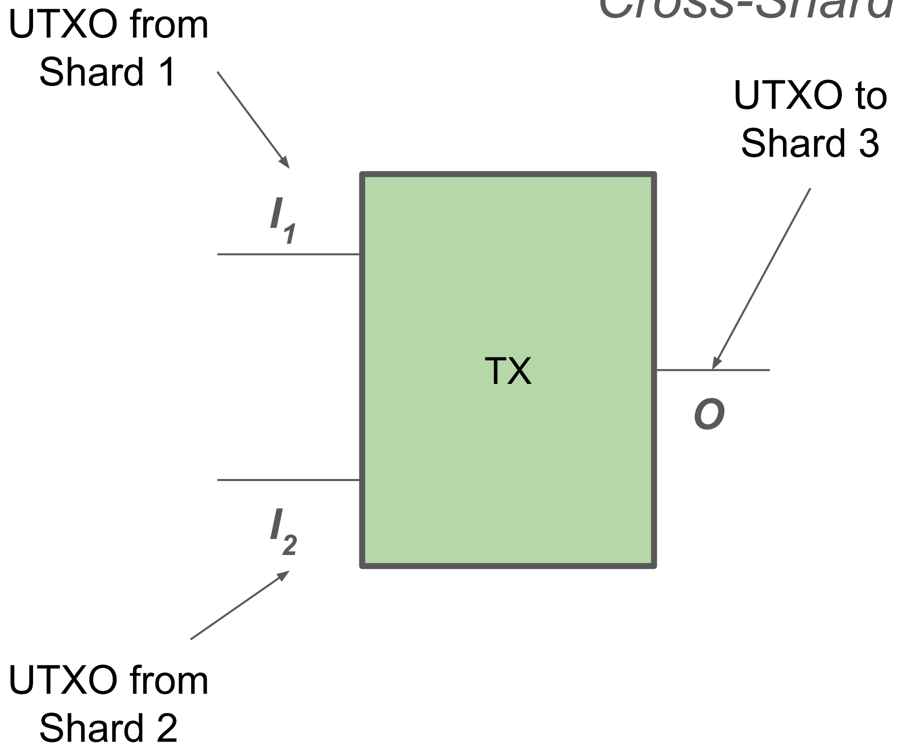
Cross-Shard Transaction



Zamani, Mahdi, Mahnush Movahedi, and Mariana Raykova. "Rapidchain: Scaling blockchain via full sharding."

RapidChain

Cross-Shard Transaction



"TX" = <$(I_1,I_2),O$>

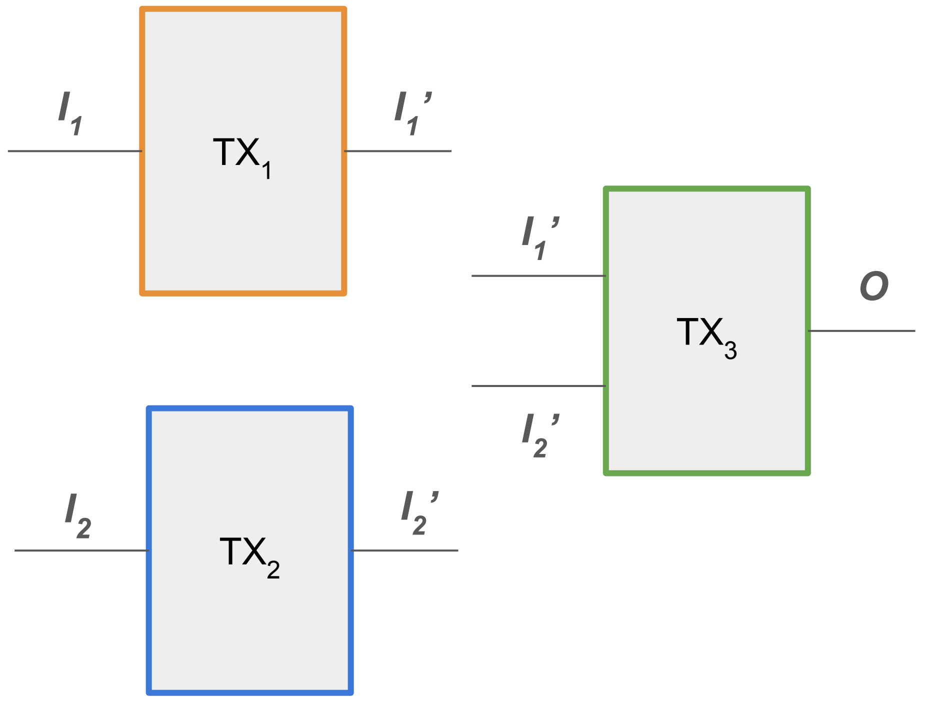
Input I_1 (from shard S_1)

Input I_2 (from shard S_2)

Output O (to shard S_3).

RapidChain

Cross-Shard Transaction



S_3, creates 3 New Transactions:

TX$_1$ = <*I1,I1'*>

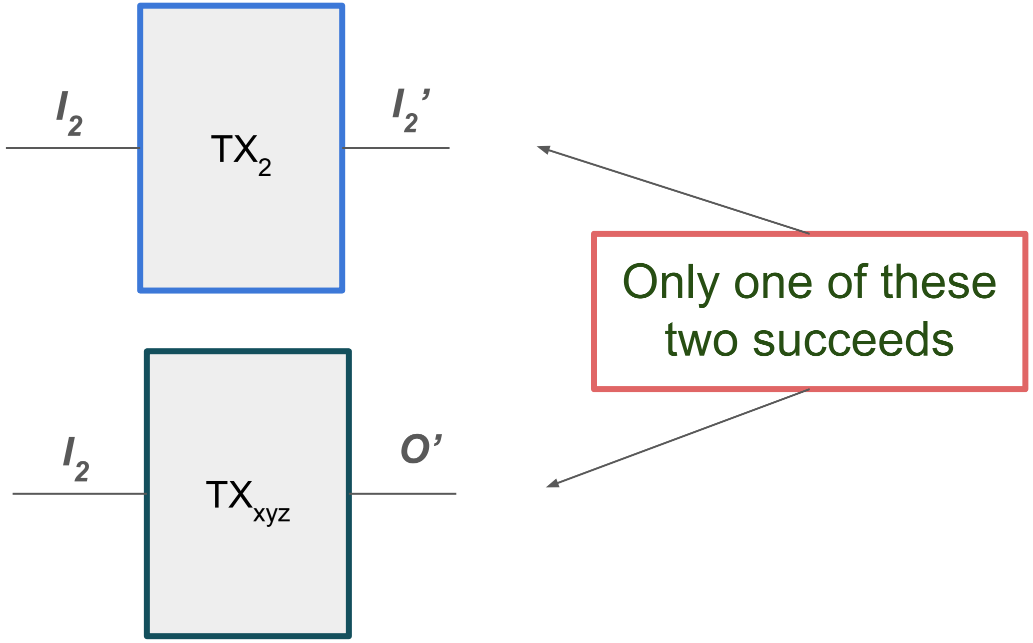
TX$_2$ = <*I2,I2'*>

TX$_3$ = <*(I1',I2'), O*>

RapidChain

Limitations

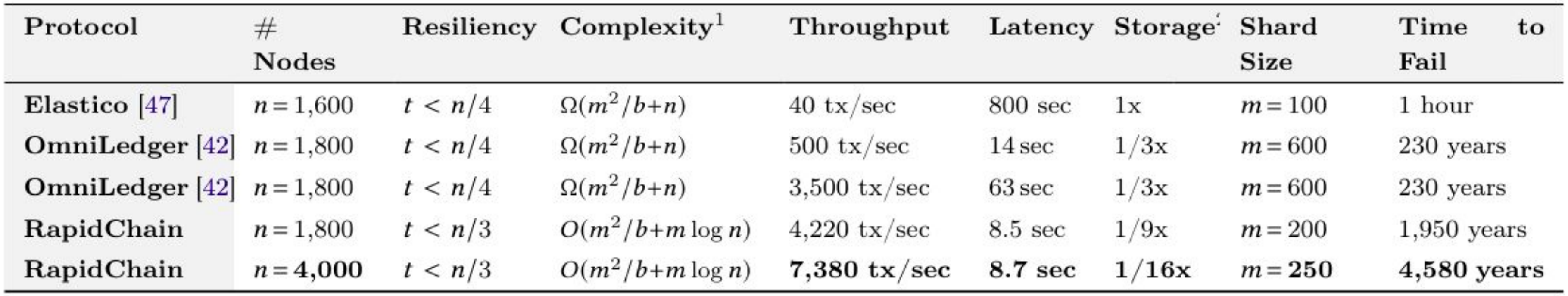
➔ Rapidchain's inability to maintain isolation is evident in certain scenarios.



If TX_{xyz} fails while TX_2 succeeds but TX_1 fails, none of the transactions would be successfully processed.

RapidChain

Throughput Comparison with Elastico and Omniledger



Protocol	**# Nodes**	**Resiliency**	**Complexity**[1]	**Throughput**	**Latency**	**Storage**[2]	**Shard Size**	**Time to Fail**
Elastico [47]	$n = 1{,}600$	$t < n/4$	$\Omega(m^2/b+n)$	40 tx/sec	800 sec	1x	$m = 100$	1 hour
OmniLedger [42]	$n = 1{,}800$	$t < n/4$	$\Omega(m^2/b+n)$	500 tx/sec	14 sec	1/3x	$m = 600$	230 years
OmniLedger [42]	$n = 1{,}800$	$t < n/4$	$\Omega(m^2/b+n)$	3,500 tx/sec	63 sec	1/3x	$m = 600$	230 years
RapidChain	$n = 1{,}800$	$t < n/3$	$O(m^2/b+m \log n)$	4,220 tx/sec	8.5 sec	1/9x	$m = 200$	1,950 years
RapidChain	$n = \mathbf{4{,}000}$	$t < n/3$	$O(m^2/b+m \log n)$	**7,380 tx/sec**	**8.7 sec**	**1/16x**	$m = \mathbf{250}$	**4,580 years**

Zamani, Mahdi, Mahnush Movahedi, and Mariana Raykova. "Rapidchain: Scaling blockchain via full sharding."

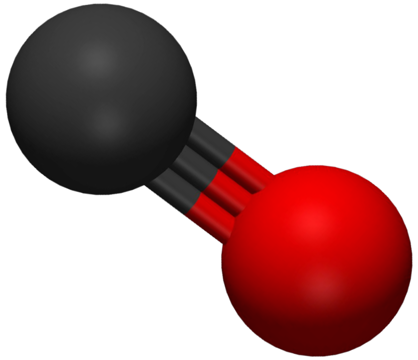
RapidChain

Limitations

- ➔ With the increment of shards, nearly all transactions evolve into cross-shard transactions. Batch processing of cross-shard transactions provides partial alleviation.
- ➔ The latency increases by 1.5 times if 10 new nodes join the network for each shard.
- ➔ There is no provision for adaptive-based sharding in RapidChain. Restarting the entire network is imperative to modify the number of shards.
- ➔ RapidChain's methodology proves inadequate for non-UTXO distributed transactions as it violates both atomicity and isolation principles.

Monoxide [NSDI'19]

Addresses Hash Power dilution by a novel PoW based mining technique

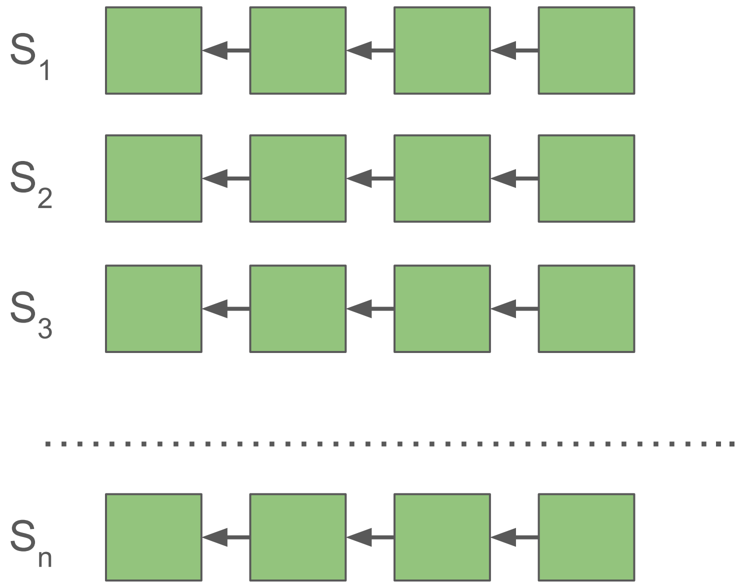


- ➔ Adopts the concept of asynchronous consensus zones, where each zone represents a shard.
- ➔ Follows Account based transaction model.
- ➔ A novel Chu-Ko-Nu mining technique rooted in the Nakamoto-based Proof-of-Work. (Fault Tolerance of 50%)
- ➔ Leverages Relay Transactions for cross-shard communication.

Monoxide

Chu-Ko-Nu Mining

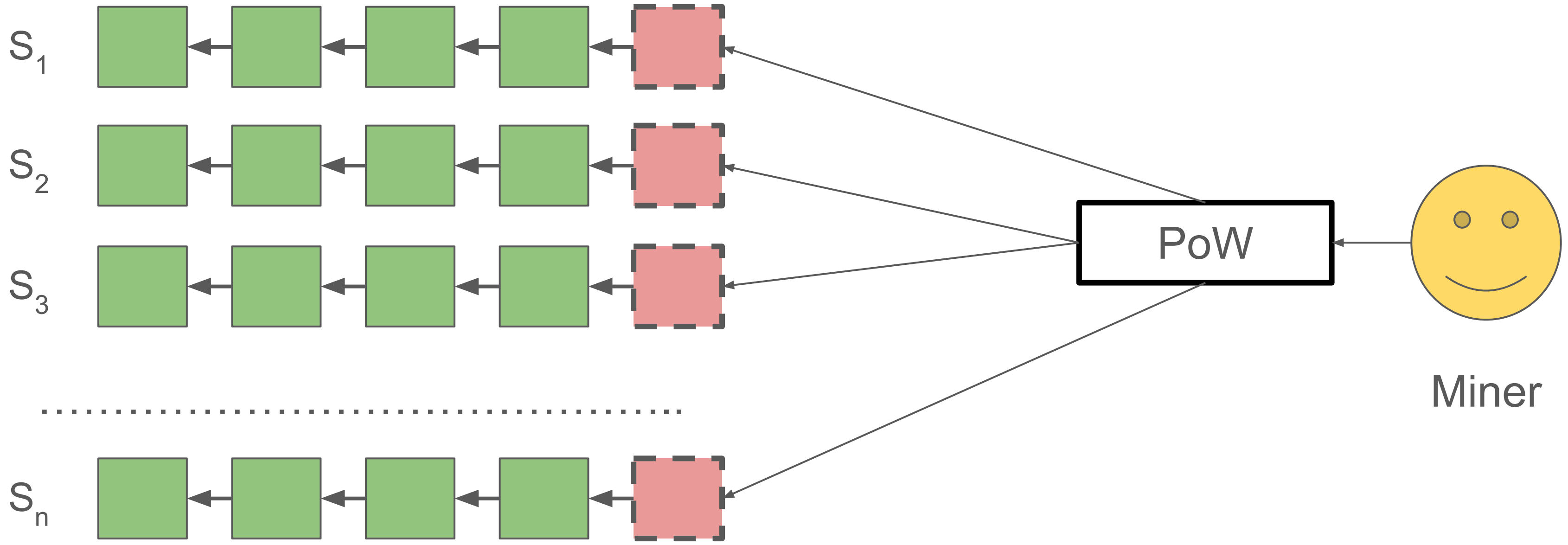
➔ Miners will be able to mine on multiple chains of different shards at the same time by solving one PoW target.



Monoxide

Chu-Ko-Nu Mining

➔ Miners will be able to mine on multiple chains of different shards at the same time by solving one PoW target.

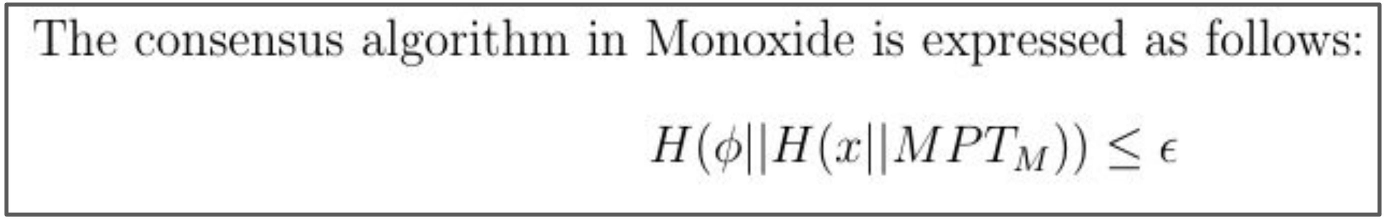


Monoxide

Chu-Ko-Nu Mining

➔ The Merkle Patricia tree (MPT) serves as the storage structure for all proposed blocks from multiple shards.

➔ MPT_M denotes the MPT root consisting of all proposed blocks of each involved shard, i.e., $[B_0, B_1, ..., B_{n-1}]$ if k = n.



The consensus algorithm in Monoxide is expressed as follows:

$$H(\phi || H(x || MPT_M)) \leq \epsilon$$

H = Hash Function
φ = Random Nonce
x = Block header content
ε = PoW target difficulty.

Monoxide

Chu-Ko-Nu Mining

➔ The Chu-ko-nu mining algorithm enables miners in any shard to generate multiple blocks upon successfully solving the hash puzzle, limited to one block per shard.

  This amplifies the computing power of miners in each shard, thereby requiring an equivalent level of computing power to attack a single shard as attacking the entire system.

Monoxide

Chu-Ko-Nu Mining

- ➔ Monoxide mandates that most miners in each shard engage in Chu-ko-nu mining (i.e., $k \rightarrow n$) to satisfy $P^*(k/n) \approx P$.

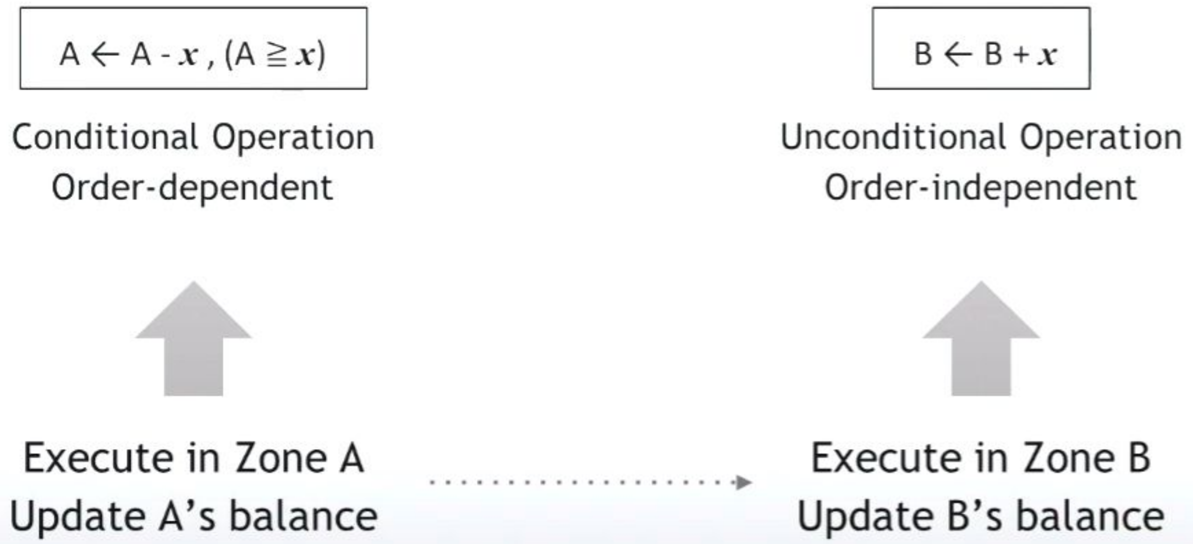
  “P” denotes the total amount of mining power,
  “n” signifies the number of shards.

- ➔ This requirement is essential because scattered mining power may not ensure the security of a single shard.

Monoxide

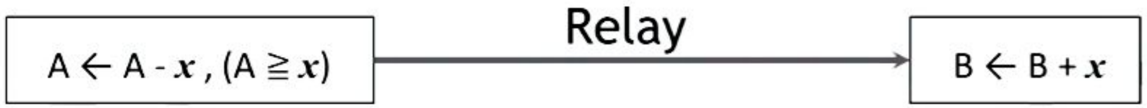
Cross-Zone Transactions

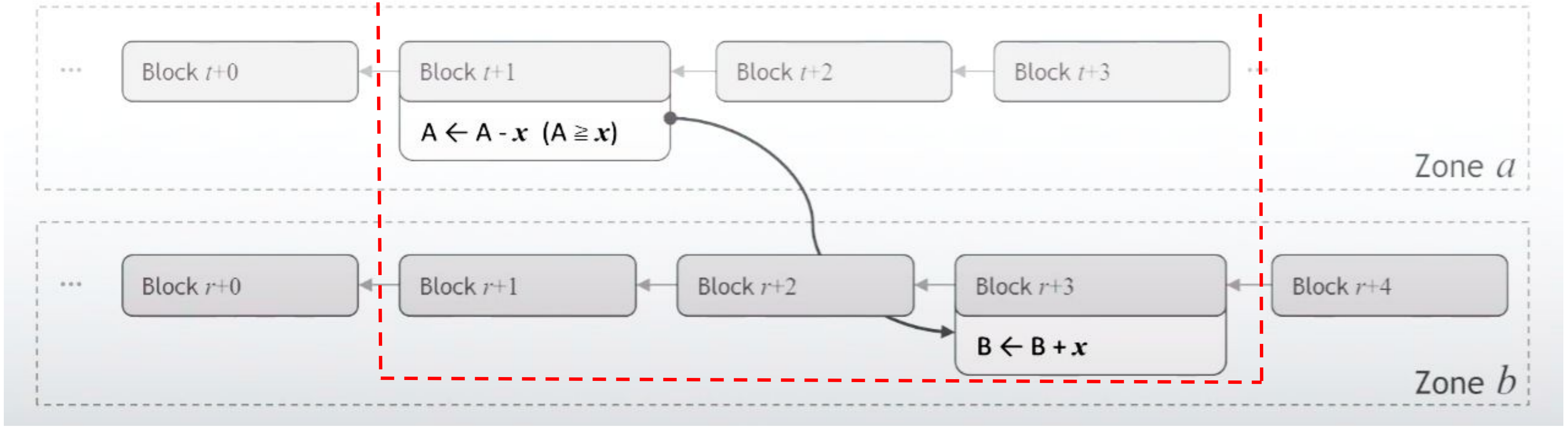
Transfer x tokens from user **A** to user **B** in different zones



Monoxide

Cross-Zone Transactions





Monoxide

Limitations

- ➔ If miners only mine on "k" out of "n" shards, where (k << n)
  - ◆ The factor expected to amplify the effective mining power will be too small to secure the mining process, hence reducing the attack cost.
- ➔ Rational miners tend to mine on all "n" shards to reap the maximum profit
  - ◆ May also result in the power centralization due to the huge cost of bandwidth, disk storage, and computing processors that only the professional mining facilities can afford.

GearBox [CCS' 22]

Adaptively Change Shard Size based on Adversary Population

- ➔ Segregates the concepts of liveness and safety within shard consensus.

GearBox [CCS' 22]

Adaptively Change Shard Size based on Adversary Population

- ➔ Segregates the concepts of liveness and safety within shard consensus.
- ➔ Dynamic adjustment of shard parameters to achieve optimal efficiency given the current corruption ratio of the system.

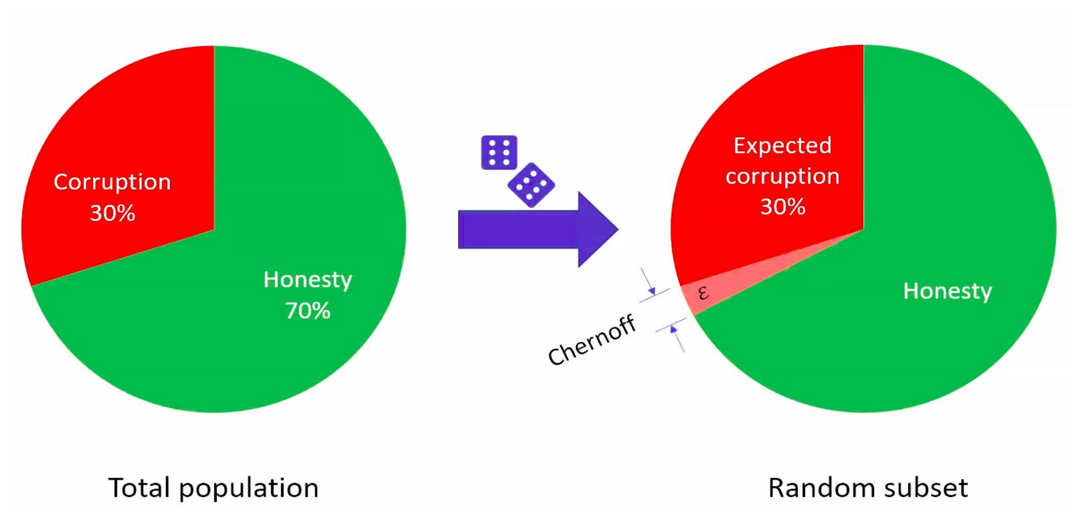
GearBox [CCS' 22]

Adaptively Change Shard Size based on Adversary Population

- ➔ Segregates the concepts of liveness and safety within shard consensus.
- ➔ Dynamic adjustment of shard parameters to achieve optimal efficiency given the current corruption ratio of the system.
- ➔ Shards identified as not meeting liveness requirements are subject to resampling with increased shard size and liveness tolerance.

GearBox

Chernoff Bound



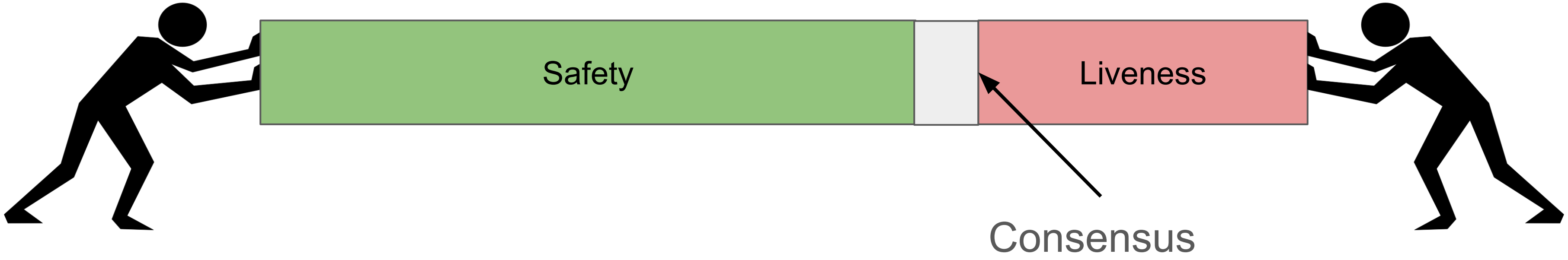
- ➔ Sampling a random subset may not always provide the expected corruption level.
- ➔ Chernoff bound offers a means to limit this additional corruption (ϵ) to an arbitrarily small value when the random subset is sufficiently large.

GearBox

Safety-Liveness Dichotomy

Safety Threshold (S)
How much adversary can my network handle without sacrificing safety?

Liveness Threshold (L)
How much adversary can my network handle without sacrificing liveness?



GearBox

Safety-Liveness Dichotomy

When S = L

GearBox

Safety-Liveness Dichotomy

When S = L

Synchronous Model

$S < 1 - L$

$S + L < 1$

$L,S < \frac{1}{2}$

Partially Synchronous Model

$S < 1 - 2L$

$S + 2L < 1$

$L,S < \frac{1}{3}$

GearBox

Safety-Liveness Dichotomy

When S = L

Synchronous Model

$S + L < 1$
$L,S < \frac{1}{2}$

Partially Synchronous Model

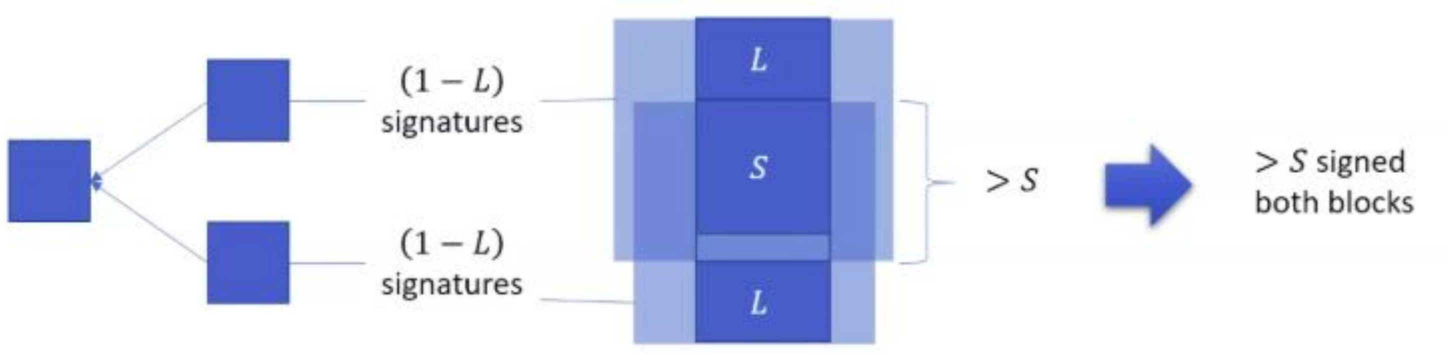
$S + 2L < 1$
$L,S < \frac{1}{3}$

To ensure liveness during block commitment, the block must be endorsed by a fraction of nodes equal to 1 − L, where the proportion of corrupted nodes is no more than L.

GearBox

Safety-Liveness Dichotomy

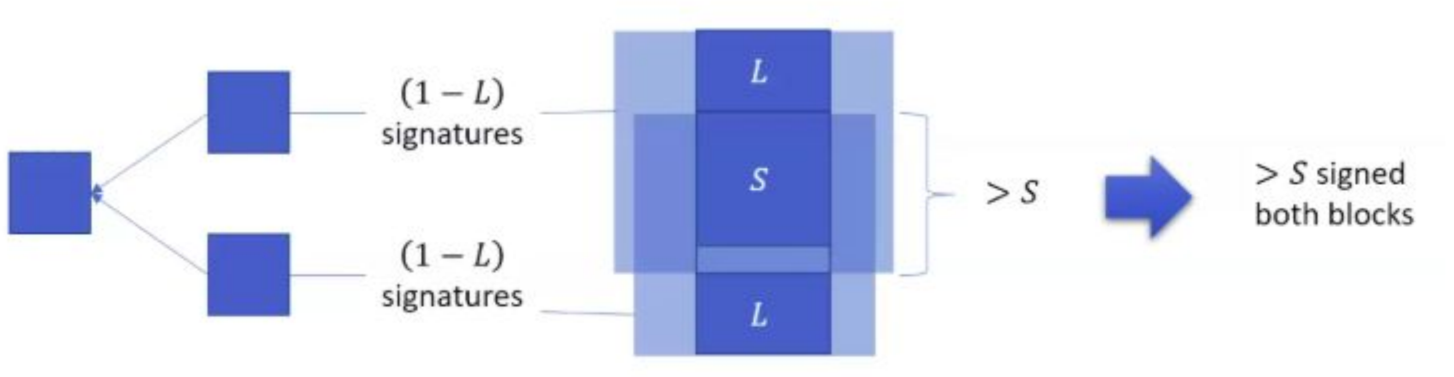
In the case of two conflicting blocks -



If “1−L” of the nodes have endorsed both blocks, there will be an intersection among the endorsers indicating that more than “S” parties have endorsed both blocks.

GearBox

Safety-Liveness Dichotomy



As “S” represents the maximum number of malicious parties, there must be at least one honest node that endorses both blocks, thus leading to the detection of the conflict.

GearBox

Safety-Liveness Dichotomy

<u>To run a consensus based on the dichotomy of this partially synchronous network $S + 2L < 1$</u>

The protocol needs to continuously change in L and S based on the malicious nodes present in the network.

GearBox

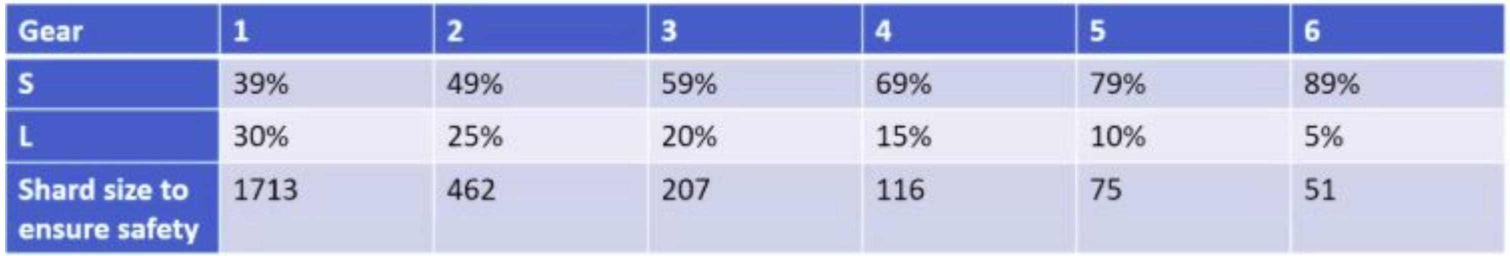
Gears for Changing Shard Size

Presents 6 gears to change the configuration of the network based on the situation.

GearBox

Gears for Changing Shard Size

Presents 6 gears to change the configuration of the network based on the situation.



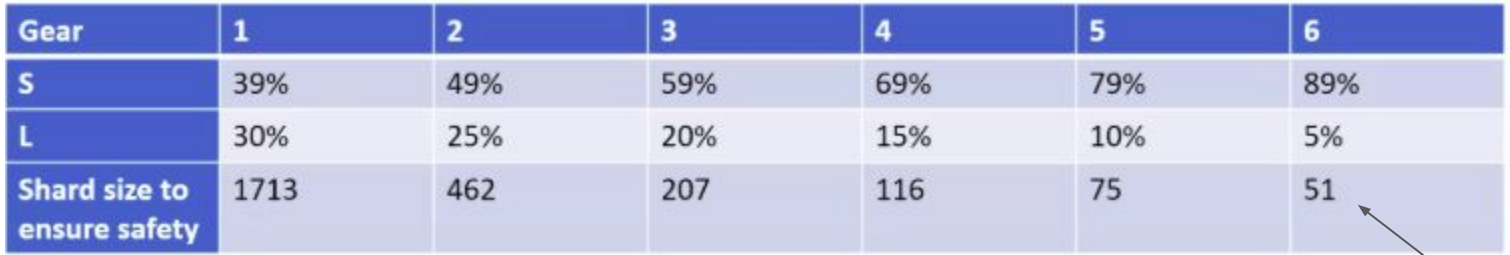
Gear	1	2	3	4	5	6
S	39%	49%	59%	69%	79%	89%
L	30%	25%	20%	15%	10%	5%
Shard size to ensure safety	1713	462	207	116	75	51

David, Bernardo, et al. "Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy."

GearBox

Gears for Changing Shard Size

Presents 6 gears to change the configuration of the network based on the situation.



Gear	1	2	3	4	5	6
S	39%	49%	59%	69%	79%	89%
L	30%	25%	20%	15%	10%	5%
Shard size to ensure safety	1713	462	207	116	75	51

David, Bernardo, et al. "Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy."

Will this get into Liveness Attack?

GearBox

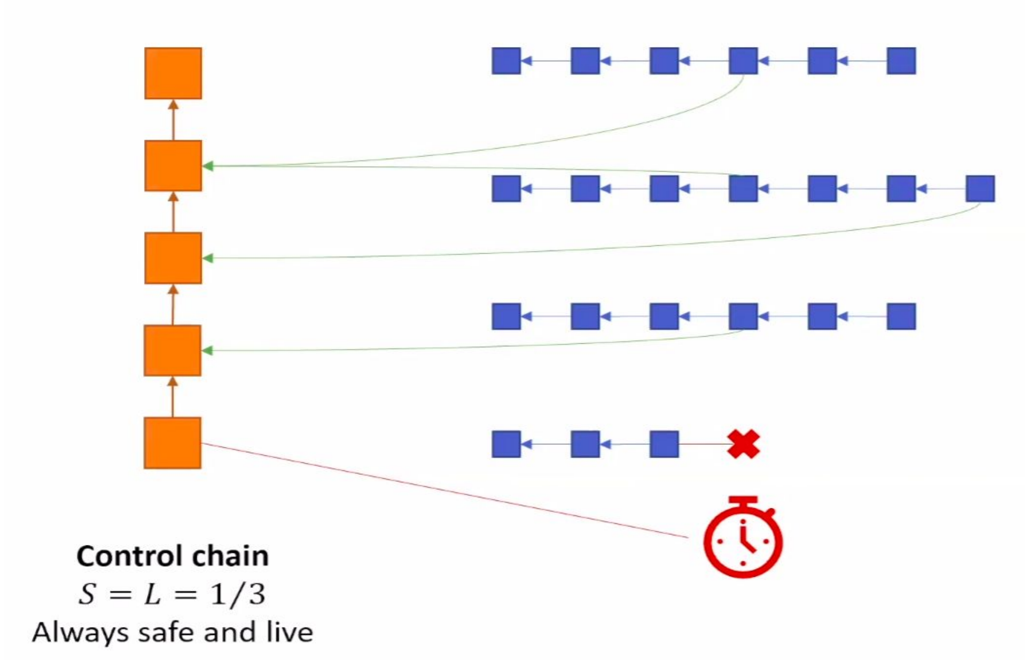
Shard Liveness Monitoring

Challenge: Monitor the liveness of each shard

GearBox

Shard Liveness Monitoring

<u>Challenge: Monitor the liveness of each shard</u>



GearBox

Limitations

Issues with overlapping shards:

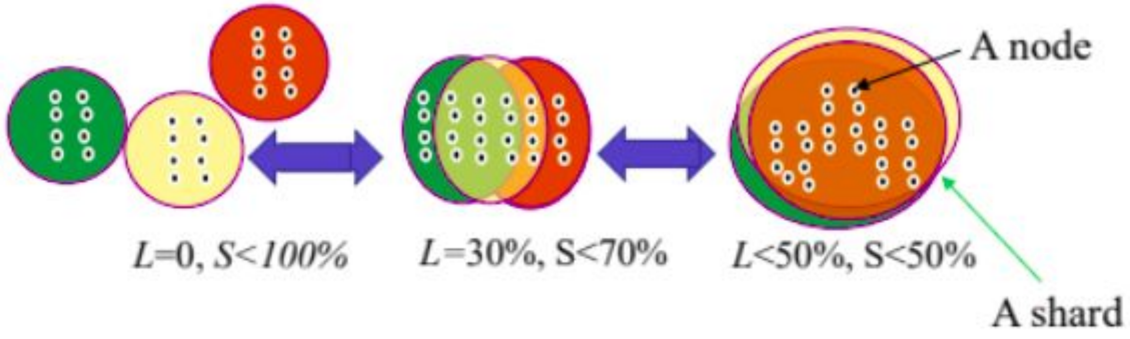
Since the protocol adjusts shard sizes without altering the total number of shards. These overlapping shards could lead to duplicated workloads, requiring additional transactions involving multiple shards. Consequently, there might be a rise in cross-shard transactions and a decrease in parallelism.

GearBox

Limitations

Issues with overlapping shards:

Since the protocol adjusts shard sizes without altering the total number of shards. These overlapping shards could lead to duplicated workloads, requiring additional transactions involving multiple shards. Consequently, there might be a rise in cross-shard transactions and a decrease in parallelism.



GearBox

Limitations

Simply altering shard sizes in both directions to accommodate dynamic changes in L is not feasible.

GearBox

Limitations

Simply altering shard sizes in both directions to accommodate dynamic changes in L is not feasible.

Attackers could exploit this method to trigger a loop of "switching gears", resulting in frequent and expensive shard adjustments.

Reticulum [NSDI' 2024]

A two-phase framework that dynamically adjusts transaction throughput in response to real-time adversarial attack.

Reticulum [NSDI' 2024]

A two-phase framework that dynamically adjusts transaction throughput in response to real-time adversarial attack.

- ➔ It comprises ’control’ and ’process’ shards organized into two layers corresponding to the two phases.

Reticulum [NSDI' 2024]

A two-phase framework that dynamically adjusts transaction throughput in response to real-time adversarial attack.

- ➔ It comprises ’control’ and ’process’ shards organized into two layers corresponding to the two phases.
- ➔ Process shards are subsets of control shards.

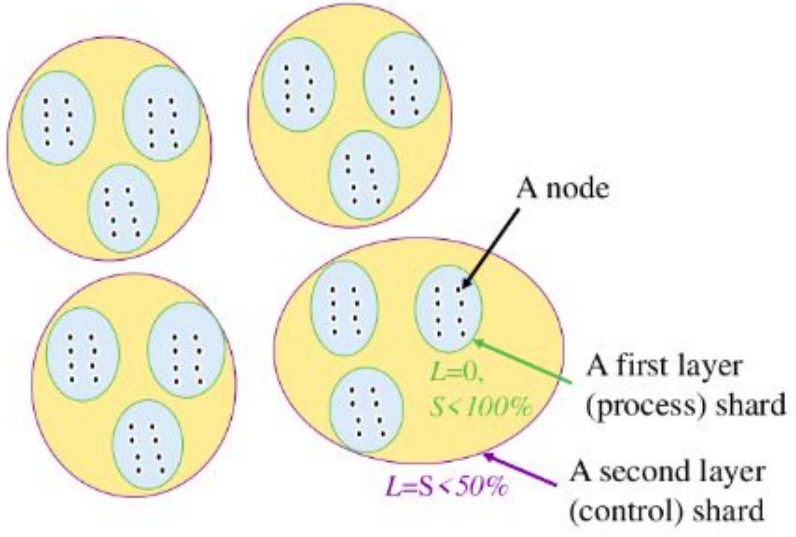
Reticulum [NSDI' 2024]

A two-phase framework that dynamically adjusts transaction throughput in response to real-time adversarial attack.

- ➔ It comprises ’control’ and ’process’ shards organized into two layers corresponding to the two phases.
- ➔ Process shards are subsets of control shards.
- ➔ Process shards leverages Safety while Control Shards leverages Liveness.

Reticulum

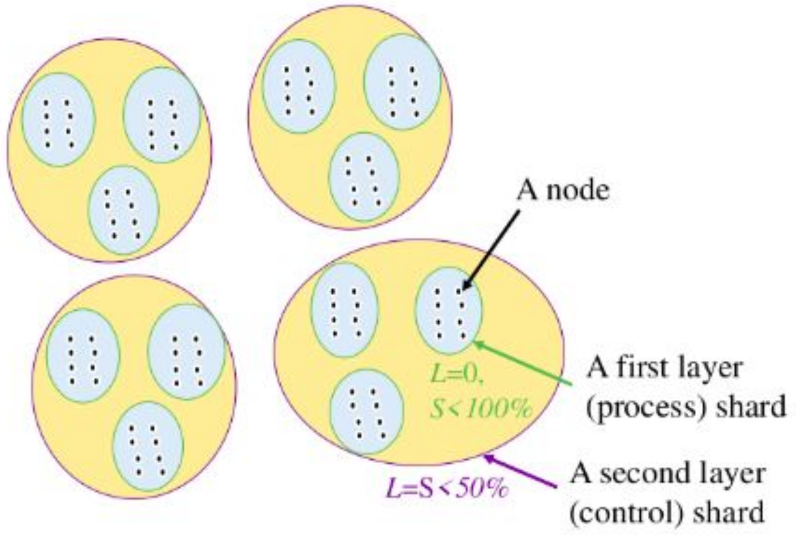
Two-Phase-Shard Consensus



Reticulum

Two-Phase-Shard Consensus

- ➔ Each **Process Shard** expected to include at least one honest node with high confidence.

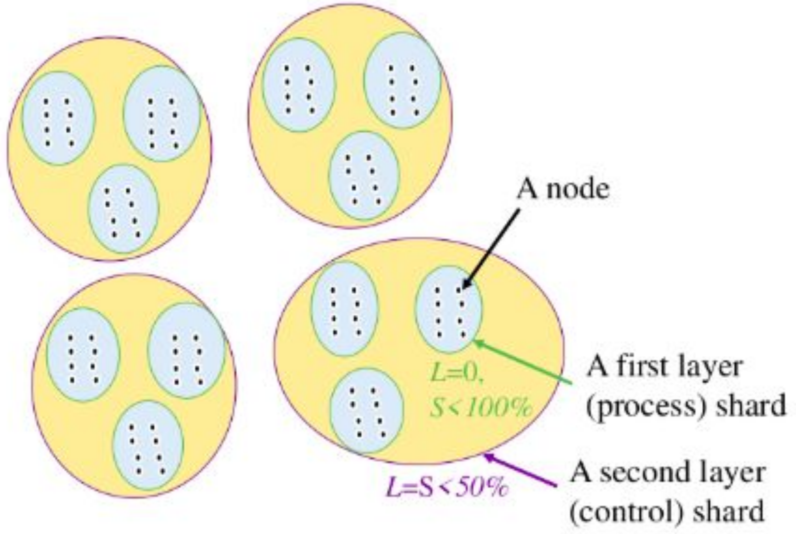


Reticulum

Two-Phase-Shard Consensus

➔ Each **Process Shard** expected to include at least one honest node with high confidence.

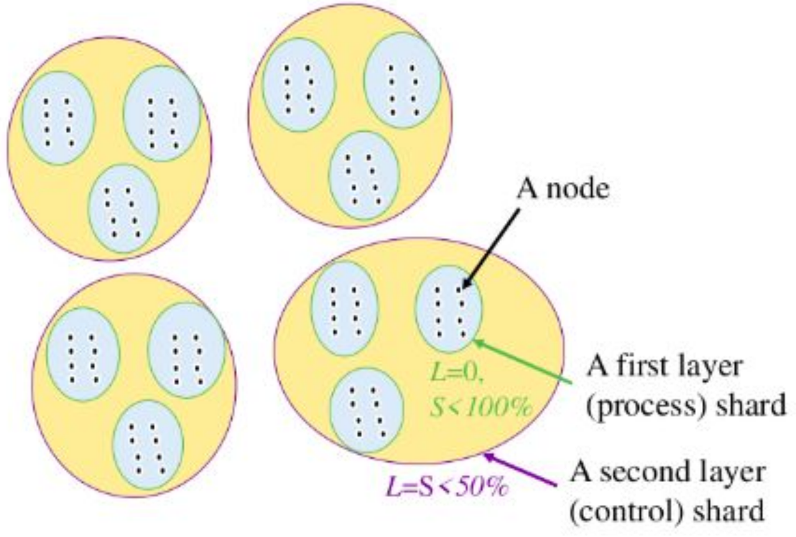
➔ Process Shards follows ***unanimous voting*** to engage fewer nodes in the First Phase.



Reticulum

Two-Phase-Shard Consensus

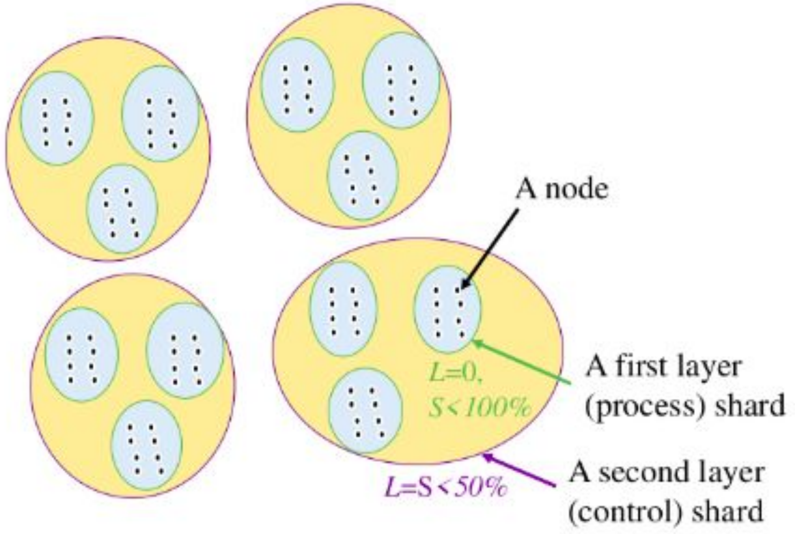
- ➔ Each **Process Shard** expected to include at least one honest node with high confidence.
- ➔ Process Shards follows ***unanimous voting*** to engage fewer nodes in the First Phase.
- ➔ Each **Control Shard** expected to consist of a majority of honest nodes with high confidence.



Reticulum

Two-Phase-Shard Consensus

- ➔ Each **Process Shard** expected to include at least one honest node with high confidence.
- ➔ Process Shards follows ***unanimous voting*** to engage fewer nodes in the First Phase.
- ➔ Each **Control Shard** expected to consist of a majority of honest nodes with high confidence.
- ➔ Control Shard then finalizes the decision made in the first phase.



Reticulum

The First Phase

➔ Unanimous voting is required within each process shard to determine a block validity.

Reticulum

The First Phase

- ➔ Unanimous voting is required within each process shard to determine a block validity.
- ➔ If even one adversarial node is present within the process shard, consensus cannot be reached.

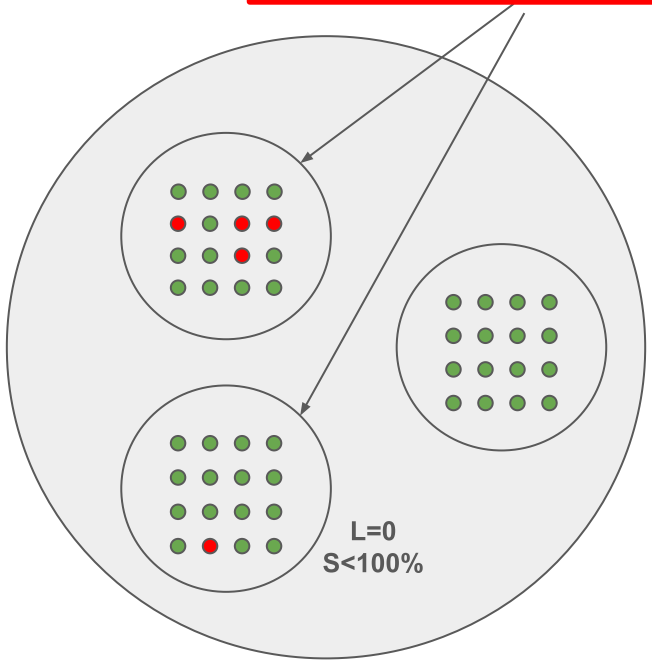
Reticulum

The First Phase

➔ Unanimous voting is required within each process shard to determine a block validity.

➔ If even one adversarial node is present within the process shard, consensus cannot be reached.

Process Shard fails even if one malicious node is there.

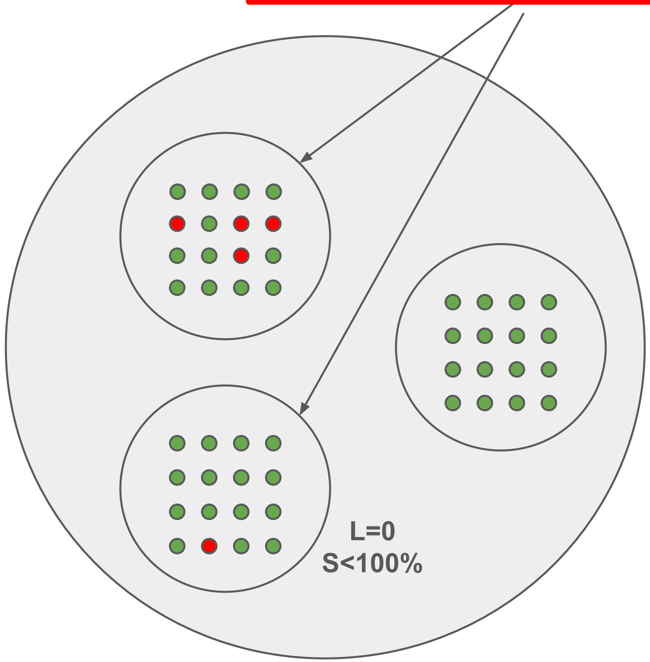


Reticulum

The First Phase

- ➔ Unanimous voting is required within each process shard to determine a block validity.
- ➔ If even one adversarial node is present within the process shard, consensus cannot be reached.
- ➔ Hence, L = 0, while S<100%.

Process Shard fails even if one malicious node is there.

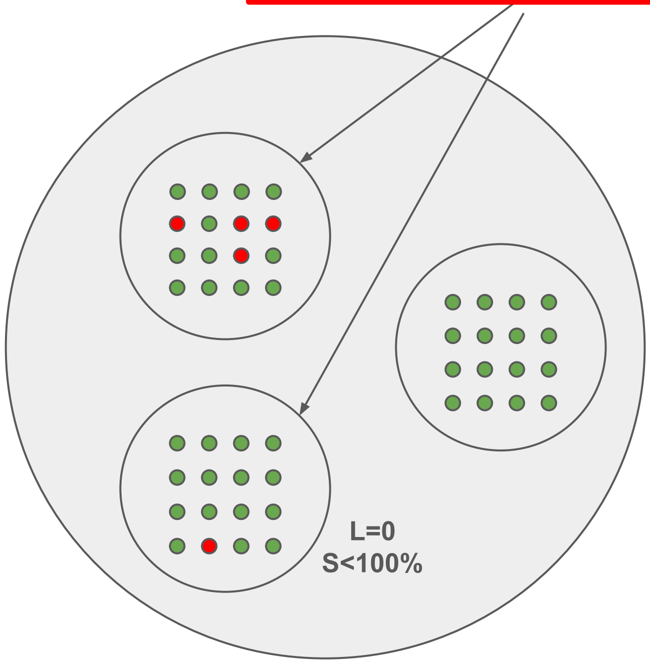


Reticulum

The First Phase

- ➔ Unanimous voting is required within each process shard to determine a block validity.
- ➔ If even one adversarial node is present within the process shard, consensus cannot be reached.
- ➔ Hence, L = 0, while S<100%.
- ➔ The first phase must be concluded within a predetermined time limit denoted as T_1.

Process Shard fails even if one malicious node is there.



Reticulum

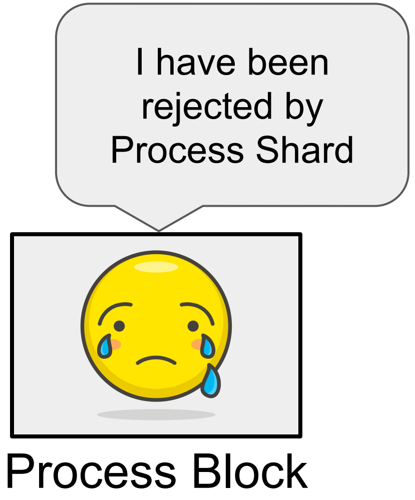
The Second Phase

➔ "Safety Net" for blocks that did not receive unanimous approval.

Reticulum

The Second Phase

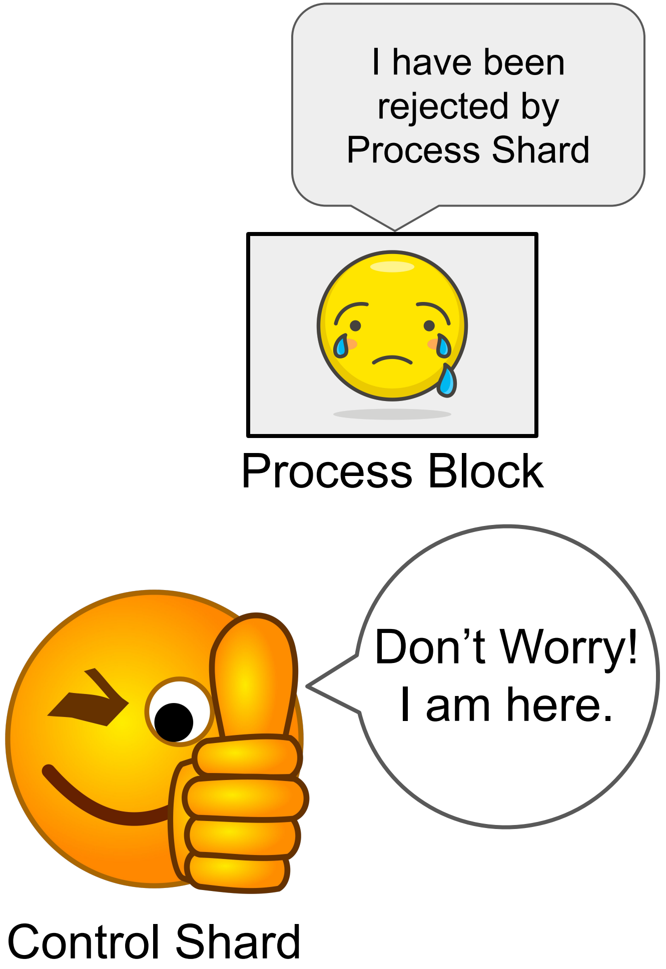
➔ "Safety Net" for blocks that did not receive unanimous approval.



Reticulum

The Second Phase

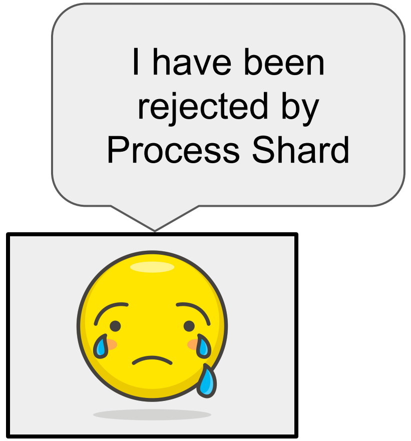
➔ "Safety Net" for blocks that did not receive unanimous approval.



Reticulum

The Second Phase

- ➔ "Safety Net" for blocks that did not receive unanimous approval.
- ➔ Leader from the Control Shard proposes a Control Block with details about the Process Blocks from each shard under its jurisdiction.



Process Block

Control Shard

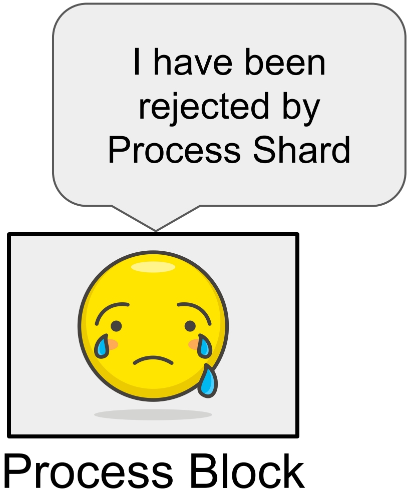
Reticulum

The Second Phase

➔ "Safety Net" for blocks that did not receive unanimous approval.

➔ Leader from the Control Shard proposes a Control Block with details about the Process Blocks from each shard under its jurisdiction.

➔ The second phase must be completed within T2, which, unlike T1, is dynamically adjusted based on the number of successful process shards within the control shard.



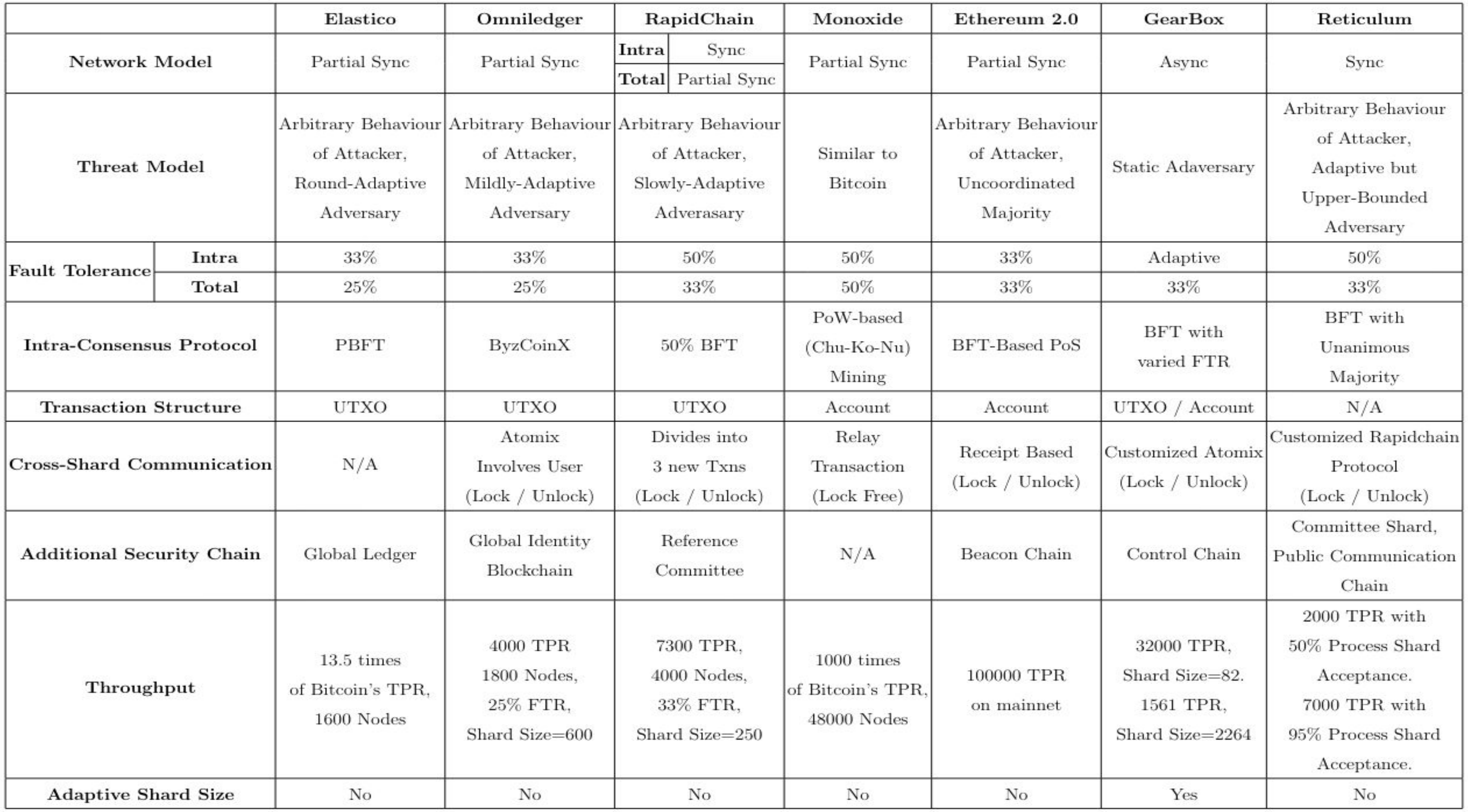
Reticulum

Drawbacks

➔ Uses a static sharding scheme, i.e. no new members can be added in runtime and no respawn of shards can take place. A system reboot is necessary for fresh node addition.

➔ No proof for cross-shard isolation is given in the paper. It could fail the same way RapidChain's isolation failed.

Comparison Table



		Elastico	Omniledger	RapidChain	Monoxide	Ethereum 2.0	GearBox	Reticulum
Network Model		Partial Sync	Partial Sync	**Intra** Sync **Total** Partial Sync	Partial Sync	Partial Sync	Async	Sync
Threat Model		Arbitrary Behaviour of Attacker, Round-Adaptive Adversary	Arbitrary Behaviour of Attacker, Mildly-Adaptive Adversary	Arbitrary Behaviour of Attacker, Slowly-Adaptive Adverasary	Similar to Bitcoin	Arbitrary Behaviour of Attacker, Uncoordinated Majority	Static Adaversary	Arbitrary Behaviour of Attacker, Adaptive but Upper-Bounded Adversary
Fault Tolerance	**Intra**	33%	33%	50%	50%	33%	Adaptive	50%
	Total	25%	25%	33%	50%	33%	33%	33%
Intra-Consensus Protocol		PBFT	ByzCoinX	50% BFT	PoW-based (Chu-Ko-Nu) Mining	BFT-Based PoS	BFT with varied FTR	BFT with Unanimous Majority
Transaction Structure		UTXO	UTXO	UTXO	Account	Account	UTXO / Account	N/A
Cross-Shard Communication		N/A	Atomix Involves User (Lock / Unlock)	Divides into 3 new Txns (Lock / Unlock)	Relay Transaction (Lock Free)	Receipt Based (Lock / Unlock)	Customized Atomix (Lock / Unlock)	Customized Rapidchain Protocol (Lock / Unlock)
Additional Security Chain		Global Ledger	Global Identity Blockchain	Reference Committee	N/A	Beacon Chain	Control Chain	Committee Shard, Public Communication Chain
Throughput		13.5 times of Bitcoin's TPR, 1600 Nodes	4000 TPR 1800 Nodes, 25% FTR, Shard Size=600	7300 TPR, 4000 Nodes, 33% FTR, Shard Size=250	1000 times of Bitcoin's TPR, 48000 Nodes	100000 TPR on mainnet	32000 TPR, Shard Size=82. 1561 TPR, Shard Size=2264	2000 TPR with 50% Process Shard Acceptance. 7000 TPR with 95% Process Shard Acceptance.
Adaptive Shard Size		No	No	No	No	No	Yes	No

Issues and Future Works

Limiting Cross-Shard Transactions

Reducing Transaction Latency

Pruning Shard's Ledger

Data Validity Problem

Data Availability Problem

References

[1] Luu, Loi, et al. "A secure sharding protocol for open blockchains." Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016.

[2] Kokoris-Kogias, Eleftherios, et al. "Omniledger: A secure, scale-out, decentralized ledger via sharding." 2018 IEEE symposium on security and privacy (SP). IEEE, 2018.

[3] Zamani, Mahdi, Mahnush Movahedi, and Mariana Raykova. "Rapidchain: Scaling blockchain via full sharding." Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. 2018.

[4] Wang, Jiaping, and Hao Wang. "Monoxide: Scale out blockchains with asynchronous consensus zones." 16th USENIX symposium on networked systems design and implementation (NSDI 19). 2019.

[5] David, Bernardo, et al. "Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy." Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022.

[6] Xu, Yibin, et al. "A Two-Layer Blockchain Sharding Protocol Leveraging Safety and Liveness for Enhanced Performance." arXiv preprint arXiv:2310.11373 (2023).

Backup Slides

Omniledger [SP'2018]

Improves on the generation of randomness and proposes Atomix Protocol for Cross-Shard Transaction

❖ Better epoch randomness - Combination of RandHound and Algorand-based Verifiable Random Function (VRF)

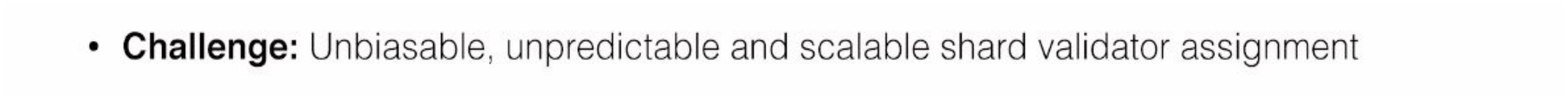
❖ Better consensus algorithm - ByzCoinX

❖ Atomix Protocol for efficient cross shard transaction.

❖ Tolerates up to $f < n/4$ adaptive byzantine adversaries.

Omniledger

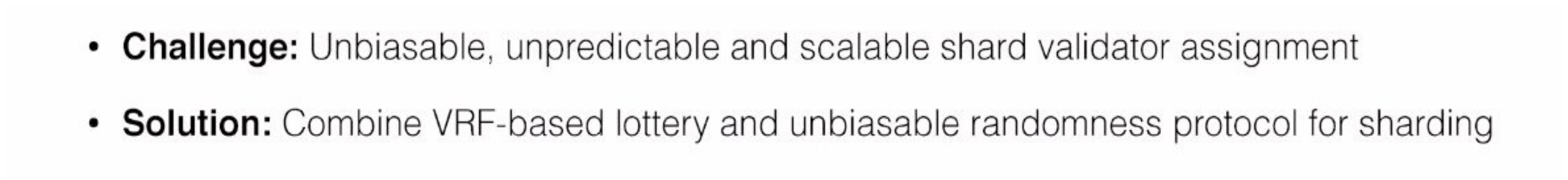
Epoch Randomness and Shard Validator Assignment



- **Challenge:** Unbiasable, unpredictable and scalable shard validator assignment

Omniledger

Epoch Randomness and Shard Validator Assignment

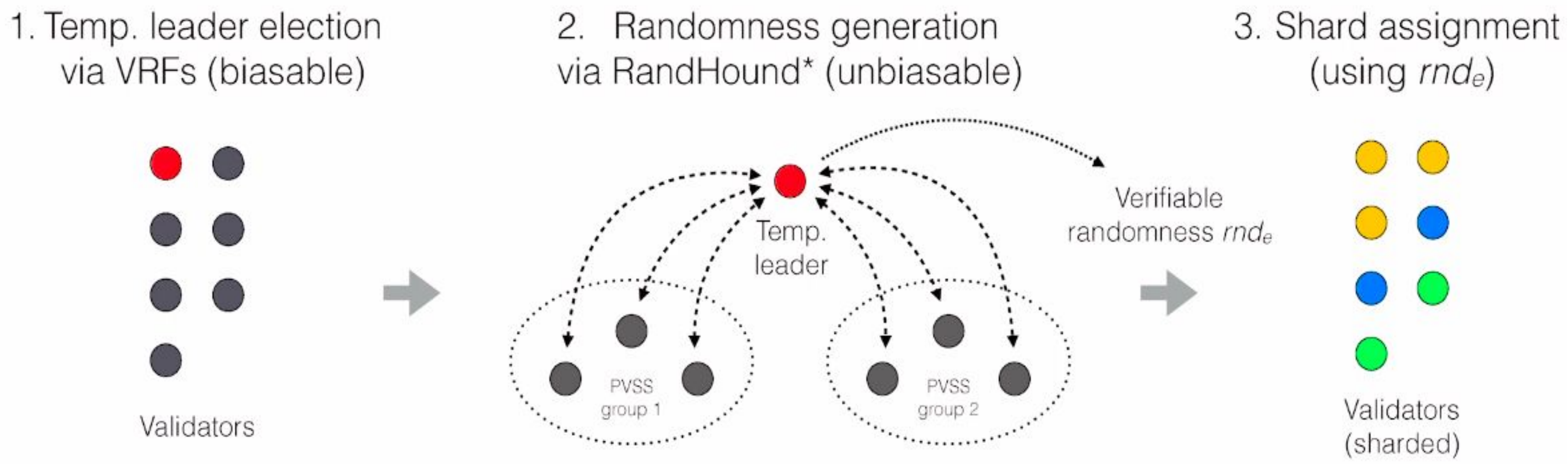


- **Challenge:** Unbiasable, unpredictable and scalable shard validator assignment
- **Solution:** Combine VRF-based lottery and unbiasable randomness protocol for sharding

Omniledger

Shard Validator Assignment

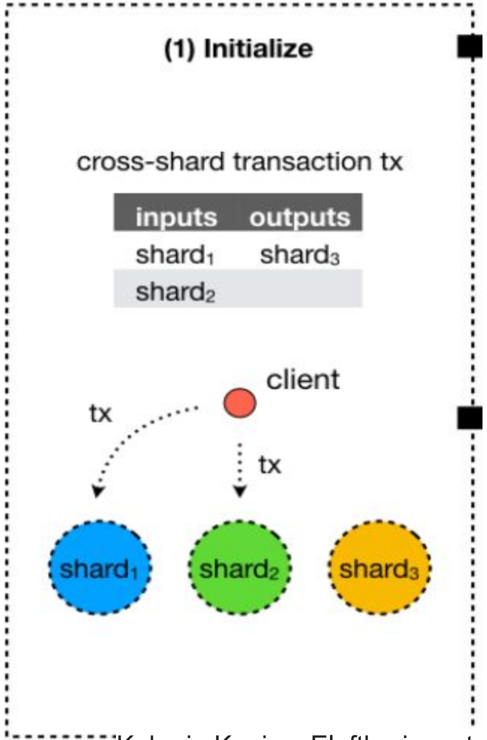
- **Challenge:** Unbiasable, unpredictable and scalable shard validator assignment
- **Solution:** Combine VRF-based lottery and unbiasable randomness protocol for sharding


1. Temp. leader election via VRFs (biasable)
2. Randomness generation via RandHound* (unbiasable)
3. Shard assignment (using rnd_e)
Validators
Temp. leader
PVSS group 1
PVSS group 2
Verifiable randomness rnd_e
Validators (sharded)

*Scalable Bias-resistant Distributed Randomness, E. Syta et al., IEEE S&P'17

Omniledger

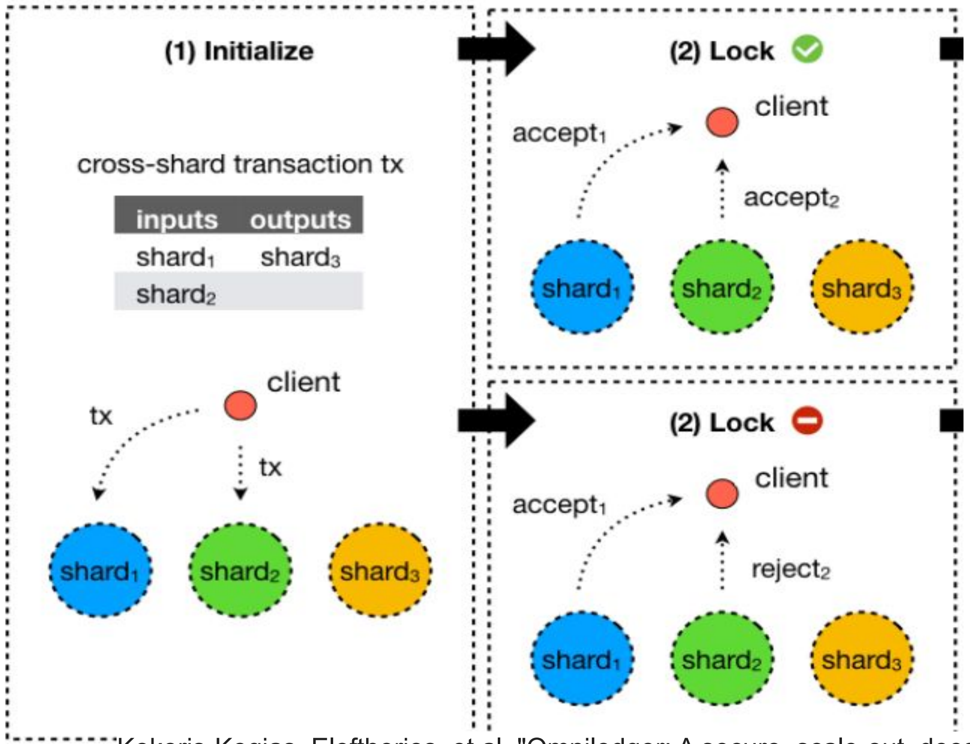
Atomix - Cross Shard Transaction



Kokoris-Kogias, Eleftherios, et al. "Omniledger: A secure, scale-out, decentralized ledger via sharding."

Omniledger

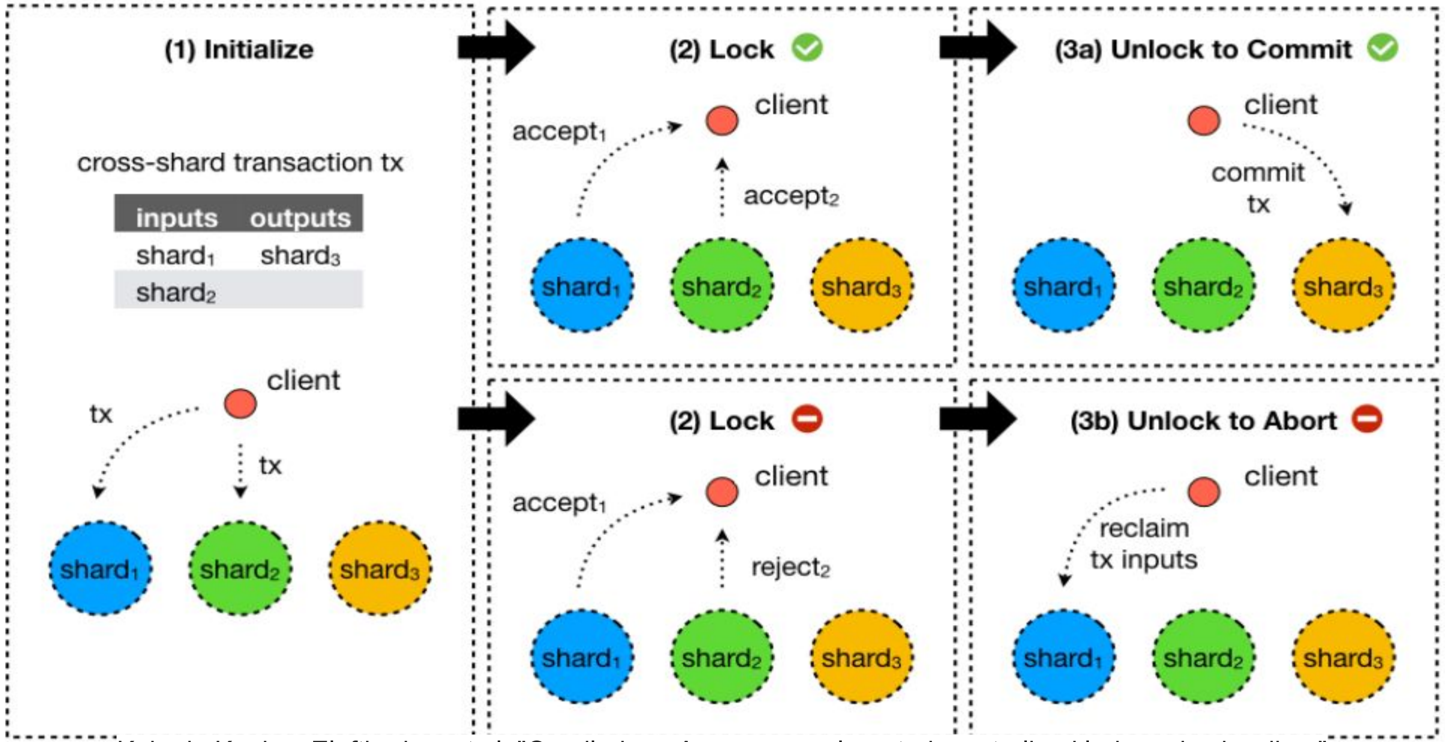
Atomix - Cross Shard Transaction



Kokoris-Kogias, Eleftherios, et al. "Omniledger: A secure, scale-out, decentralized ledger via sharding."

Omniledger

Atomix - Cross Shard Transaction



Kokoris-Kogias, Eleftherios, et al. "Omniledger: A secure, scale-out, decentralized ledger via sharding."

Omniledger

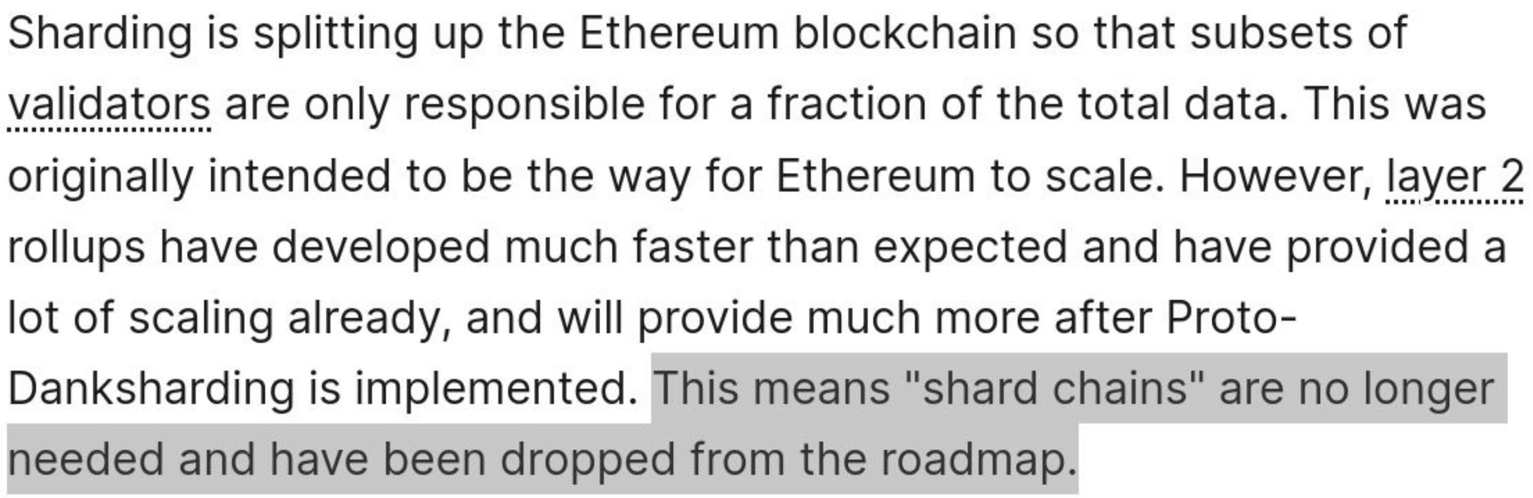
Limitations

➔ The combination of RandHound and VRF suffers from the reliance on a third-party initial randomness pre-defined in the genesis block.

➔ Atomix Protocol requires a client to actively participate in cross-shard transactions.

➔ This client-driven protocol suffers from indefinite blocking if the client is malicious. If Payee is a client that coordinates a transaction that transfers funds from a payer’s account. A malicious payee may pretend to crash indefinitely during the lock/unlock protocol, hence, the payer’s funds are locked forever.

Ethereum 2.0



Sharding is splitting up the Ethereum blockchain so that subsets of validators are only responsible for a fraction of the total data. This was originally intended to be the way for Ethereum to scale. However, layer 2 rollups have developed much faster than expected and have provided a lot of scaling already, and will provide much more after Proto-Danksharding is implemented. This means "shard chains" are no longer needed and have been dropped from the roadmap.

Ethereum 2.0

Dank-Sharding

- Rollups are a way to scale Ethereum by batching transactions off-chain and then posting the results to Ethereum.

- A Rollup is essentially composed of two parts: Data and Execution check.

- The data is the full sequence of transactions that is being processed by a rollup.

- The execution check is the re-execution of those transactions by some honest actor (a "prover") to ensure that the proposed state change is correct

Ethereum 2.0

Dank-Sharding

- To perform the execution check, the transaction data has to be available for long enough for anyone to download and check.

- Rollups uses CALLDATA for posting transactions, which proves costly as it is processed by all Ethereum nodes and permanently stored on-chain.

- Danksharding introduces Data Blobs. Rollups post commitments to their transaction data on-chain and also make the actual data available in these data blobs.

Ethereum 2.0

Dank-Sharding

- The data in these blobs is not accessible to the EVM and is automatically deleted after a fixed time period.

- At the node-level, the blobs of data are held in the consensus client. The consensus clients attest that they have seen the data and that it has been propagated around the network.

- Validators will have enough time to check the commitments and challenge data of a rollup transaction while the consensus client can provide proof for it.