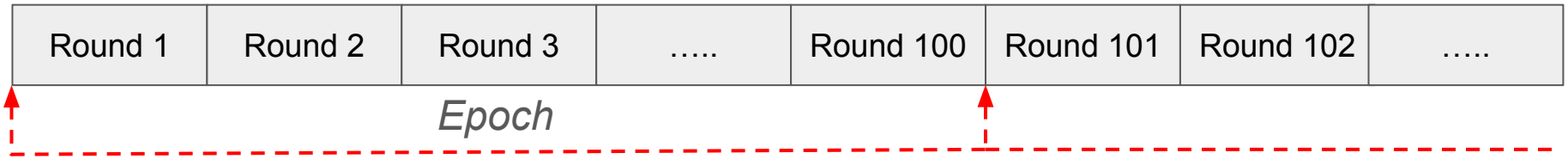
Performance Analysis of a Live Sharded Blockchain System

Debdoot Roy Chowdhury (23M0675)
Guided By: Prof. Vinay J. Ribeiro

What is MultiVersX?

- MultiversX, previously known as Elrond, founded in 2019 by Benjamin Mincu.
- It is a distributed blockchain network for web3 applications that utilizes a sharded state architecture and secure Proof of Stake consensus mechanism.
- Their main network has over 3000 active validators and over 400M confirmed transactions.
- They provide efficient ecosystem to create dApps within 30 minutes, and has partnered with many popular Google Cloud, Tencent, AWS, Digital Ocean, Coinbase e.t.c.

Terminologies and Architecture



➔ Time is divided into fixed time slots of Epochs and Rounds.

➔ Each Epoch consist of multiple rounds of 6 Seconds.

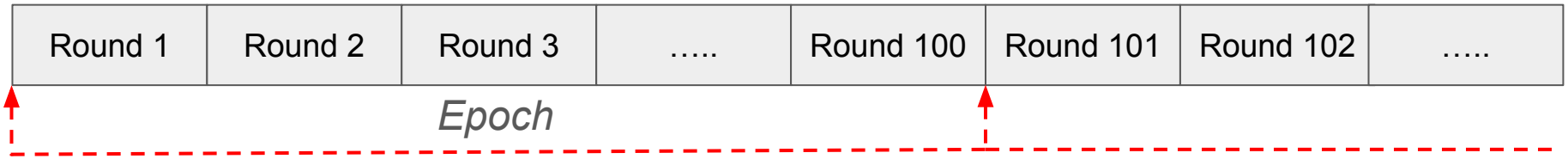
Terminologies and Architecture

➔ Time is divided into fixed time slots of Epochs and Rounds.

➔ Each Epoch consist of multiple rounds of 6 Seconds.

➔ Each Shard produces at most 1 Block in a Round.

Terminologies and Architecture



➔ Time is divided into fixed time slots of Epochs and Rounds.

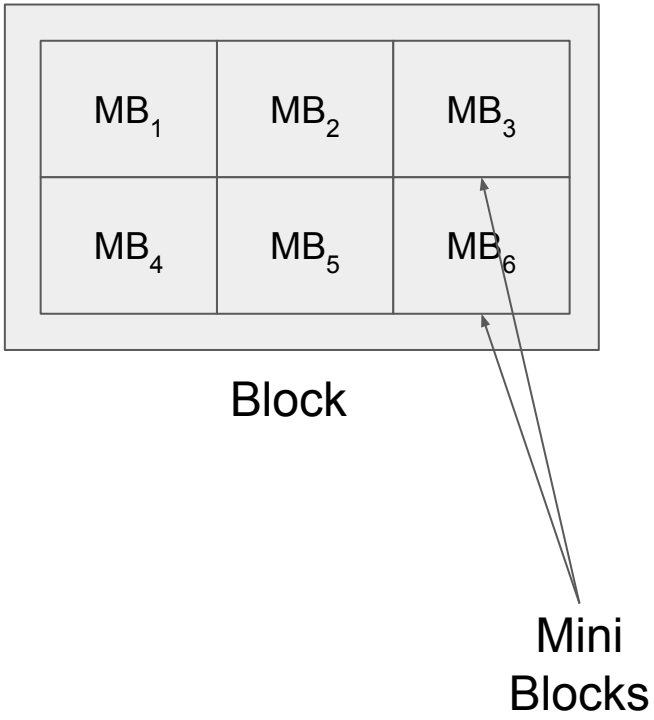
➔ Each Epoch consist of multiple rounds of 6 Seconds.

➔ Each Shard produces at most 1 Block in a Round.

➔ 100 Rounds compiles to 1 Epoch.

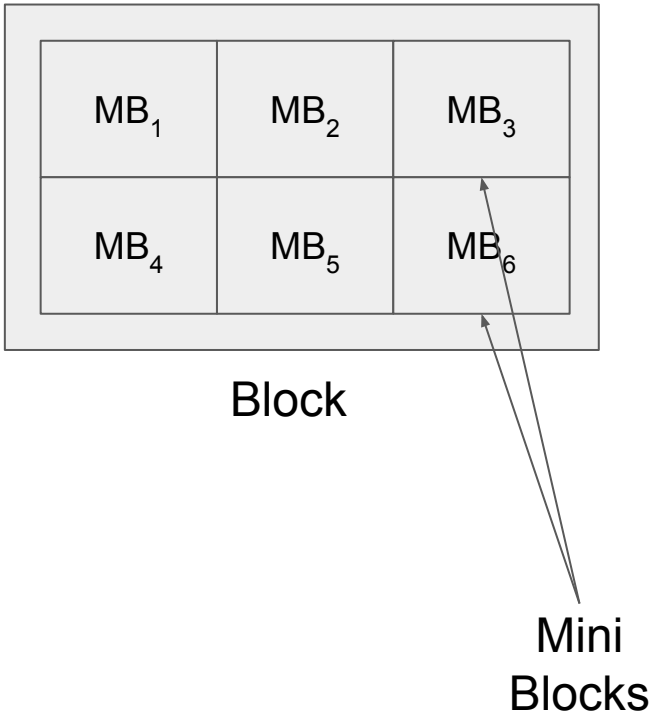
➔ After each Epoch, shard reorganization and pruning takes place.

Terminologies and Architecture



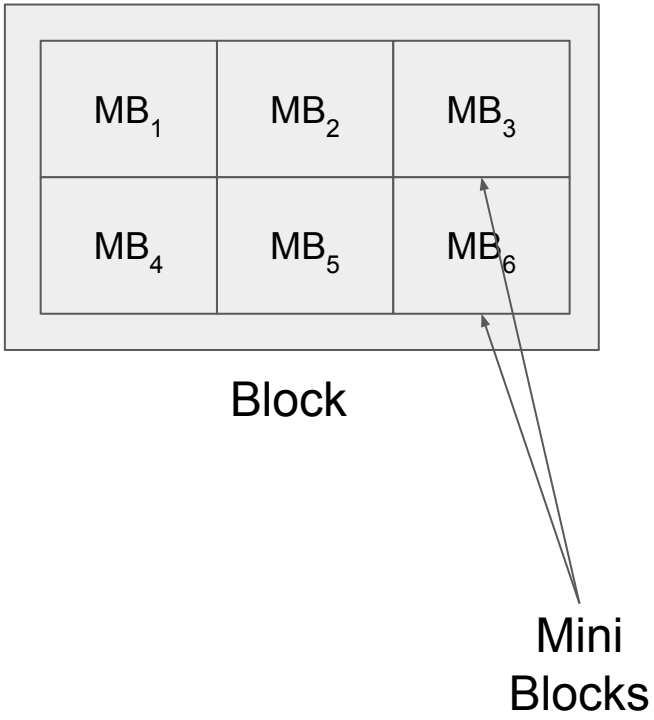
➔ Each Block consists of several Mini-Blocks.

Terminologies and Architecture



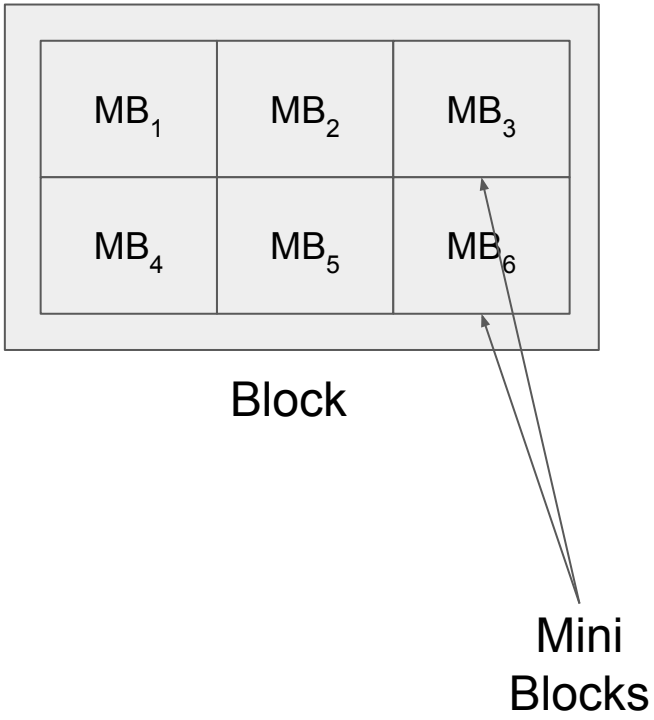
- ➔ Each Block consists of several Mini-Blocks.
- ➔ Each Mini-Block consists of similar type of Transactions (Txns to a specific shard is stored in one MiniBlock).

Terminologies and Architecture



- ➔ Each Block consists of several Mini-Blocks.
- ➔ Each Mini-Block consists of similar type of Transactions (Txns to a specific shard is stored in one MiniBlock).
- ➔ Each Mini-Block can hold at most 5000 transactions. And each block can hold at most 20000 transactions.

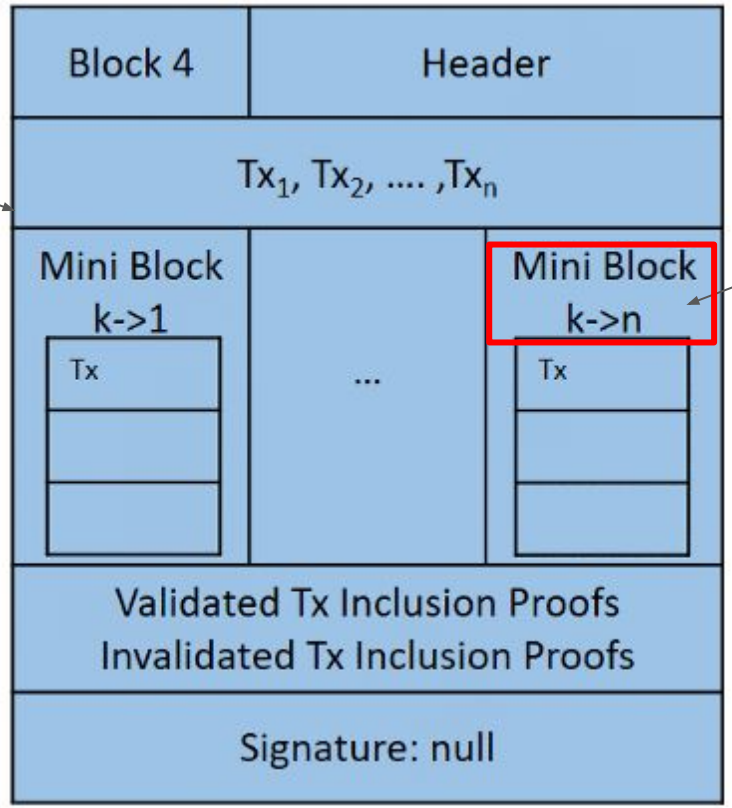
Terminologies and Architecture



- ➔ Each Block consists of several Mini-Blocks.
- ➔ Each Mini-Block consists of similar type of Transactions (Txns to a specific shard is stored in one MiniBlock).
- ➔ Each Mini-Block can hold at most 5000 transactions. And each block can hold at most 20000 transactions.
- ➔ It is useful to segregate different transactions into different mini-blocks and speed up fetching and storing.

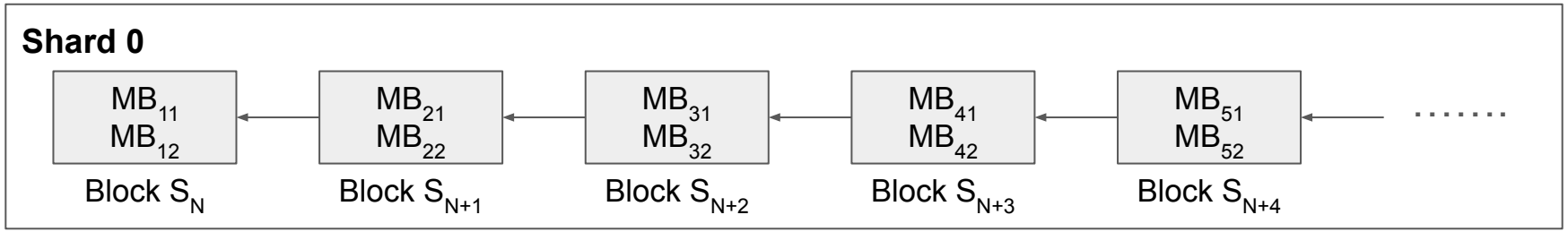
Terminologies and Architecture

Belongs to Shard k

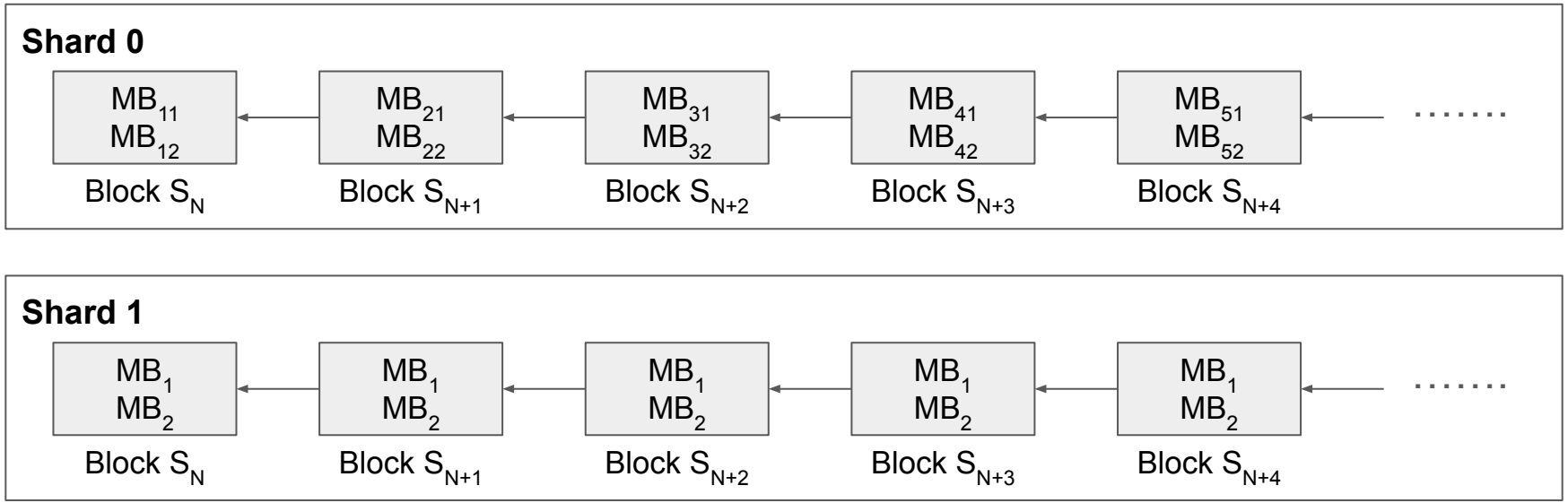


Mini-Block consisting of transactions from Shard k to n

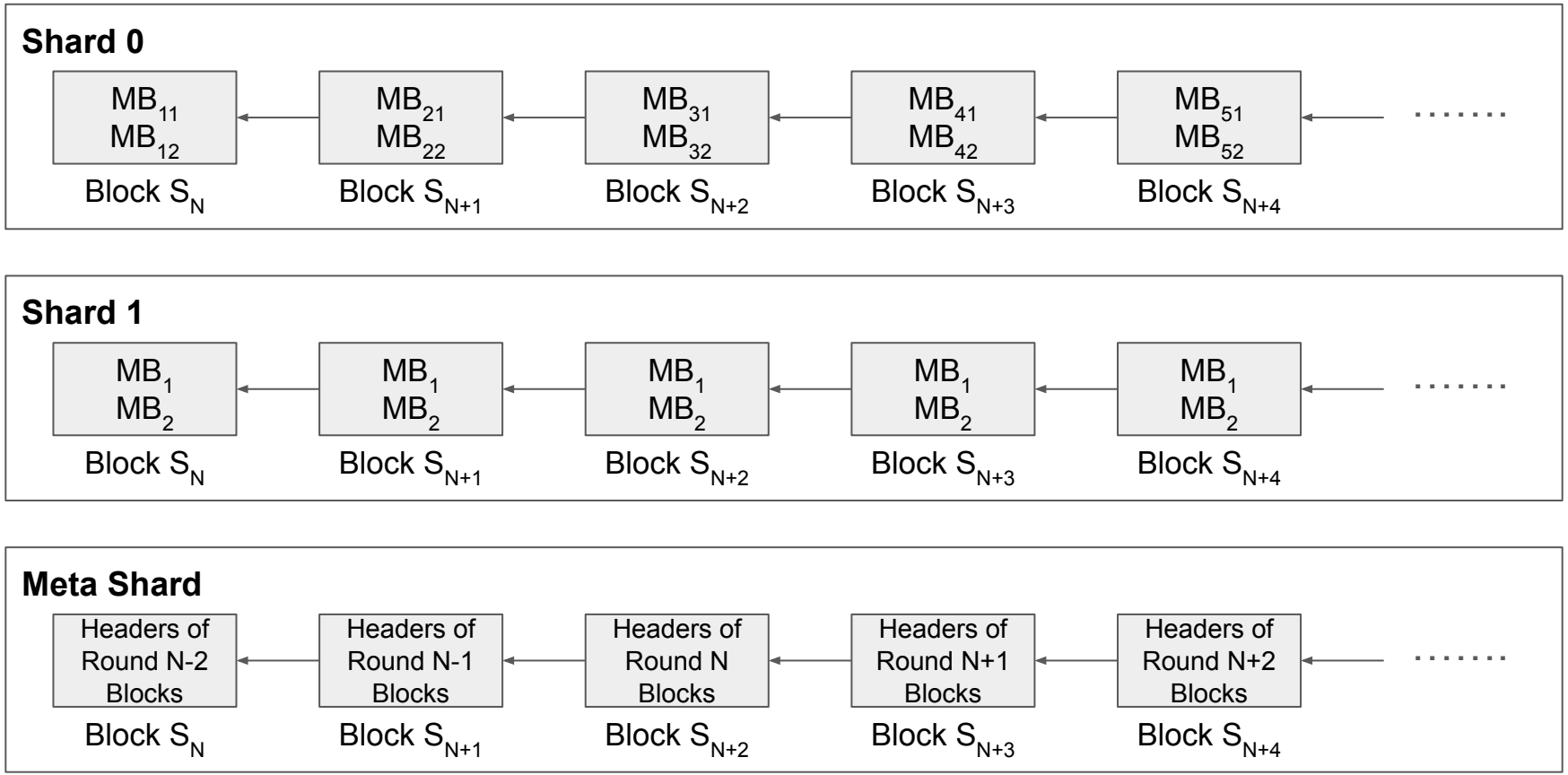
Terminologies and Architecture



Terminologies and Architecture



Terminologies and Architecture



Terminologies and Architecture

➔ “Secure” Proof of Stake is used for Consensus.

Terminologies and Architecture

- ➔ “Secure” Proof of Stake is used for Consensus.
- ➔ Adaptive State Sharding technique is followed to dynamically change the number of shards.

Terminologies and Architecture

- ➔ “Secure” Proof of Stake is used for Consensus.
- ➔ Adaptive State Sharding technique is followed to dynamically change the number of shards.
- ➔ Follows Account-Based transaction model.

Terminologies and Architecture

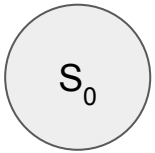
➔ “Secure” Proof of Stake is used for Consensus.

➔ Adaptive State Sharding technique is followed to dynamically change the number of shards.

➔ Follows Account-Based transaction model.

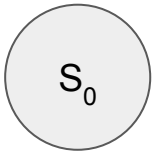
➔ Each shard process transactions of a distinct set of accounts based on the “bech32” address provided to that account.

Adaptive State Sharding



Starts with
1 Shard

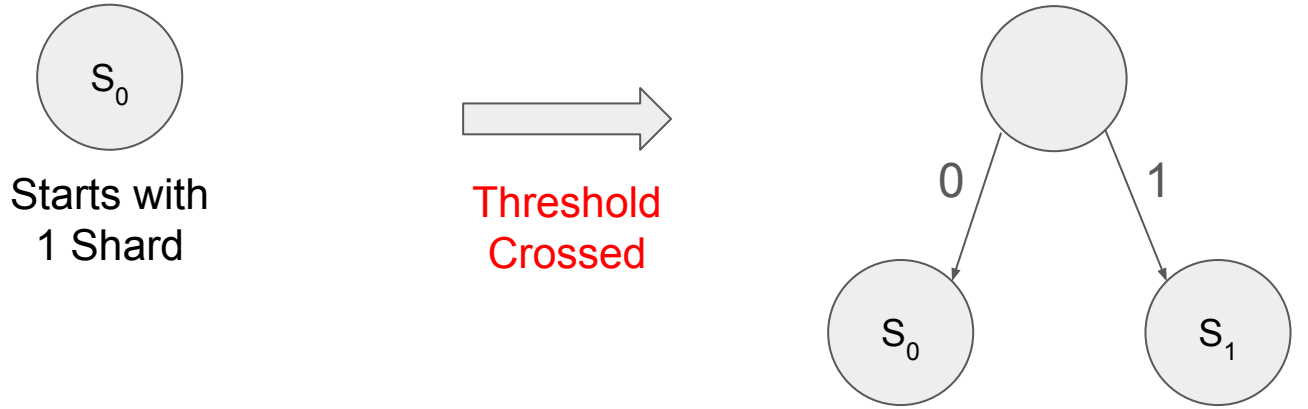
Adaptive State Sharding



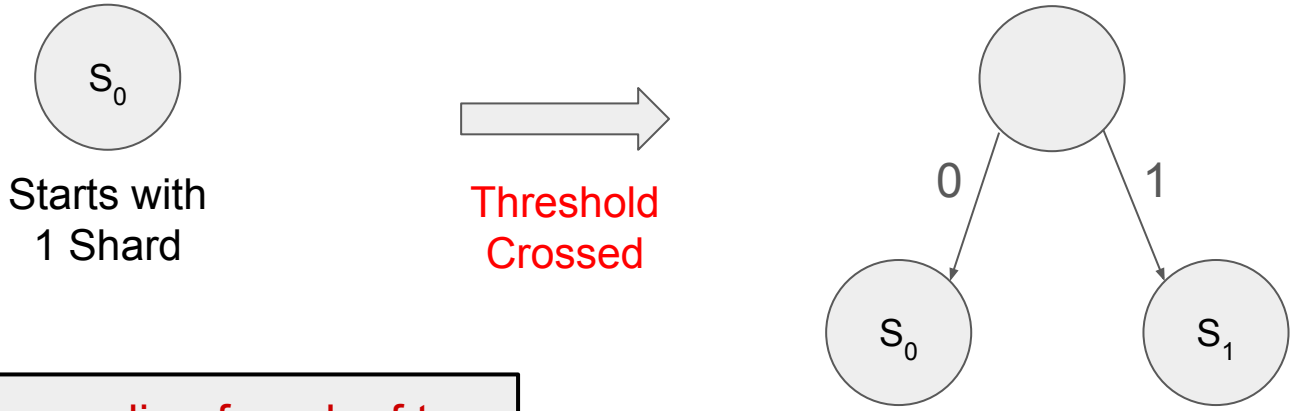
Starts with
1 Shard

Threshold
Crossed

Adaptive State Sharding

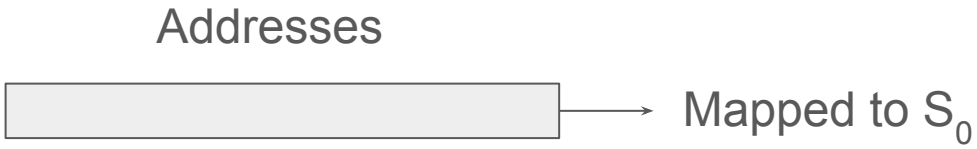


Adaptive State Sharding

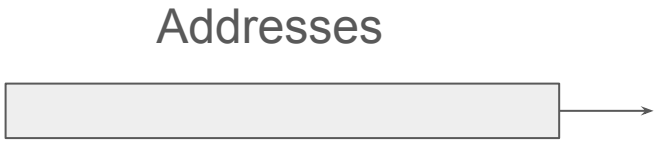


Branch encoding from leaf to root, represents the last n-bits of the account addresses that will have their originating transaction processed by that shard.

Adaptive State Sharding

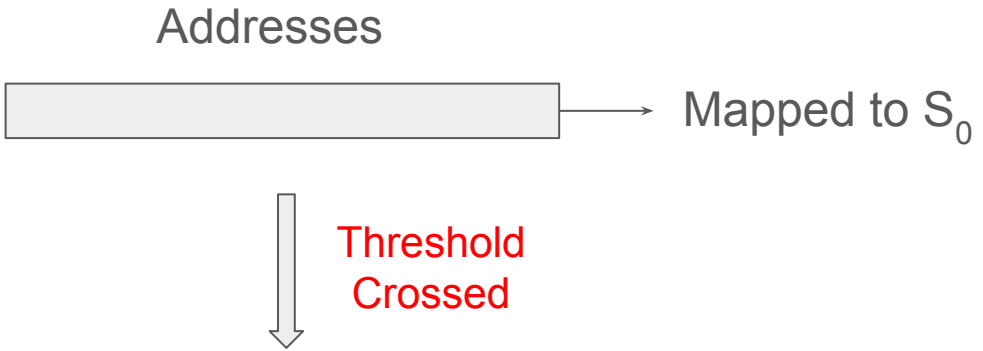


Adaptive State Sharding



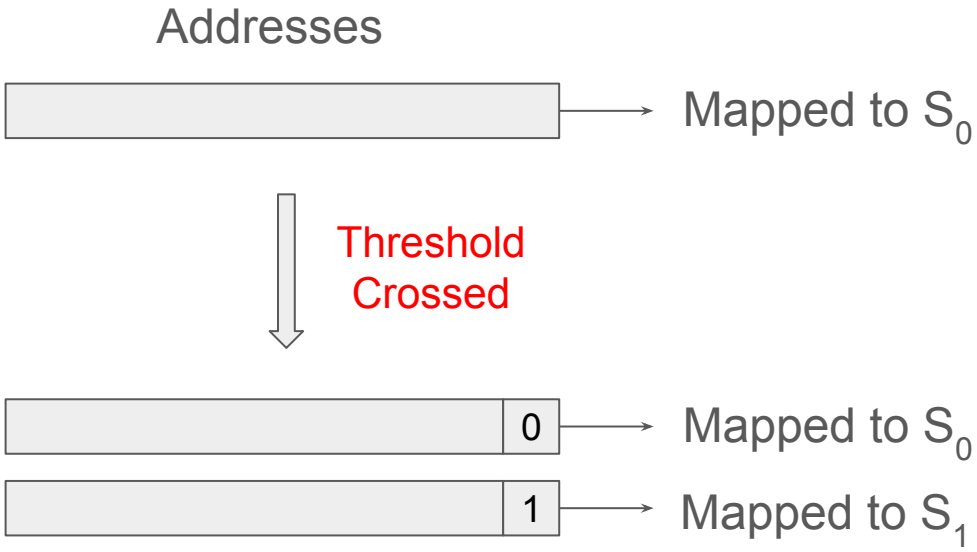
S_0 gets 100% of the entire transactions.

Adaptive State Sharding



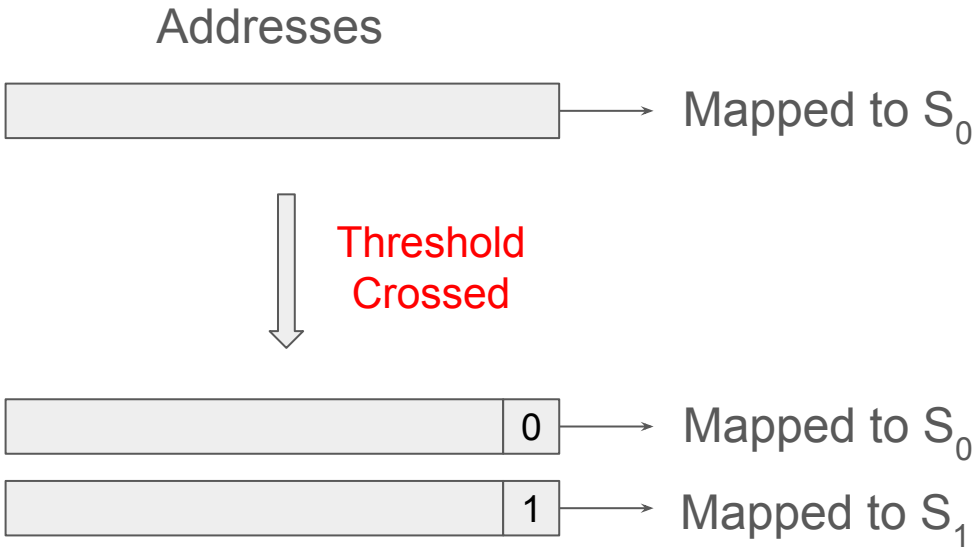
S_0 gets 100% of the entire transactions.

Adaptive State Sharding



S_0 gets 100% of the entire transactions.

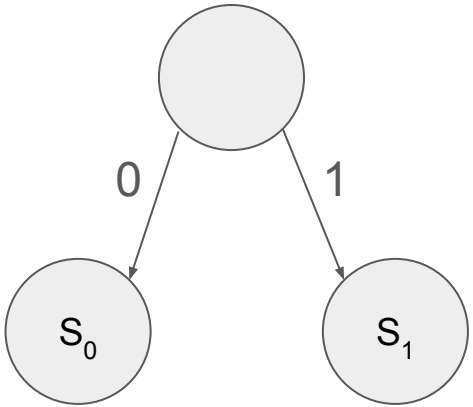
Adaptive State Sharding



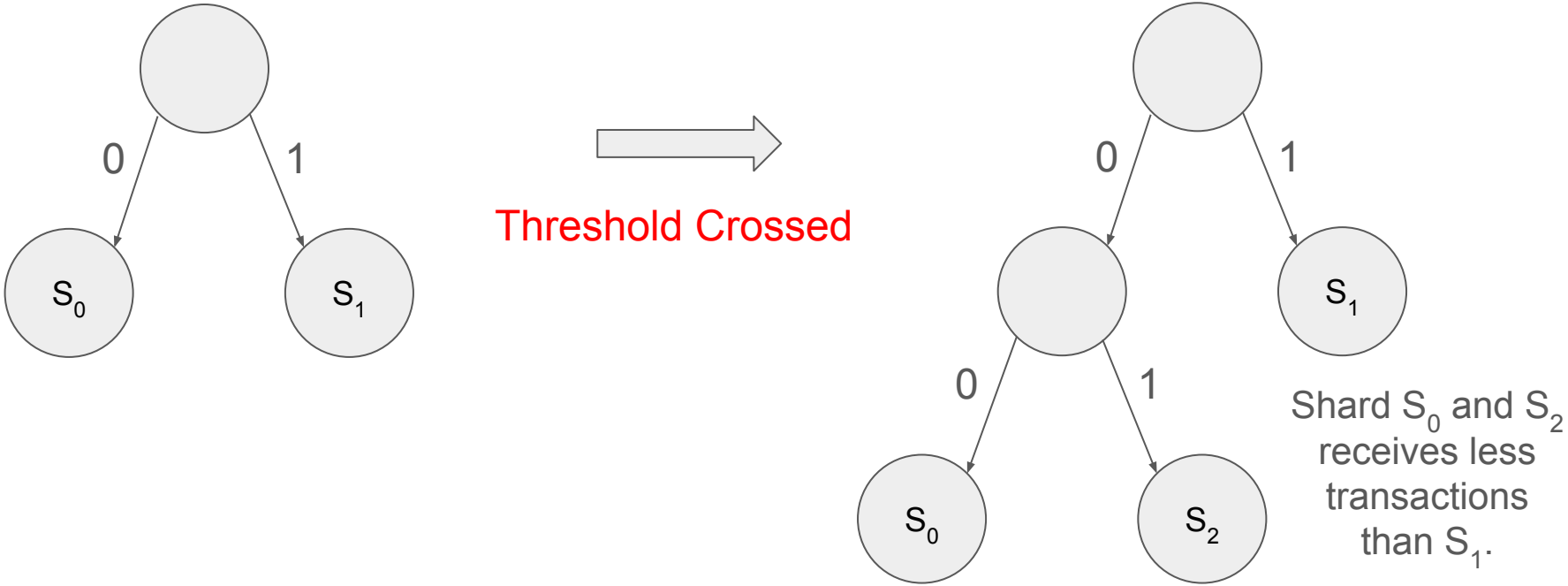
S_0 gets 100% of the entire transactions.

S_0 and S_1 each gets 50% of the entire transactions.

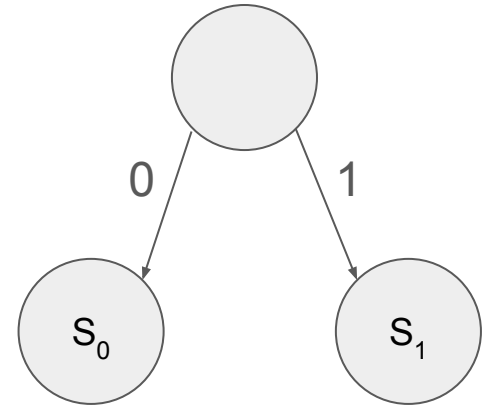
Adaptive State Sharding



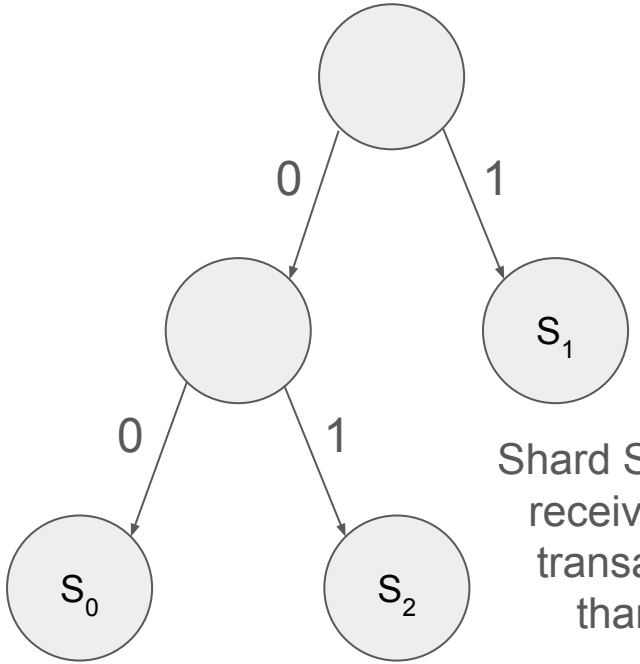
Adaptive State Sharding



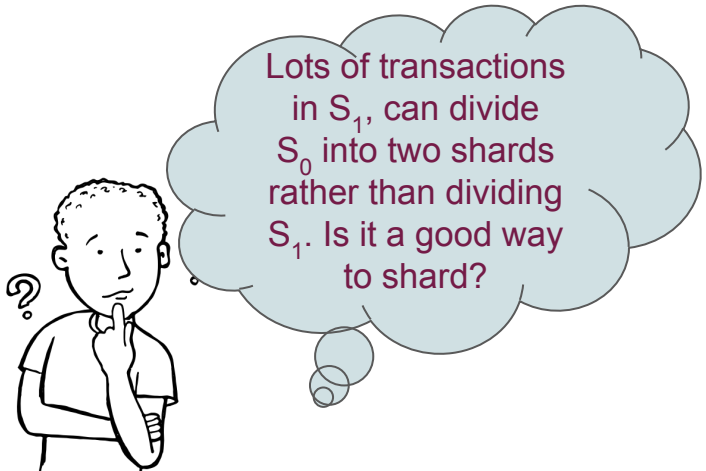
Adaptive State Sharding



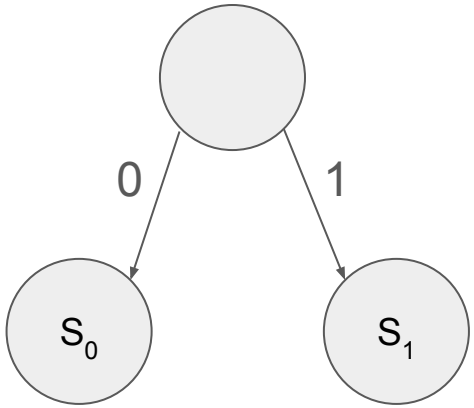
Threshold Crossed



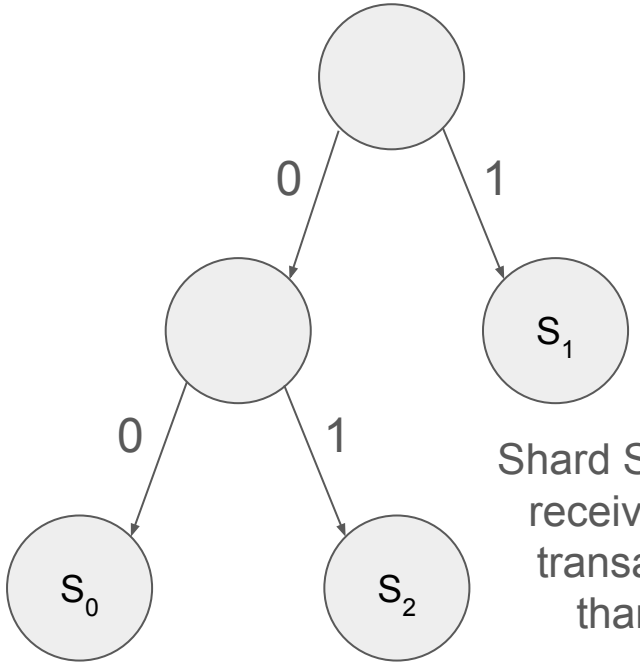
Shard S_0 and S_2 receives less transactions than S_1.



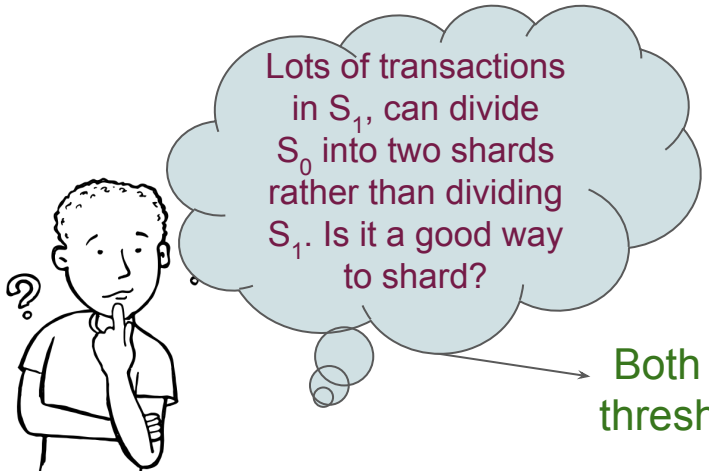
Adaptive State Sharding



Threshold Crossed

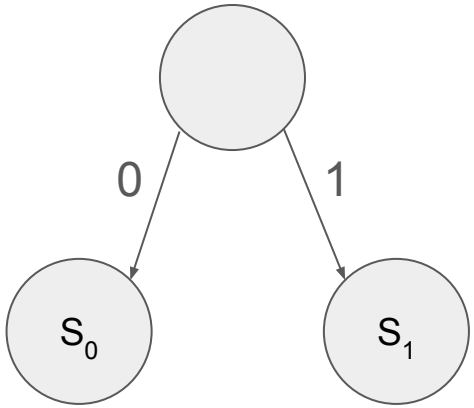


Shard S_0 and S_2 receives less transactions than S_1.

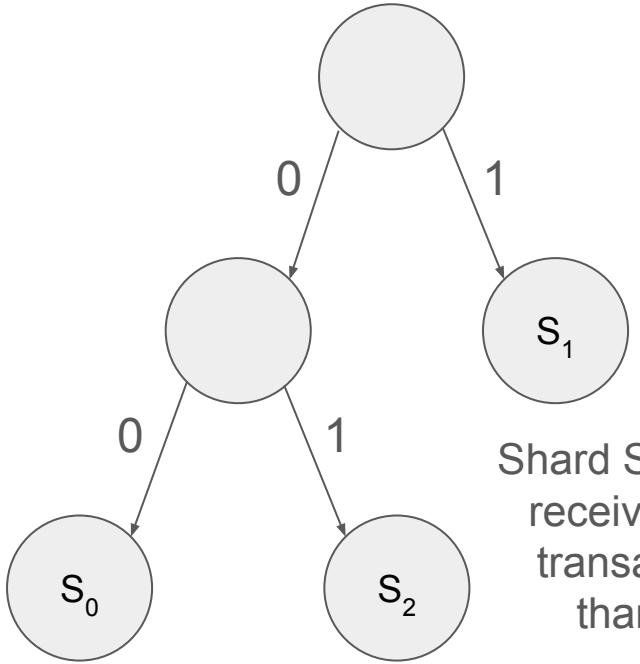


Both shards needs to reach the threshold before sharding occurs

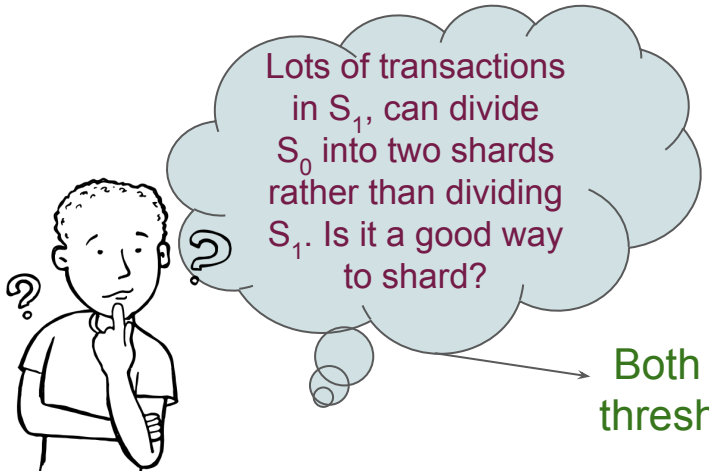
Adaptive State Sharding



Threshold Crossed

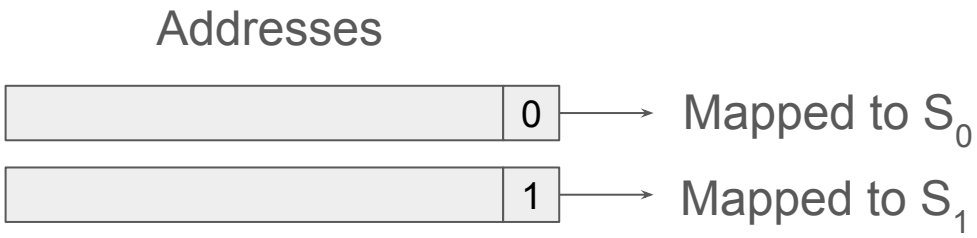


Shard S_0 and S_2 receives less transactions than S_1.

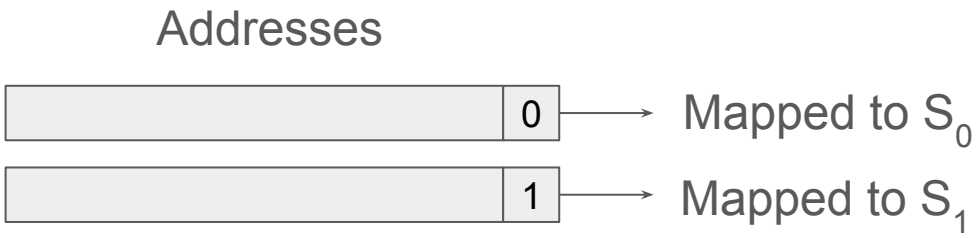


Both shards needs to reach the threshold before sharding occurs → S1 can get congested with network traffic, but still sharding will not happen

Adaptive State Sharding

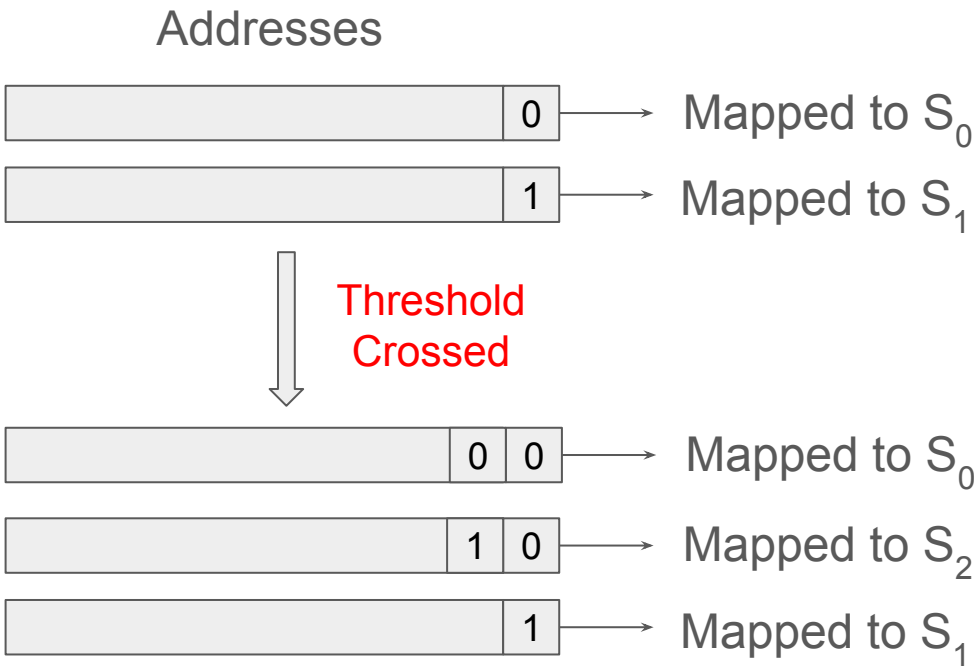


Adaptive State Sharding



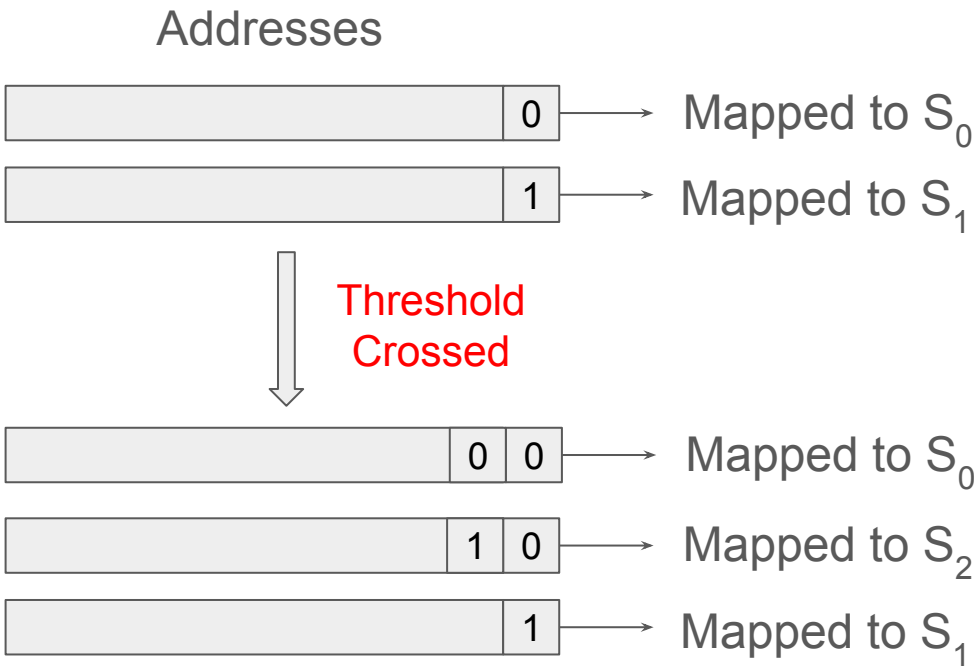
S_0 and S_1 each gets 50% of the entire transactions.

Adaptive State Sharding



S_0 and S_1 each gets 50% of the entire transactions.

Adaptive State Sharding



S_0 and S_1 each gets 50% of the entire transactions.

S_0 and S_2 each gets 25% of the entire transactions, while S_1 gets the rest 50%.

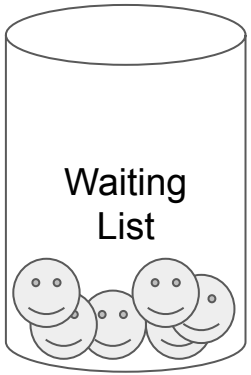
Secure Proof of Stake

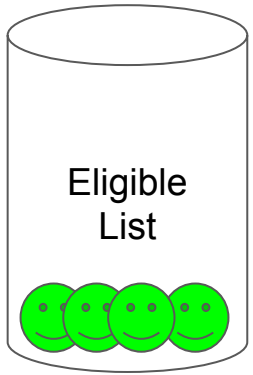
➔ Random Validators' Selection + Eligibility through Stake + Rating + Optimal Dimension for the Consensus Group.

Secure Proof of Stake

- ➔ Random Validators' Selection + Eligibility through Stake + Rating + Optimal Dimension for the Consensus Group.

- ➔ For each shard, there will be two lists of Nodes - Eligible List and Waiting List.

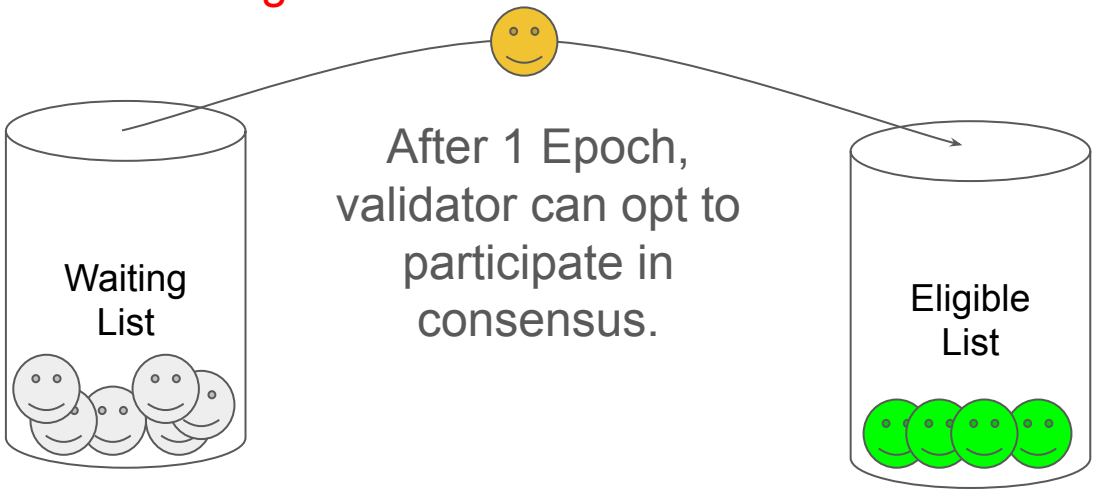




Secure Proof of Stake

- ➔ Random Validators' Selection + Eligibility through Stake + Rating + Optimal Dimension for the Consensus Group.

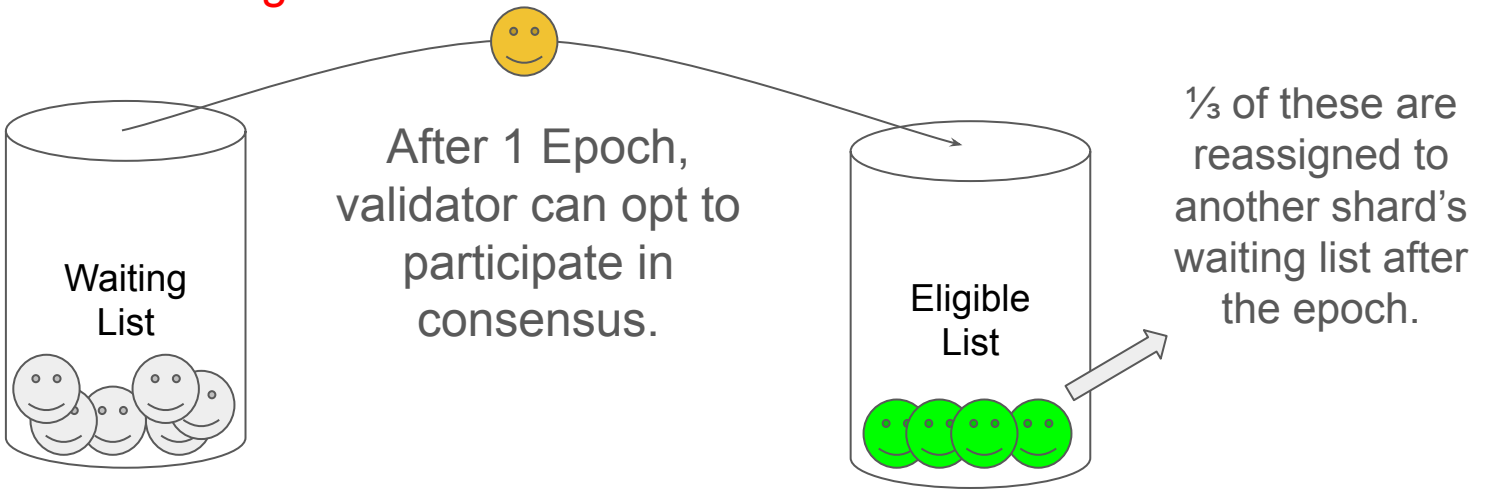
- ➔ For each shard, there will be two lists of Nodes - Eligible List and Waiting List.



Secure Proof of Stake

- ➔ Random Validators' Selection + Eligibility through Stake + Rating + Optimal Dimension for the Consensus Group.

- ➔ For each shard, there will be two lists of Nodes - Eligible List and Waiting List.

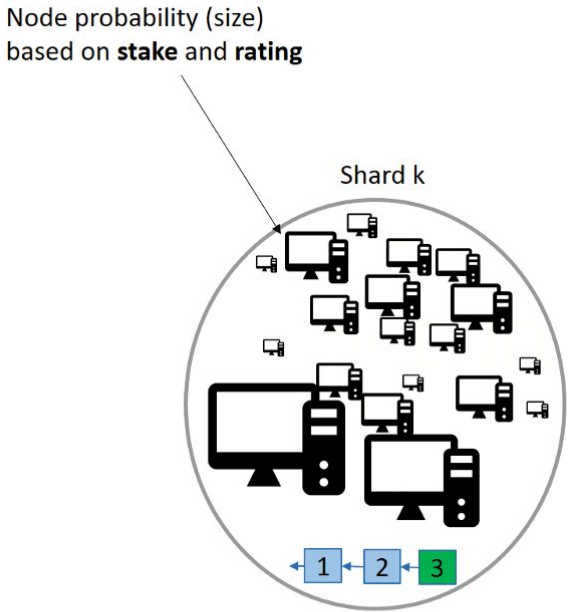


Secure Proof of Stake

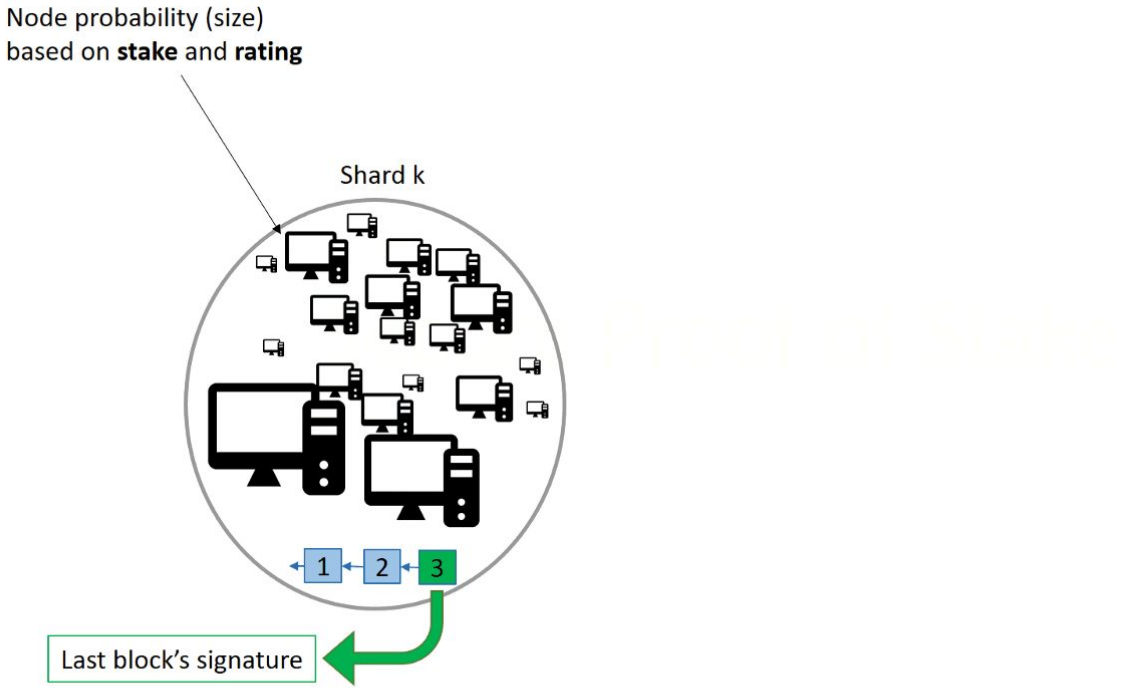
➔ How does the network supports such high throughput?

- Using small consensus groups to reduce communication overhead.
- Keeping transaction size less.
- Shard pruning keeps the nodes light.

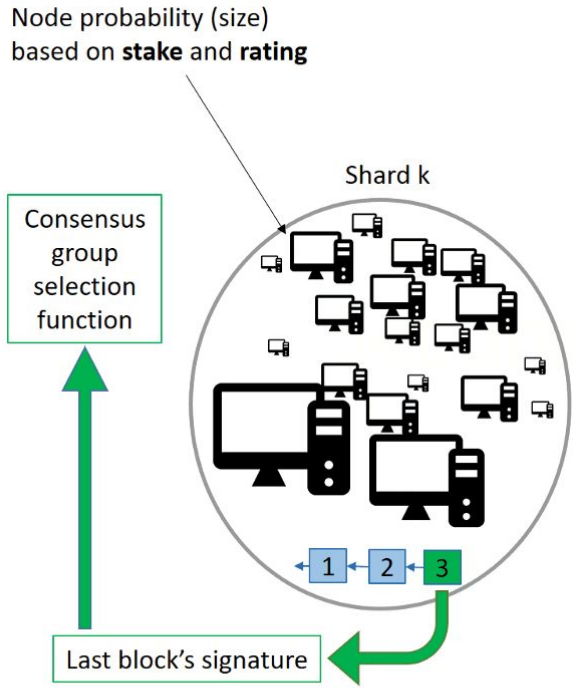
Secure Proof of Stake



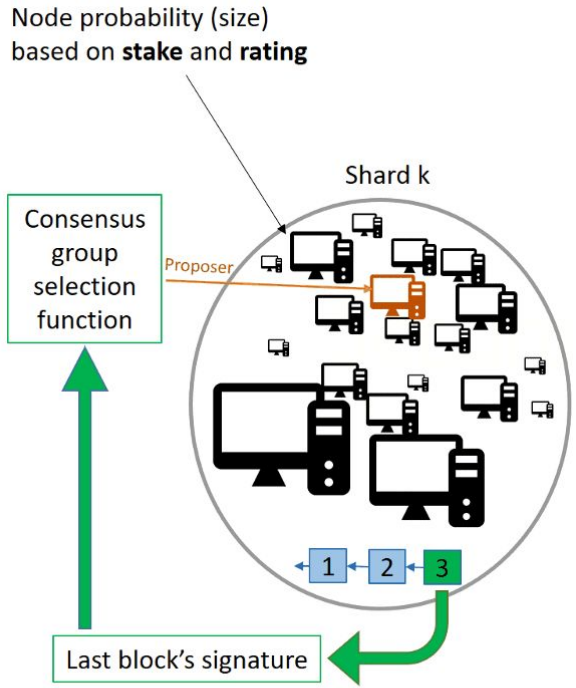
Secure Proof of Stake



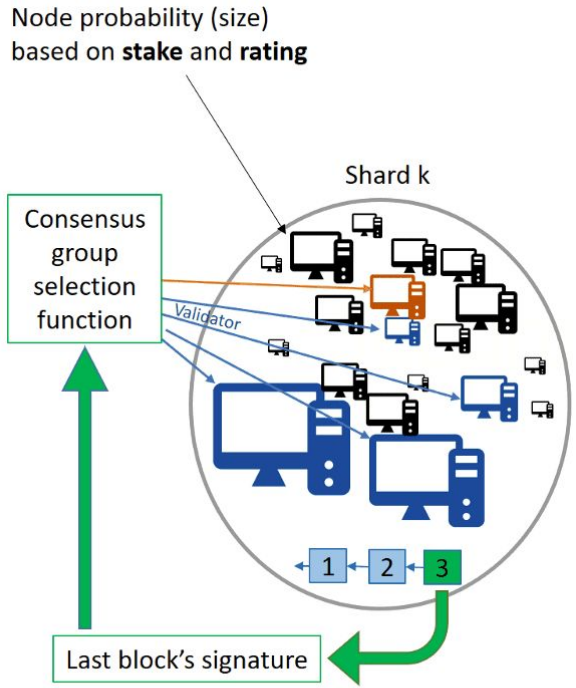
Secure Proof of Stake



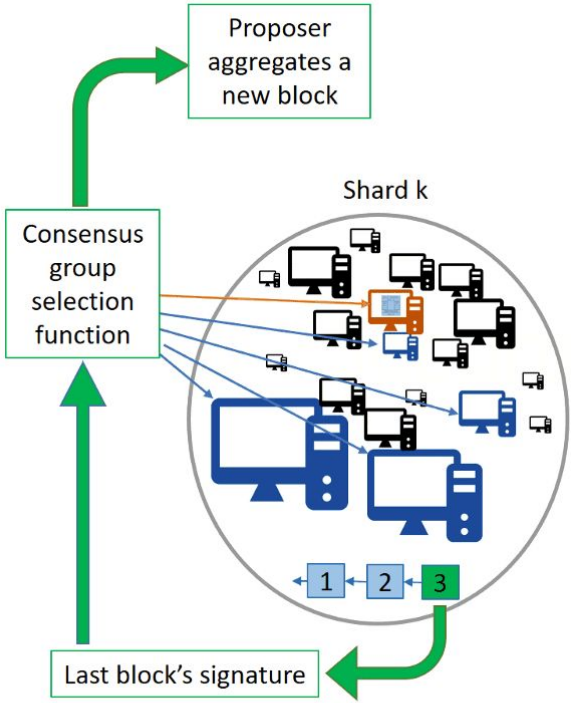
Secure Proof of Stake

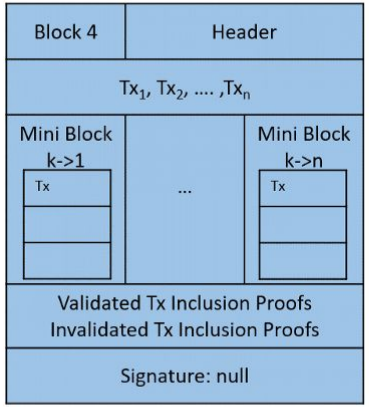


Secure Proof of Stake

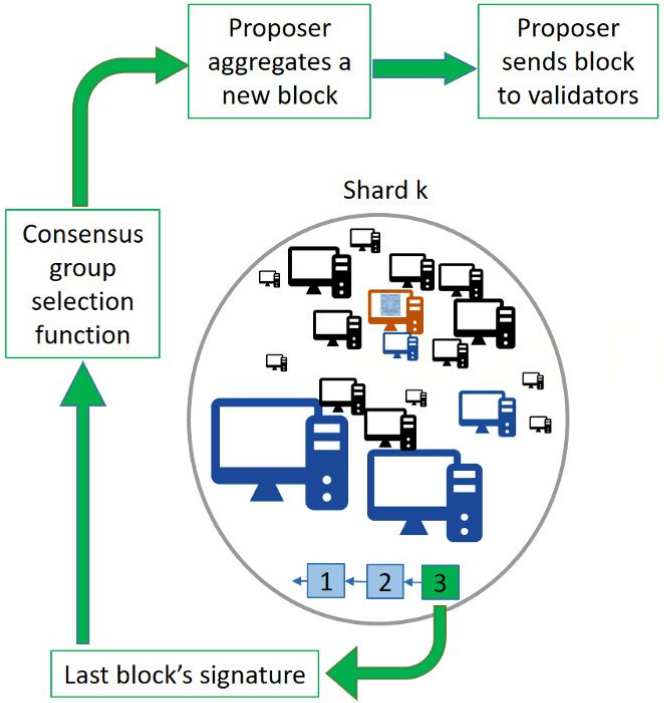


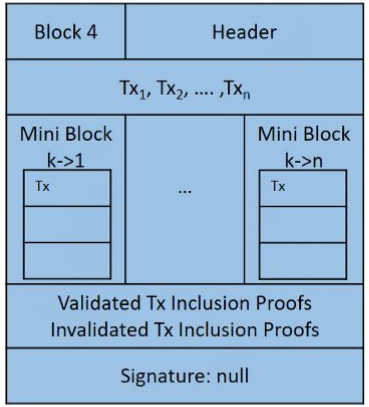
Secure Proof of Stake



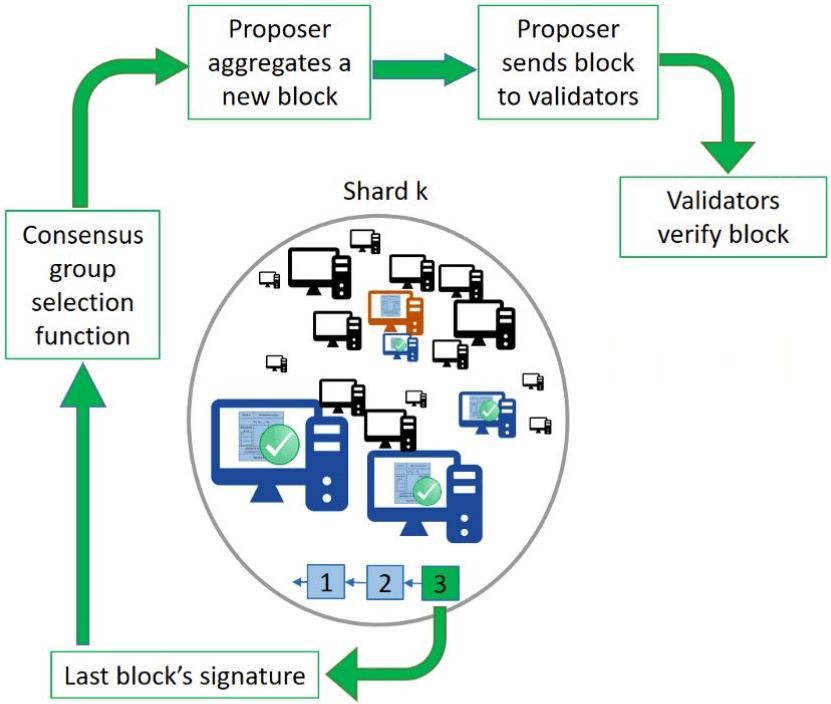


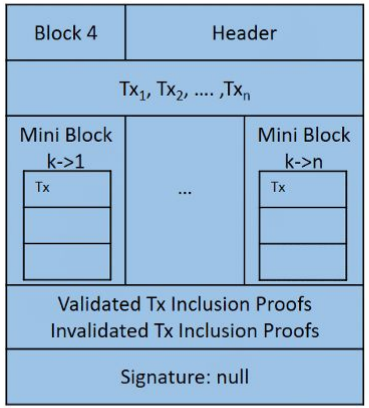
Secure Proof of Stake



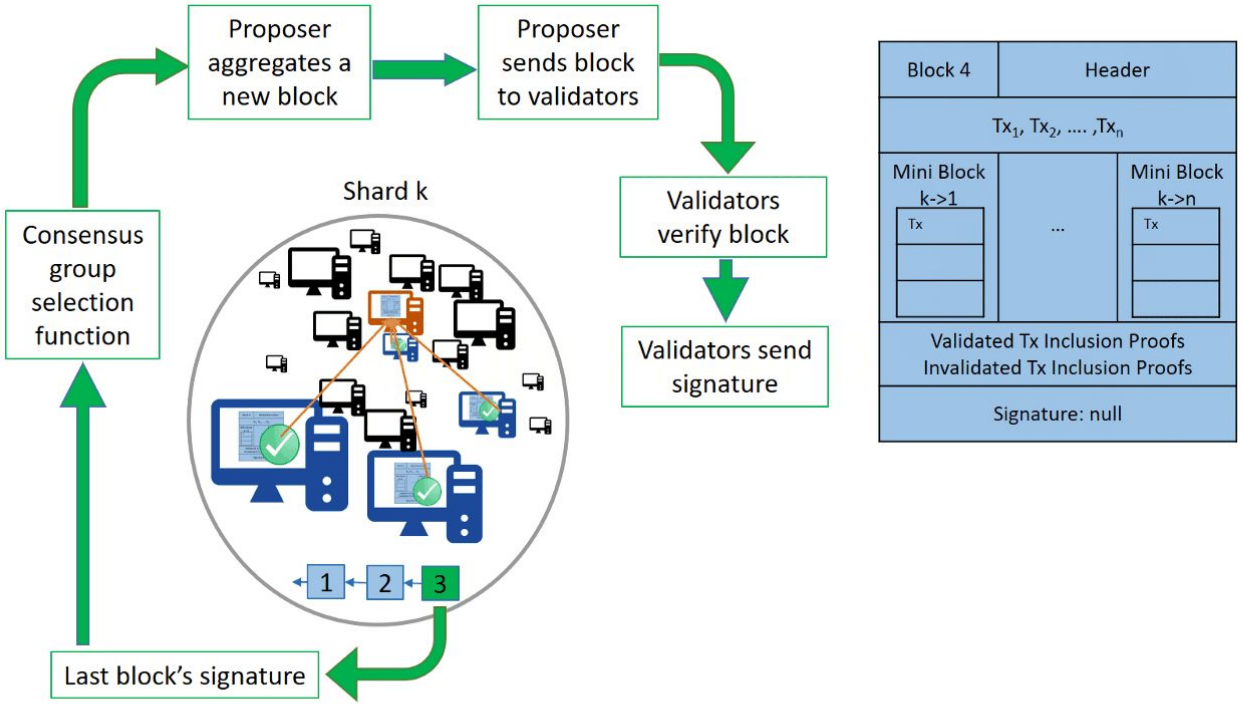


Secure Proof of Stake

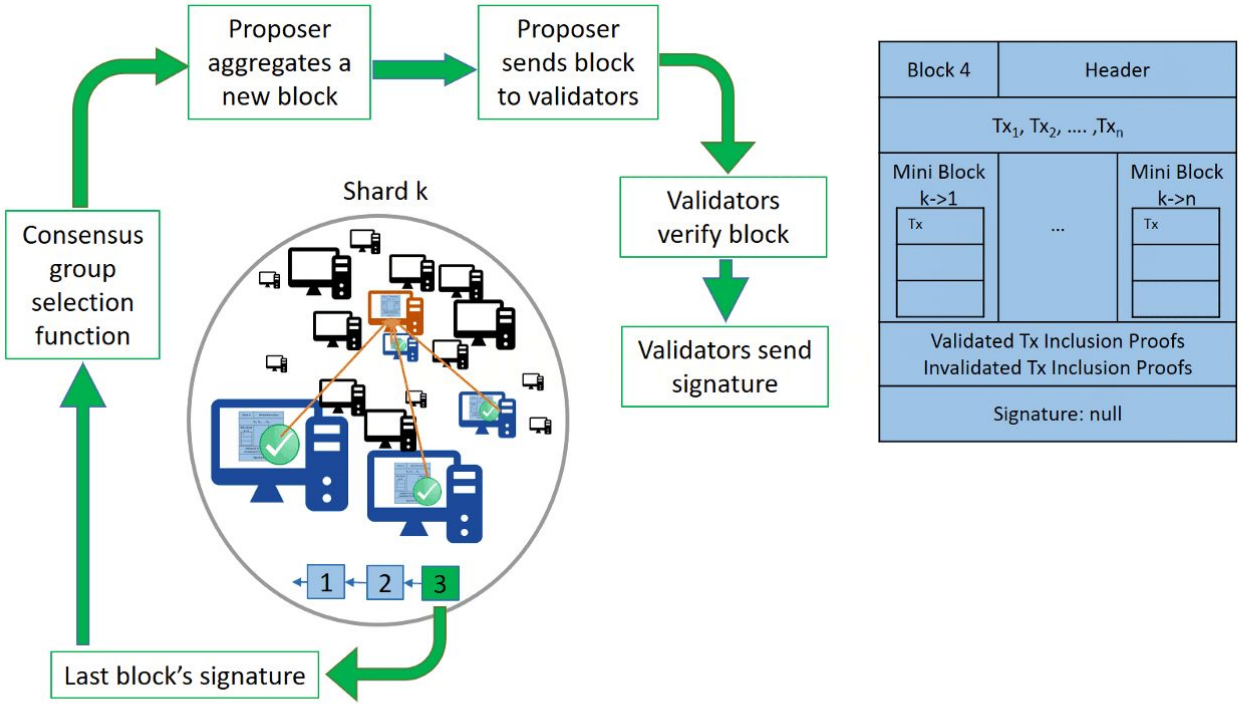




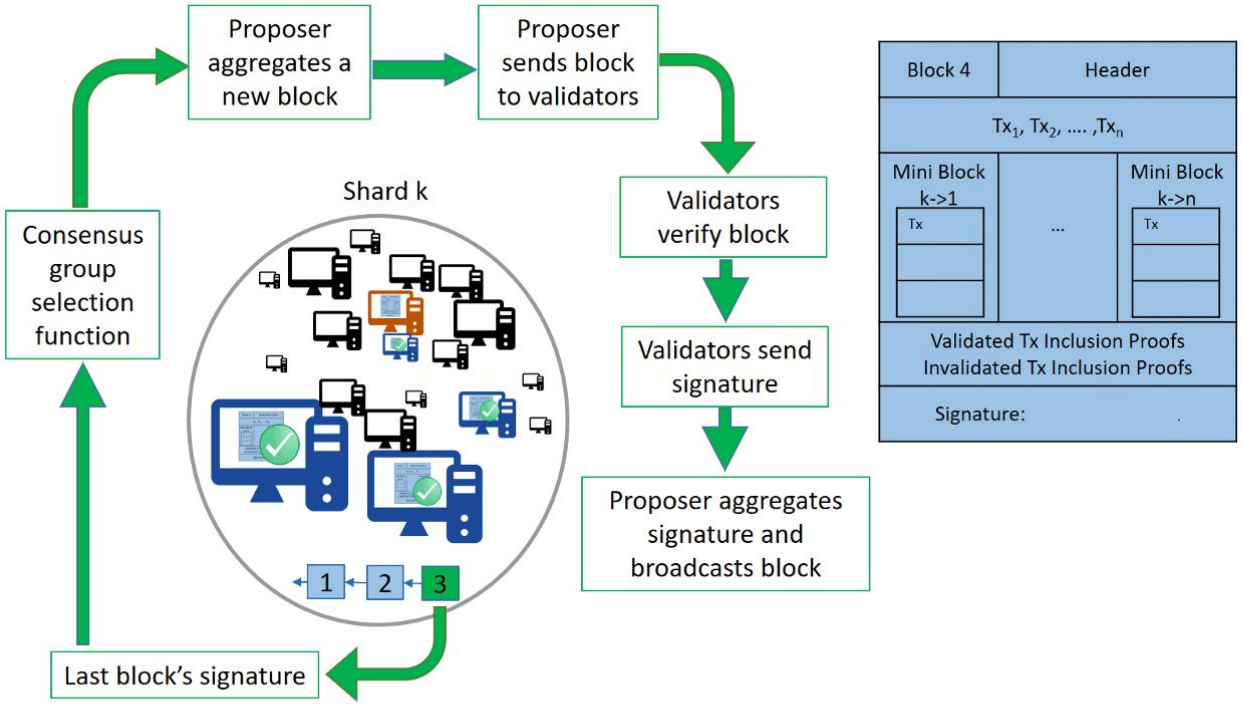
Secure Proof of Stake



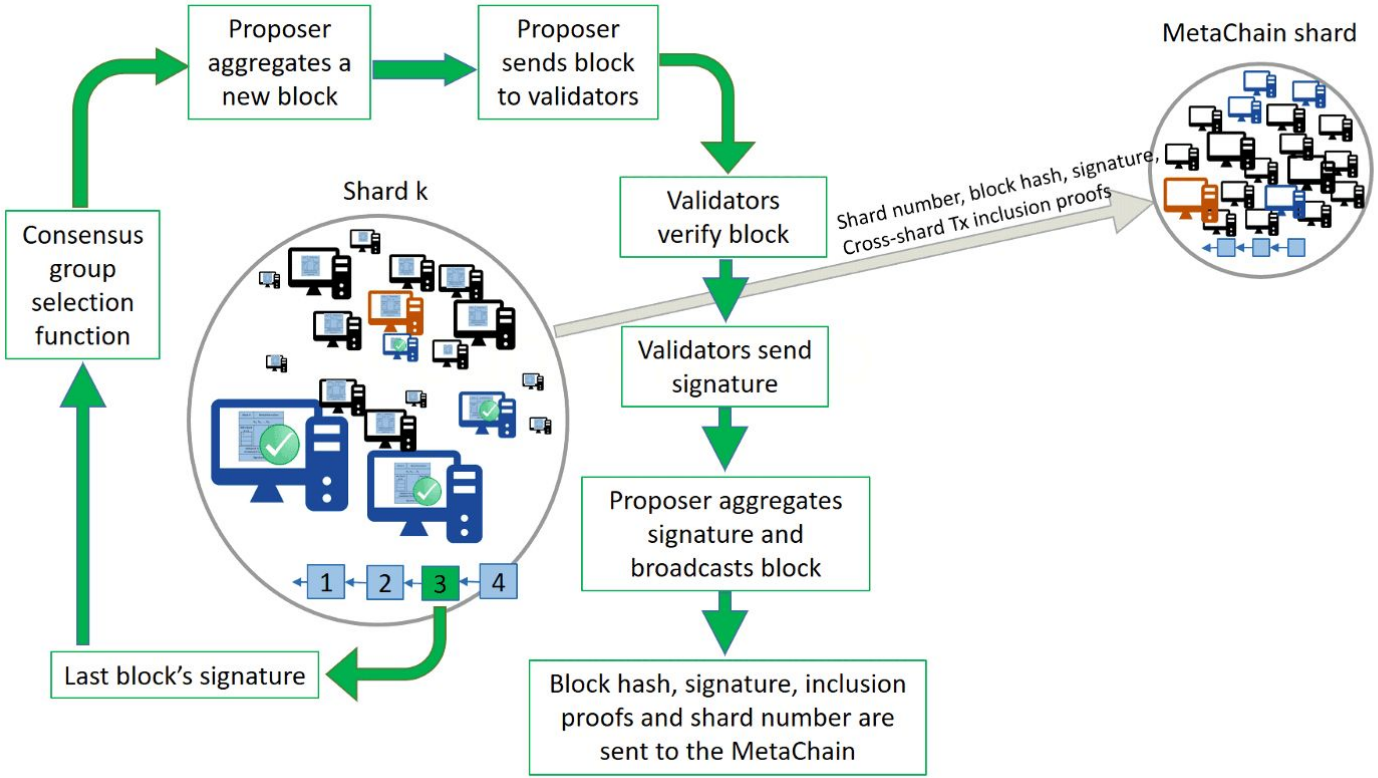
Secure Proof of Stake



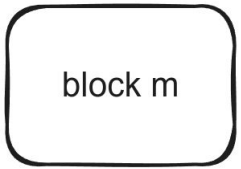
Secure Proof of Stake



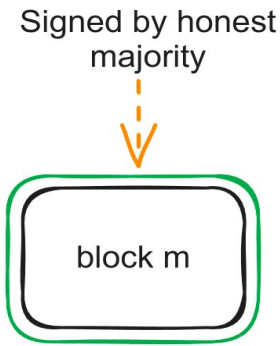
Secure Proof of Stake



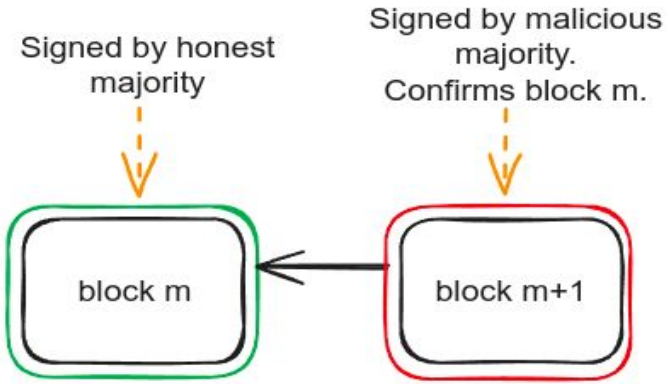
Security



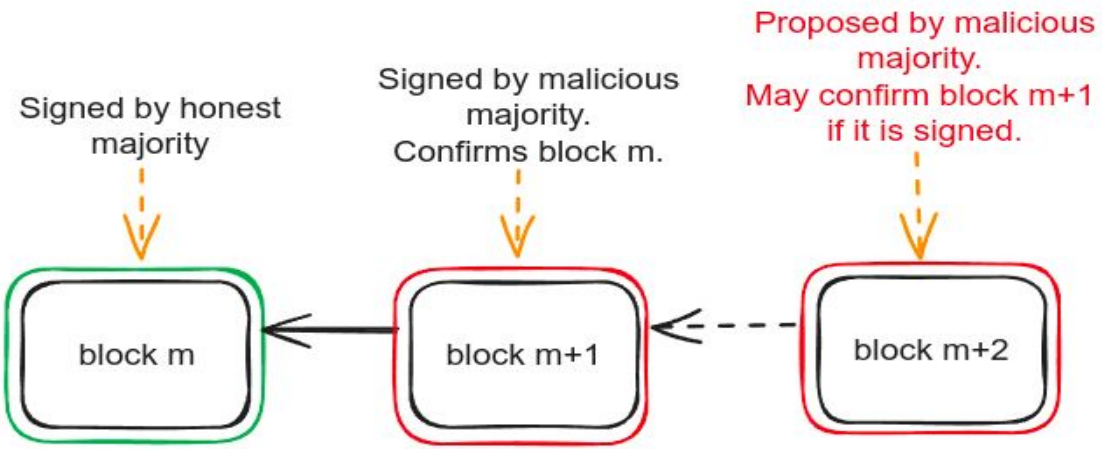
Security



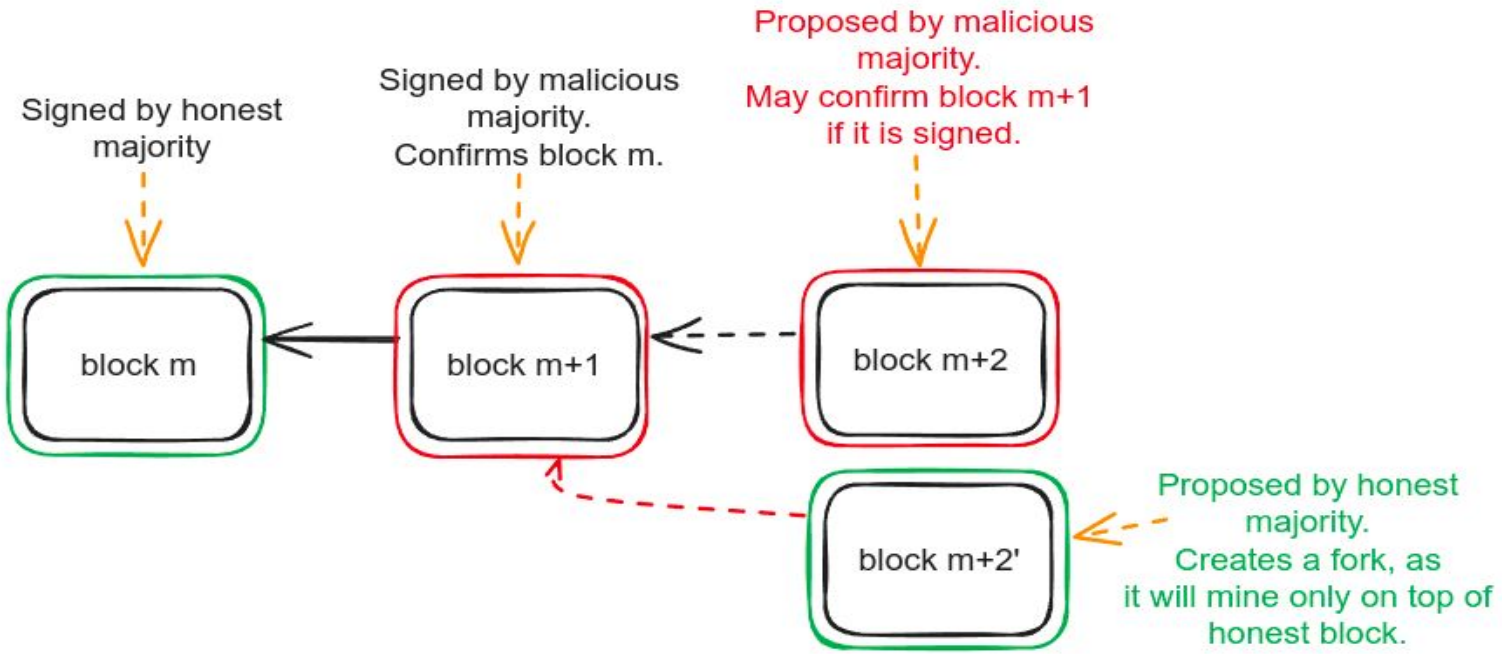
Security


Signed by honest majority
Signed by malicious majority. Confirms block m.
block m
block m+1

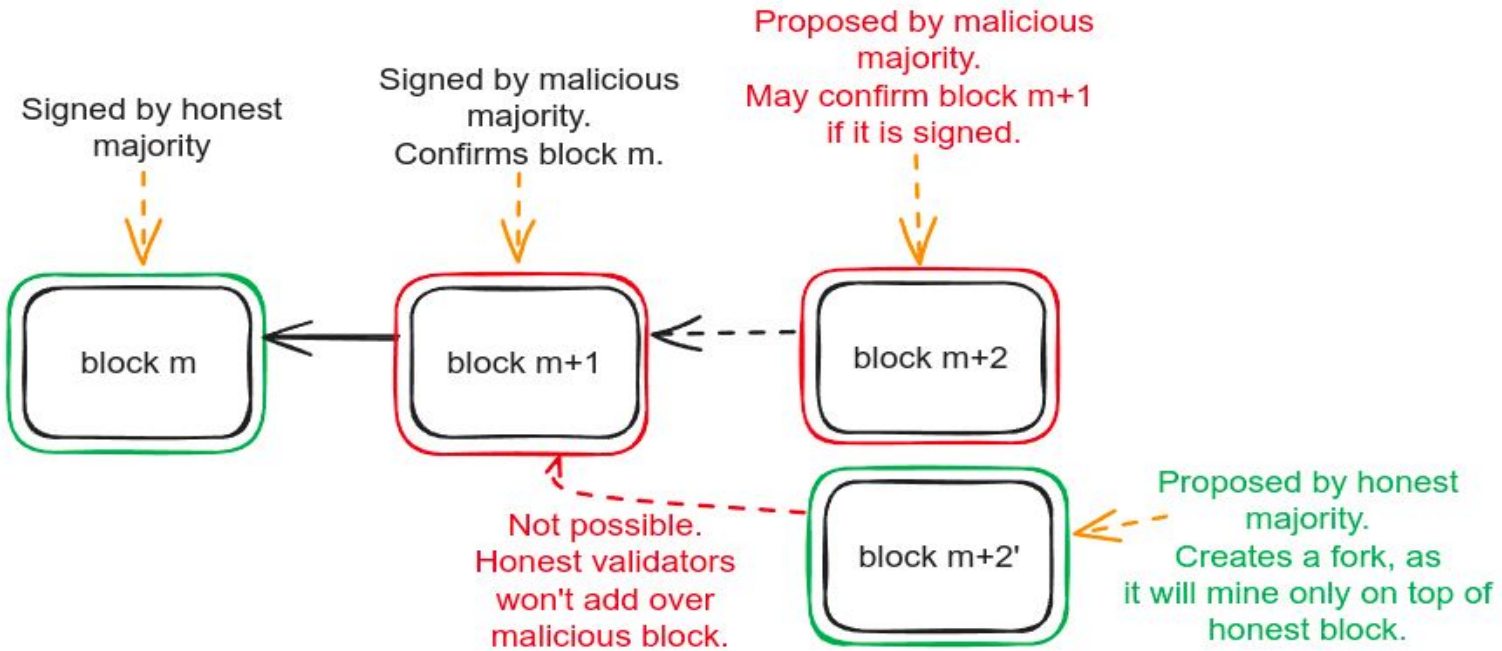
Security


Signed by honest majority
Signed by malicious majority. Confirms block m.
Proposed by malicious majority. May confirm block m+1 if it is signed.
block m
block m+1
block m+2

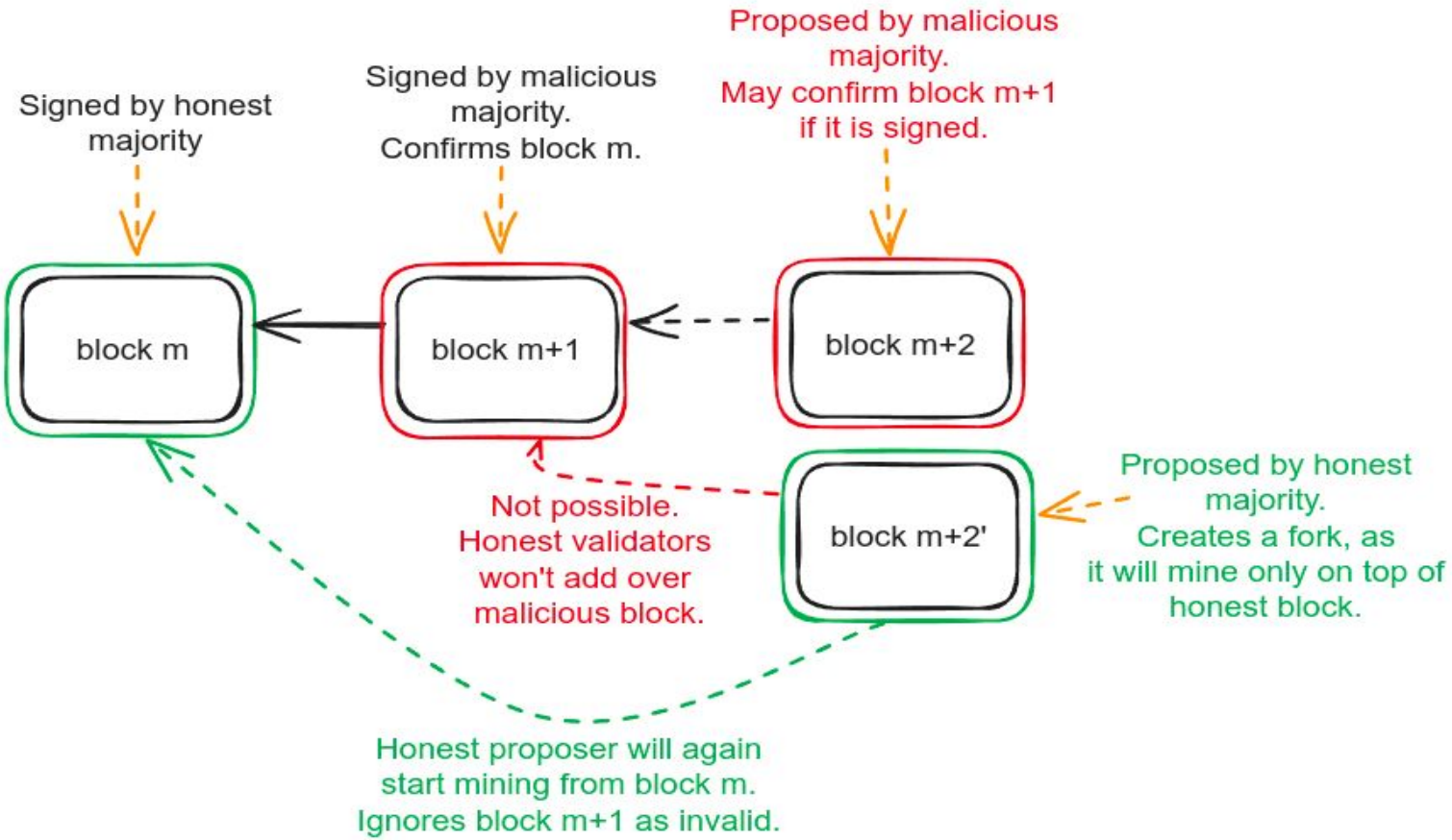
Security


Signed by honest majority
Signed by malicious majority. Confirms block m.
Proposed by malicious majority. May confirm block m+1 if it is signed.
block m
block m+1
block m+2
block m+2'
Proposed by honest majority. Creates a fork, as it will mine only on top of honest block.

Security


Signed by honest majority
Signed by malicious majority. Confirms block m.
Proposed by malicious majority. May confirm block m+1 if it is signed.
block m
block m+1
block m+2
Not possible. Honest validators won't add over malicious block.
block m+2'
Proposed by honest majority. Creates a fork, as it will mine only on top of honest block.

Security

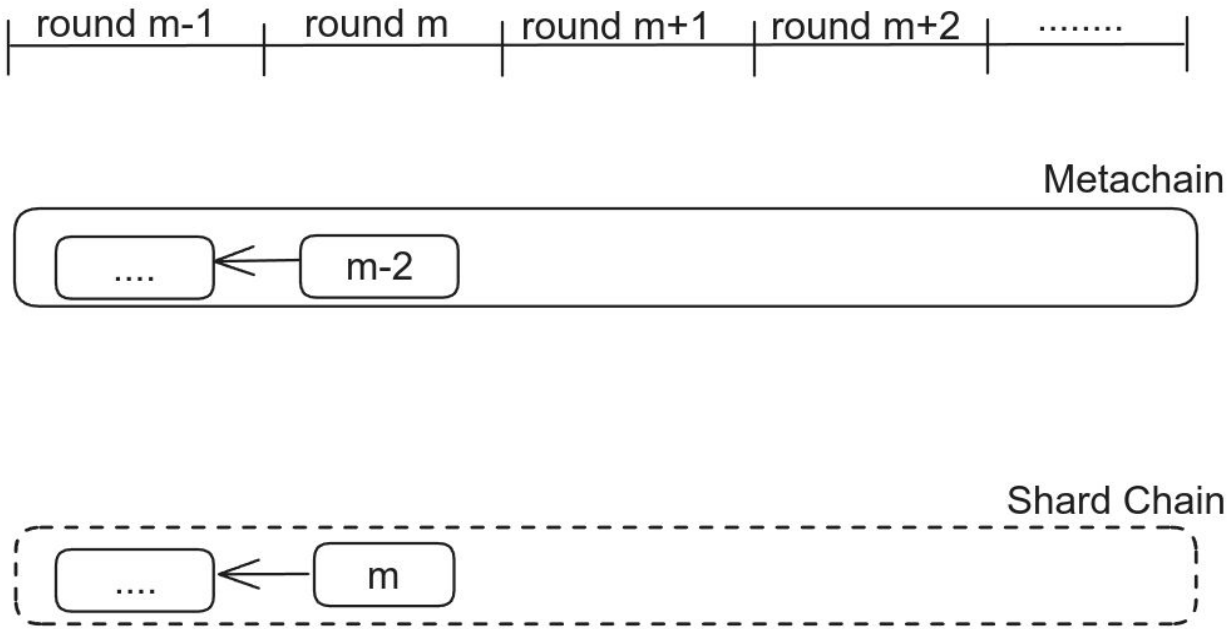

Signed by honest majority
Signed by malicious majority. Confirms block m.
Proposed by malicious majority. May confirm block m+1 if it is signed.
block m
block m+1
block m+2
block m+2'
Not possible. Honest validators won't add over malicious block.
Proposed by honest majority. Creates a fork, as it will mine only on top of honest block.
Honest proposer will again start mining from block m. Ignores block m+1 as invalid.

Intra-Shard Transaction

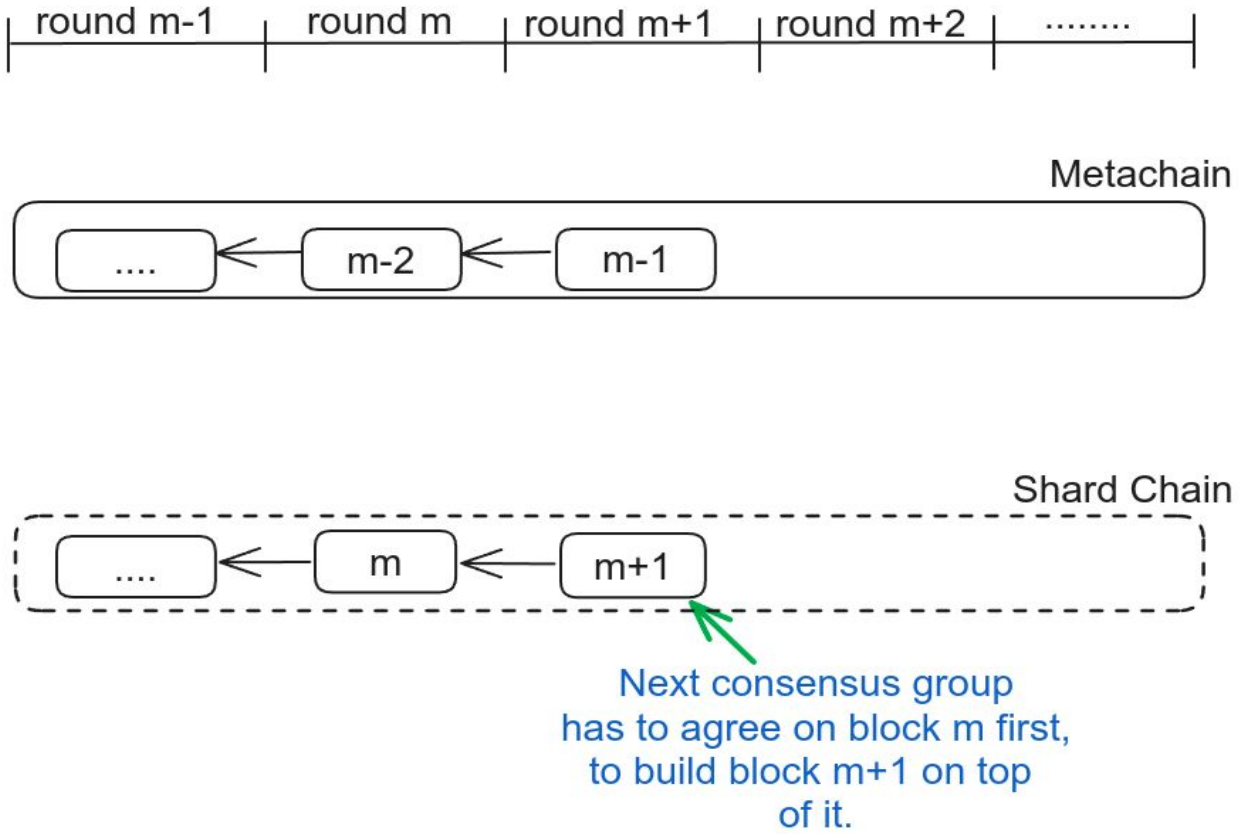
- ➔ An Intra-Shard transaction can be called processed when it is included in a block and added to the respective chain of the shard associated with its *source address*.

- ➔ An Intra-Shard transaction can be called confirmed when the header of the block where the transaction is present is included in a block added to the metachain.

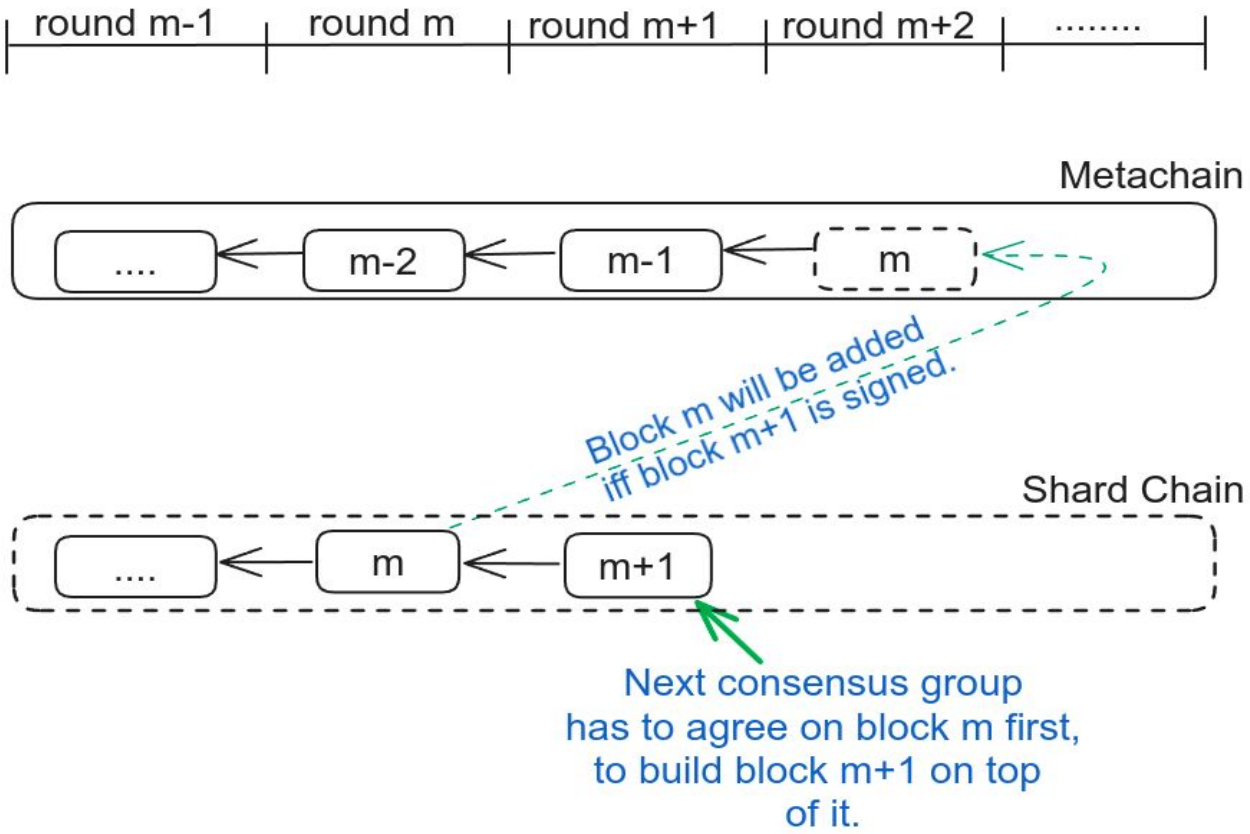
Intra-Shard Transaction



Intra-Shard Transaction



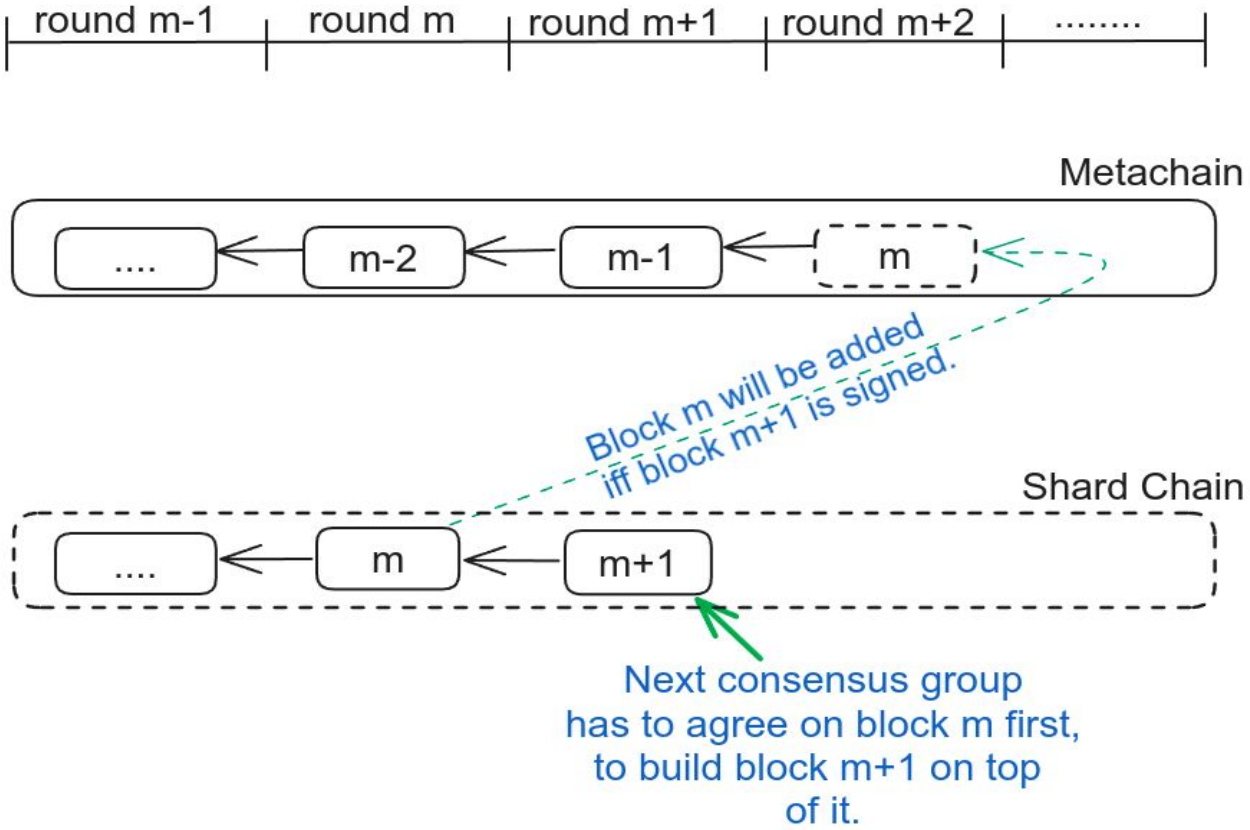
Intra-Shard Transaction



Intra-Shard Transaction

1 Round to Process

3 Rounds to Confirm

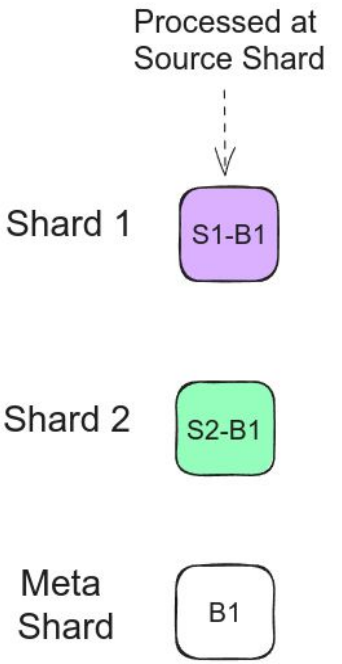


Cross-Shard Transaction

- A Cross-Shard transaction can be called processed when it is included in a block and added to the respective chain of the shard associated with its *destination address*.

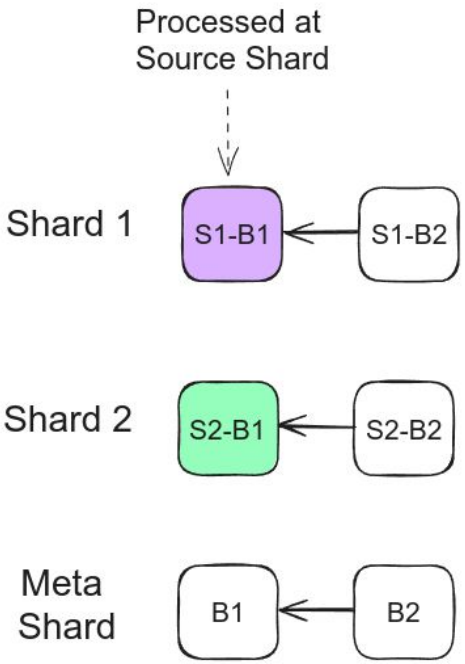
- A Cross-Shard transaction can be called confirmed when the header of the cross-shard block where the transaction is present is included in a block added to the metachain.

Cross-Shard Transaction



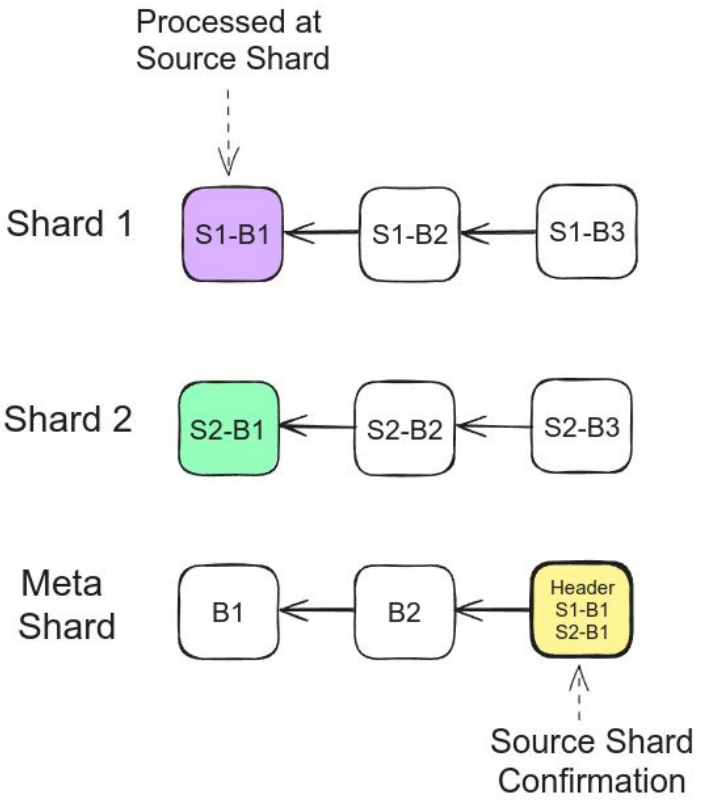
Si-Bj represents the j'th block of shard "i".

Cross-Shard Transaction



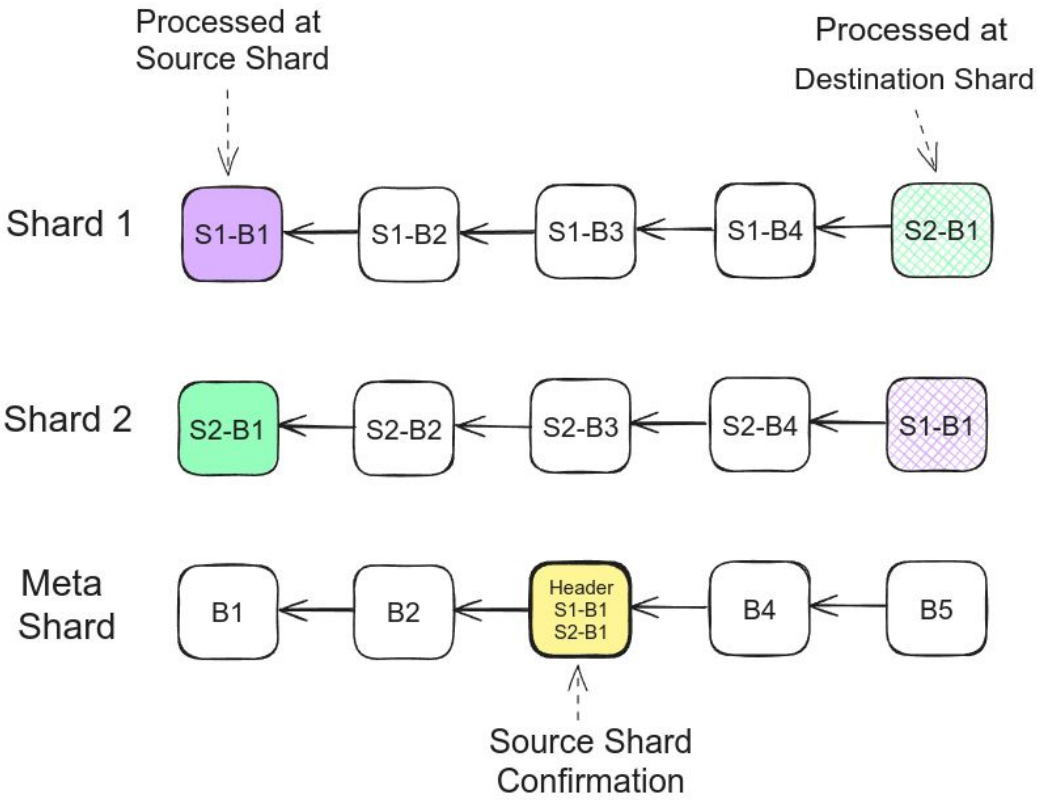
Si-Bj represents the j'th block of shard "i".

Cross-Shard Transaction



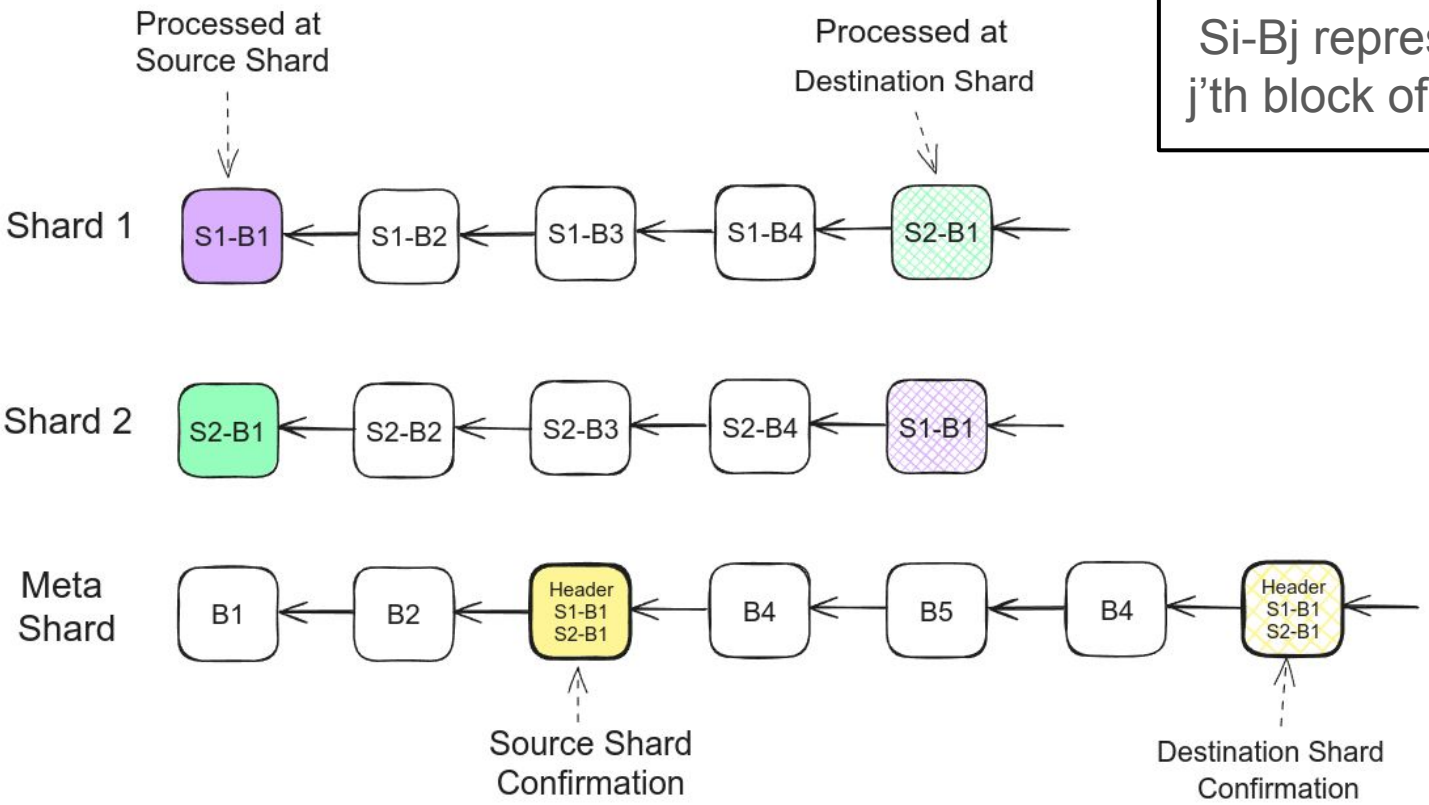
Si-Bj represents the j'th block of shard "i".

Cross-Shard Transaction



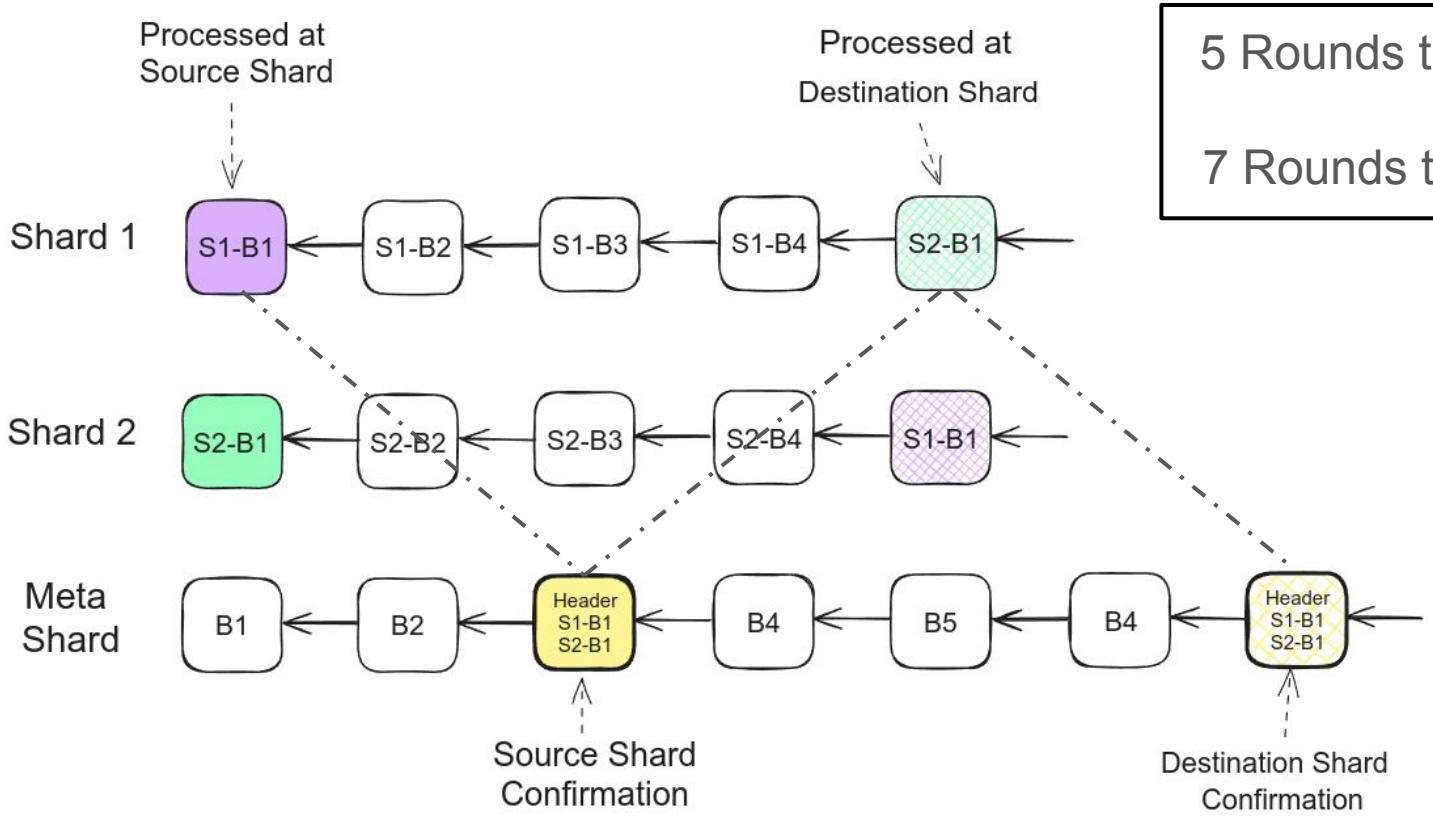
Si-Bj represents the j'th block of shard "i".

Cross-Shard Transaction



Si-Bj represents the j'th block of shard "i".

Cross-Shard Transaction



5 Rounds to Process

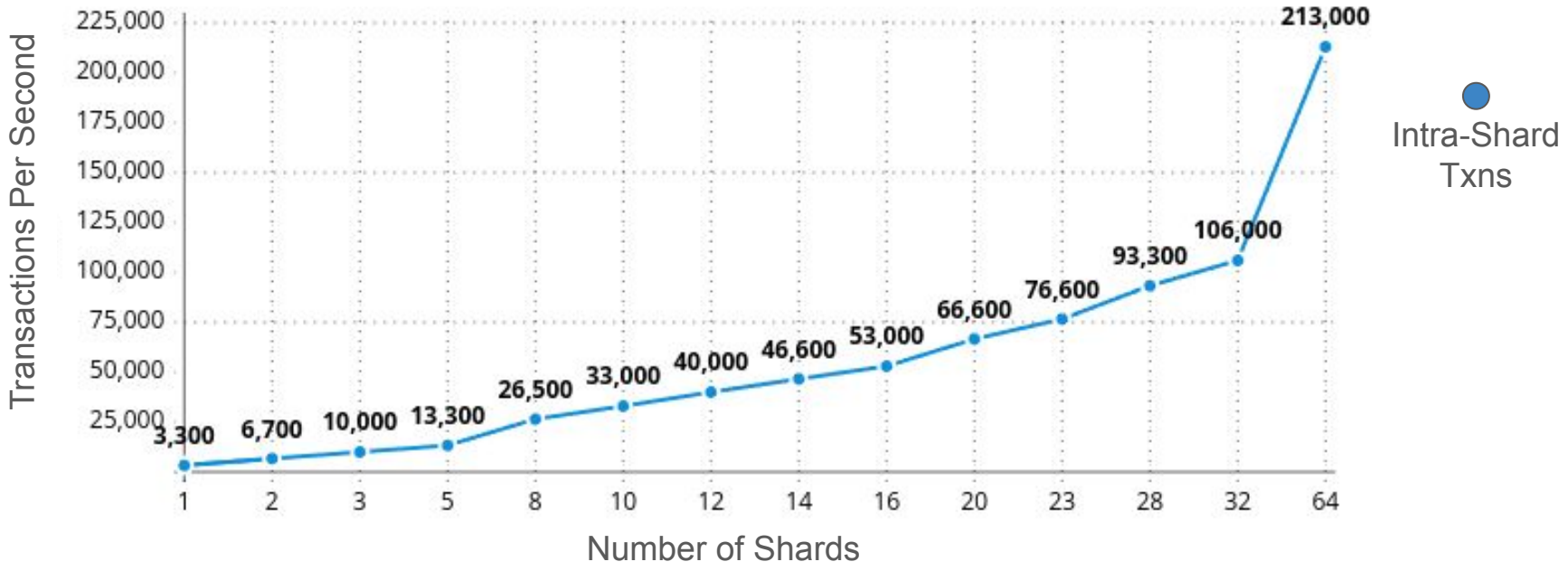
7 Rounds to Confirm

Experiment

- ➔ Simulated their local testnet with different configuration of shards. Integrated their Proxy Node and Go & Python SDK with the local testnet.
- ➔ Made a Load Testing repository in Python for interacting with the local testnet for creating accounts, loading various types of transaction into the network and functions to analyse the network status.
- ➔ Made changes in the local testnet code to work along with Load Testing script.
- ➔ Every experiment is run for 10 epochs of 100 rounds with different loads.
- ➔ Finally analysed the performance of the network and compared it with other popular models.

Load - Intra-Shard Transactions

Loading only intra-shard transactions to every shard such that each shard reaches maximum throughput at each round.



Load - Intra-Shard Transactions (Observations)

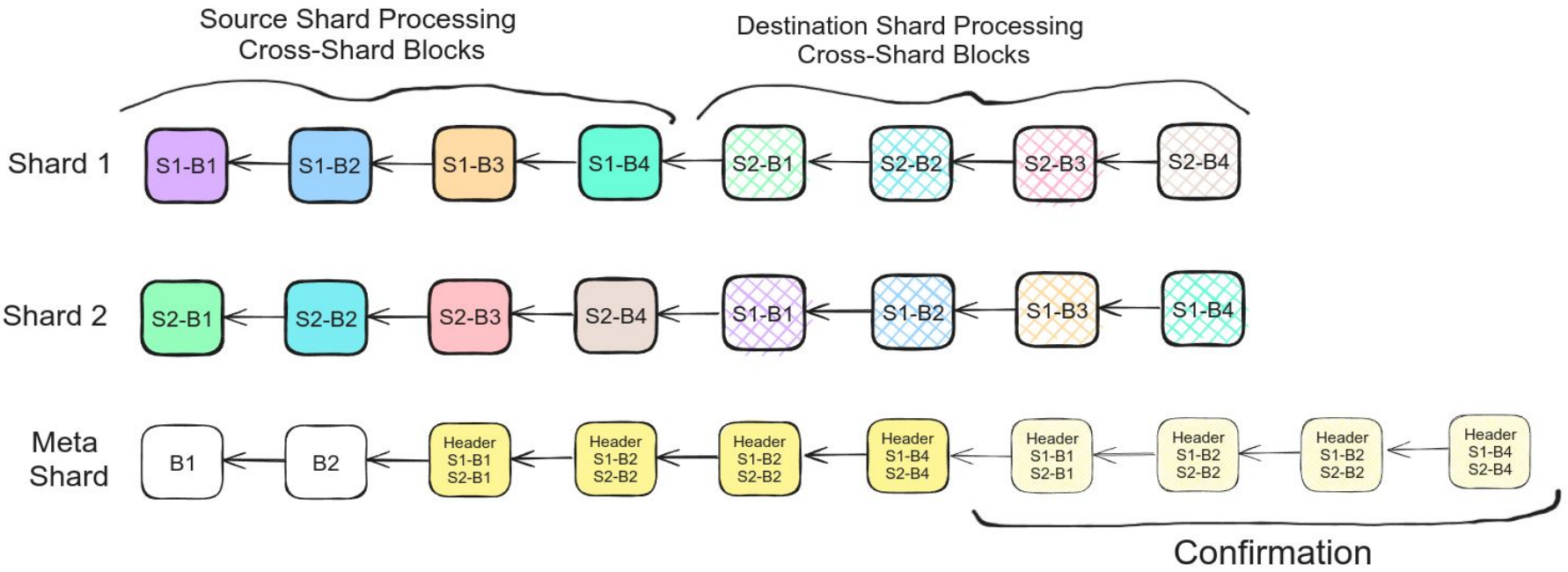
- ➔ The throughput almost increases linearly as expected.
- ➔ Whenever the network reaches the threshold, number of shard increases, to increase the overall throughput.
- ➔ The metachain may act as the bottleneck for confirmation latency if it gets full with shard block headers, thus requiring more than 1 meta block to confirm.
- ➔ But this is not practical in real time. When the number of shard increases, there can be huge number of cross shard transactions initiated.

Load - Cross-Shard Transactions

Cross-Shard transactions need 5 rounds for processing.
The transaction can be said processed only when it gets added to the chain of the destination shard.

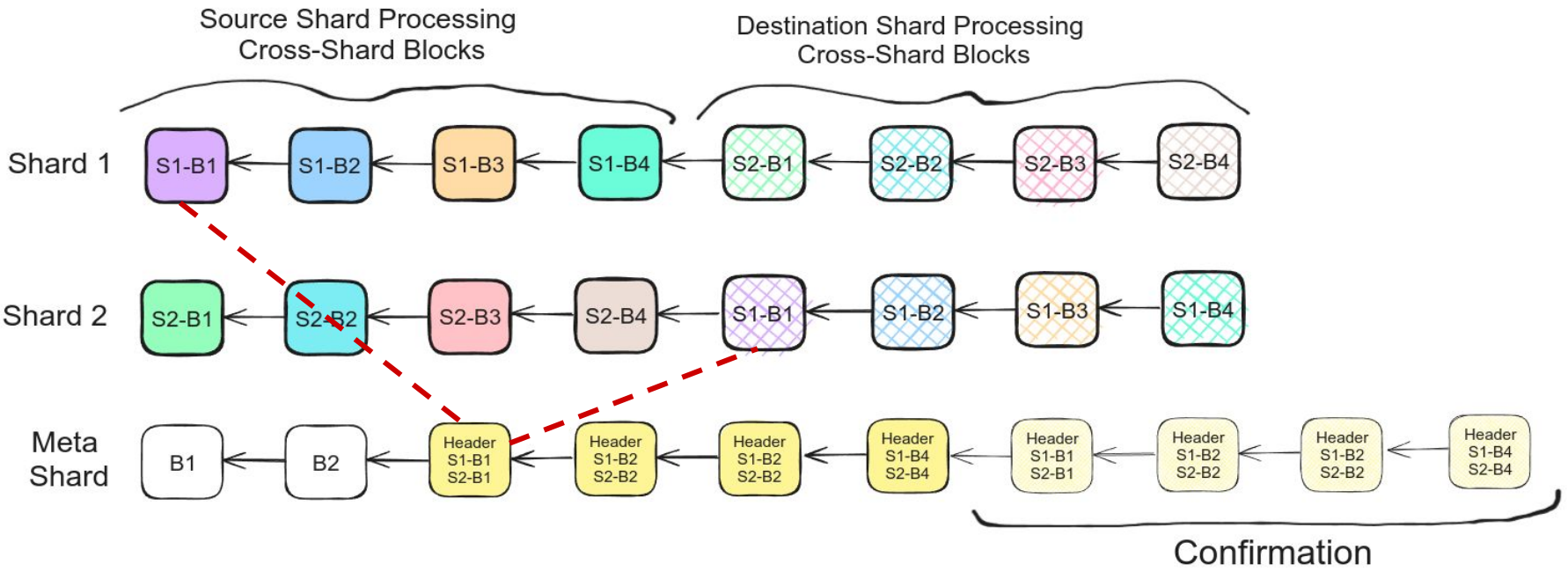
Load - Cross-Shard Transactions

Cross-Shard transactions need 5 rounds for processing.
The transaction can be said processed only when it gets added to the chain of the destination shard.



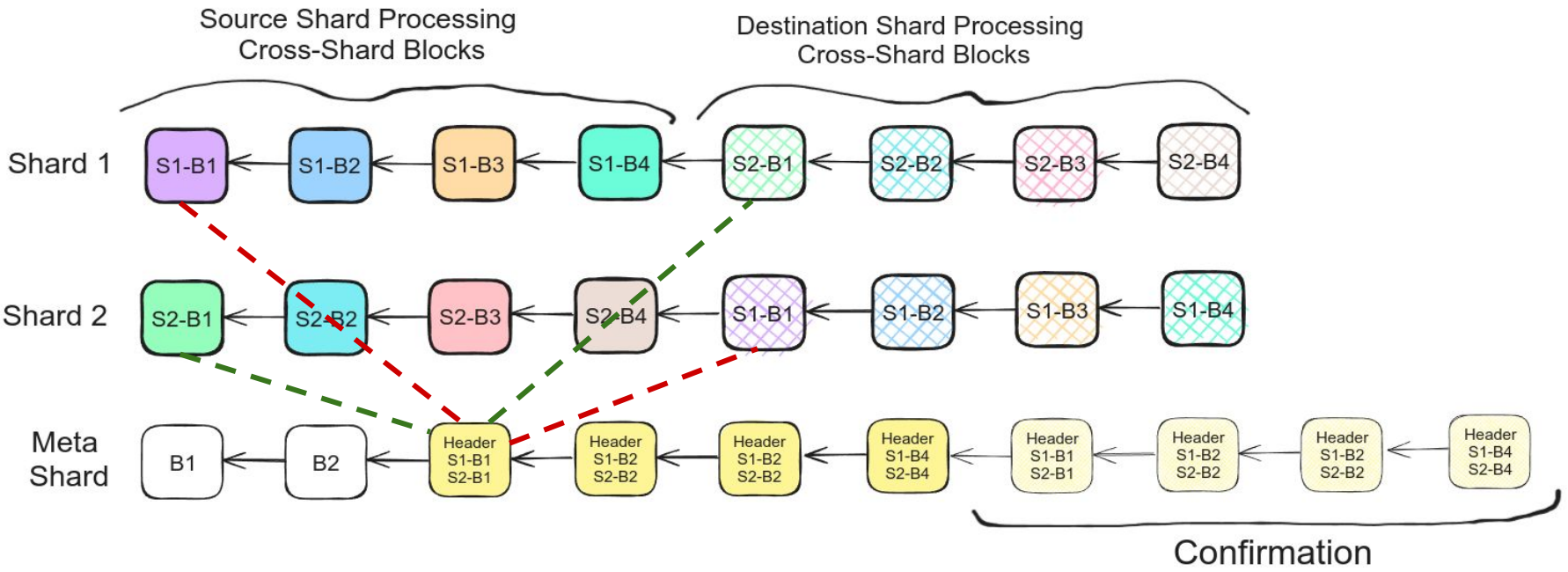
Load - Cross-Shard Transactions

Cross-Shard transactions need 5 rounds for processing.
The transaction can be said processed only when it gets added to the chain of the destination shard.



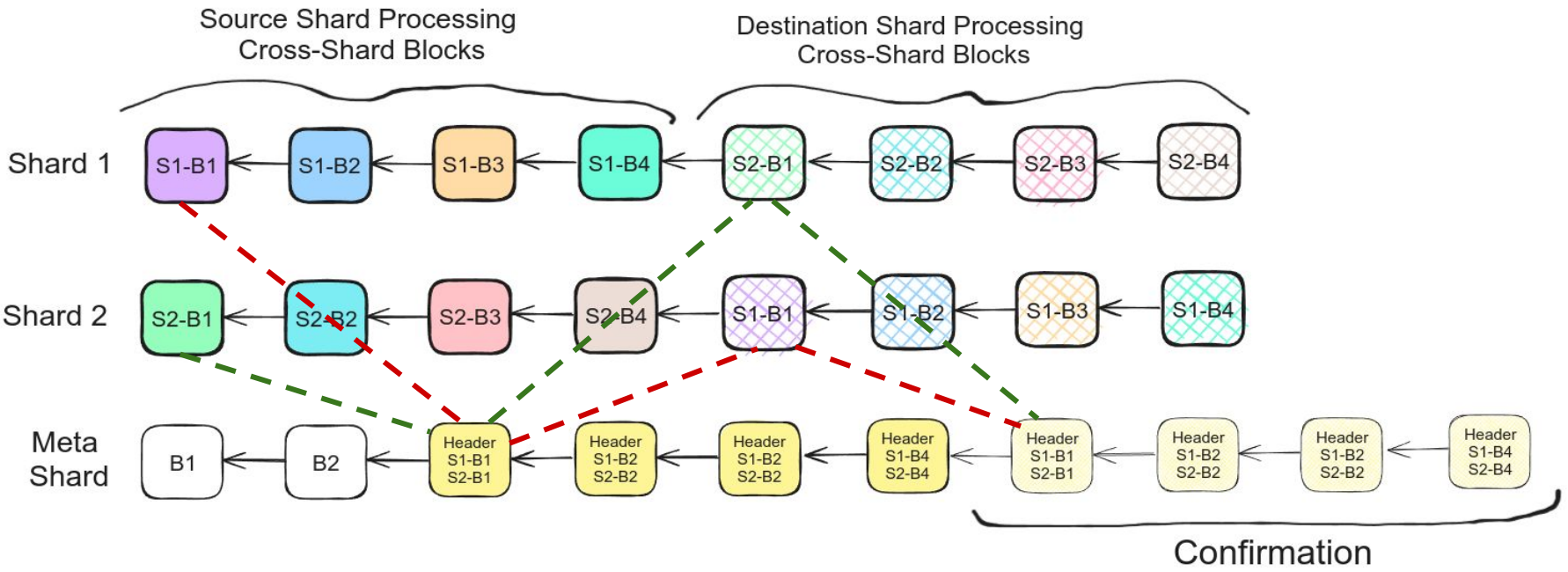
Load - Cross-Shard Transactions

Cross-Shard transactions need 5 rounds for processing.
The transaction can be said processed only when it gets added to the chain of the destination shard.

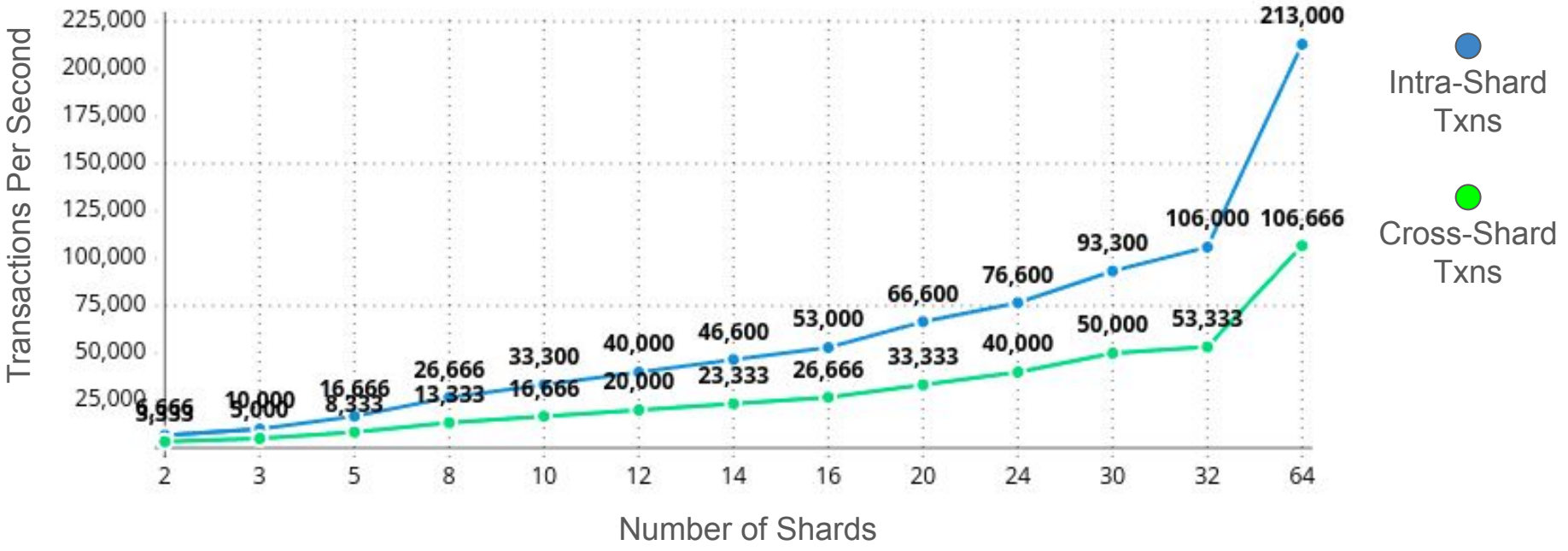


Load - Cross-Shard Transactions

Cross-Shard transactions need 5 rounds for processing.
The transaction can be said processed only when it gets added to the chain of the destination shard.



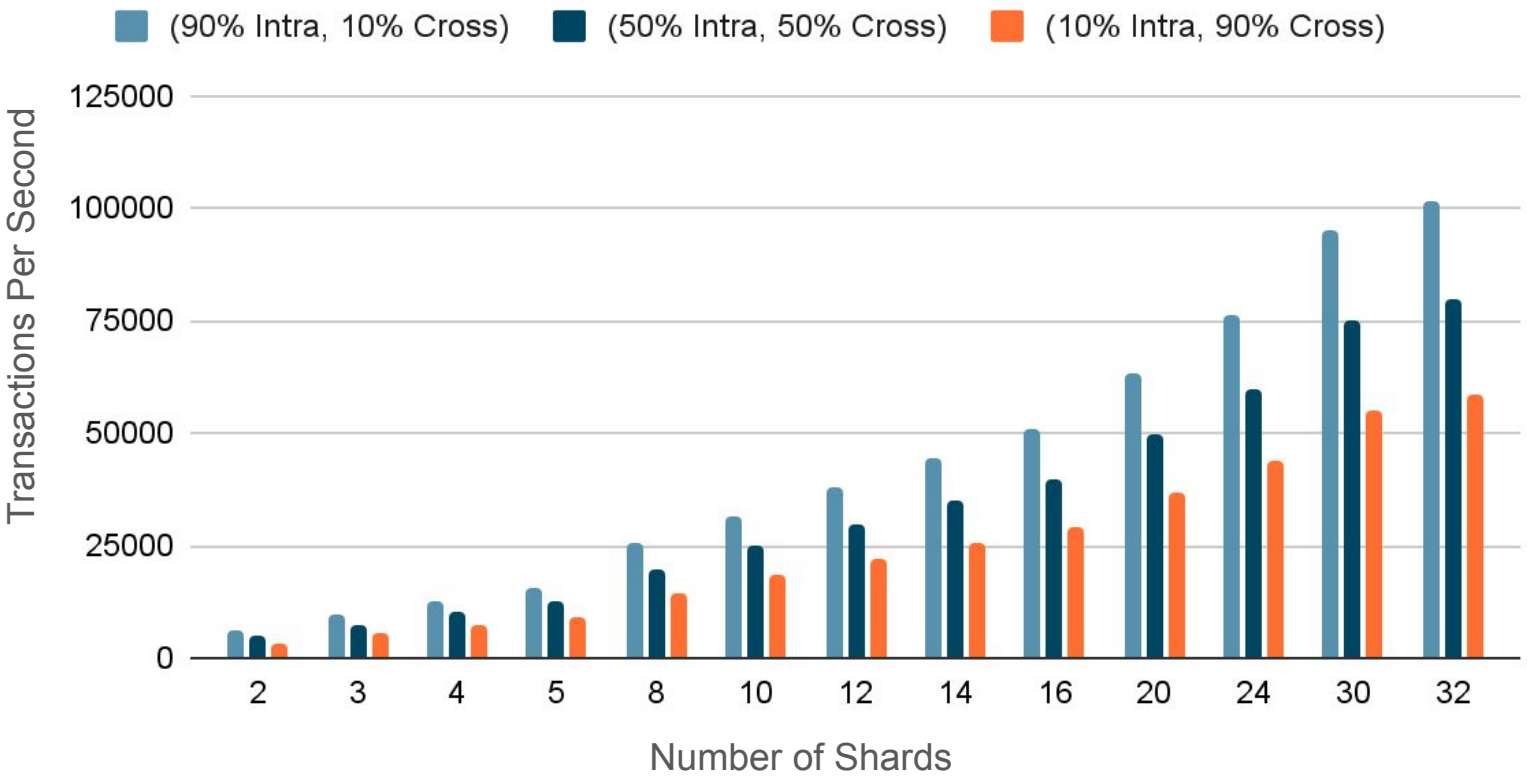
Load - Cross-Shard Transactions



Load - Cross-Shard Transactions (Observations)

- ➔ Every transaction's destination account has been chosen uniformly from all the accounts in the network.
- ➔ The cross-shard gives exactly half the throughput achieved in Intra-Shard when every block is filled up i.e. (Peak Throughput).
- ➔ The peak throughput is only achievable when a shard is receiving maximum transactions per second from other shard.
  Otherwise, increase in shard may lead to more number of cross-shard transactions and thus a dip in the throughput.
- ➔ There can be some cases where one shard gets overwhelmed with cross-shard transactions from other shards.

Load - Mixed-Shard Transactions

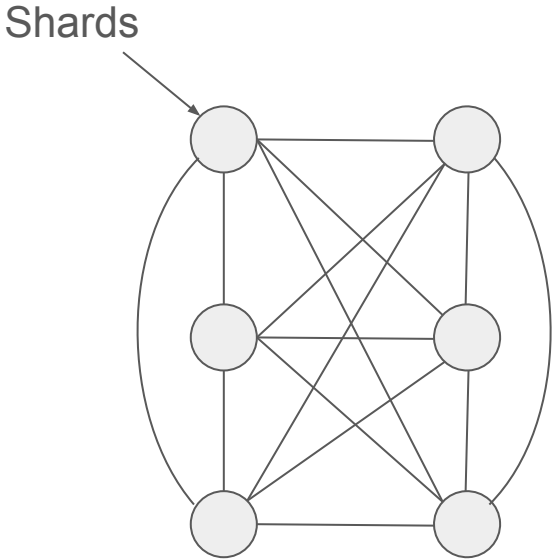


Observations

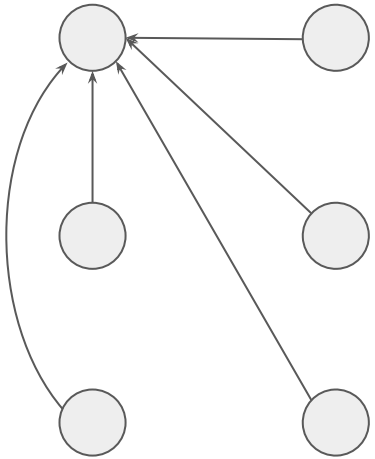
- ➔ Every cross transaction's destination account has been chosen uniformly from all the accounts in the network.
- ➔ With increase in number of shards, the throughput increases linearly even in the case of Mixed transactions.
- ➔ The same assumption is made here that every block is receiving full transactions, to get measurement of the peak throughput.

Load - Cross-Shard Transactions (OvA)

Send Cross-Shard transaction to only one specific Shard.

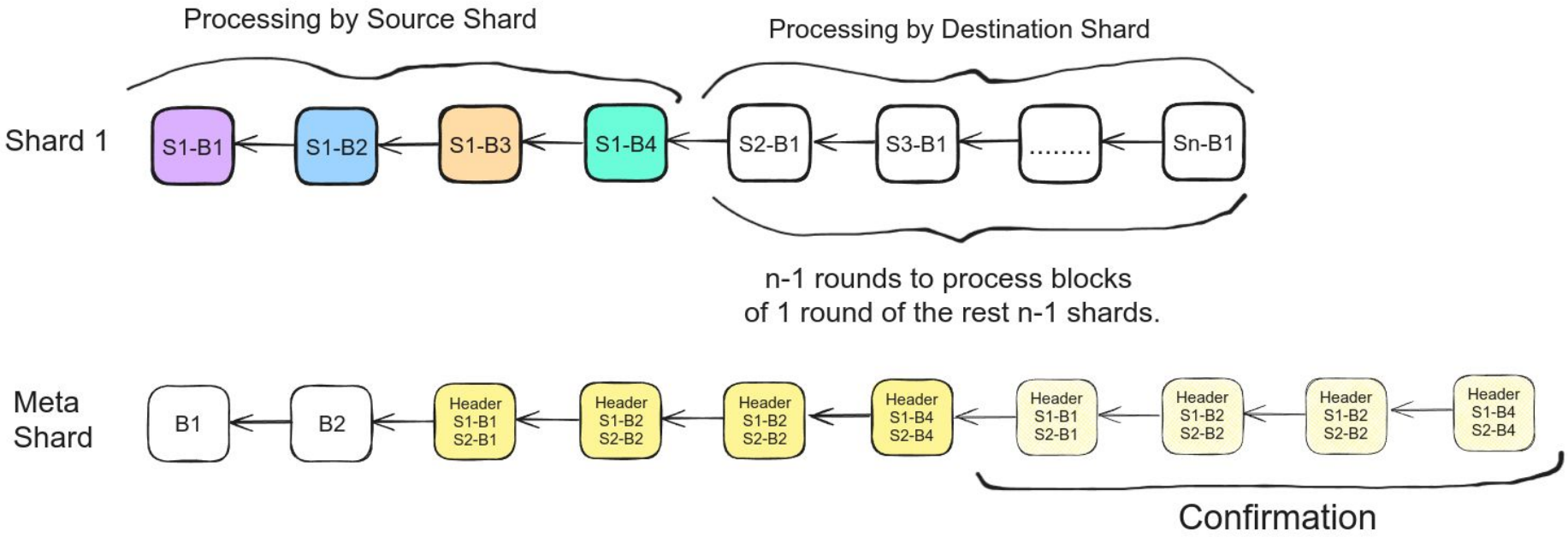


Before

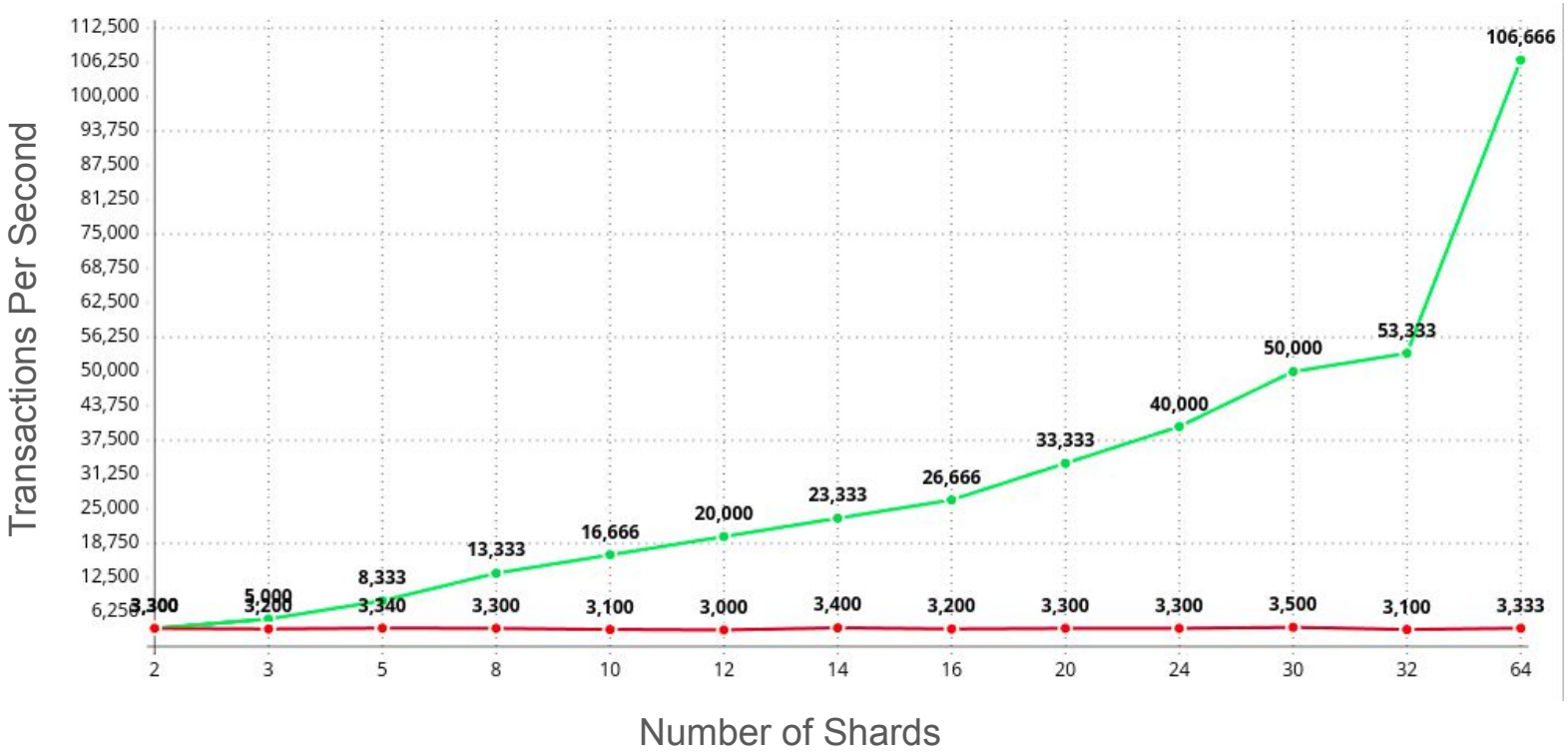


Now
One Vs. All (OvA)

Load - Cross-Shard Transactions (OvA)



Load - Cross-Shard Transactions (OvA)



Observations

- ➔ With a timeline of 5 epochs where the source shard can create 4 blocks at the first 4 rounds, the rest of the rounds it spent finalizing the cross-shard blocks from other transactions. Thus, even if we increase the size of the shard, the throughput remains const approx 3500 TPS.

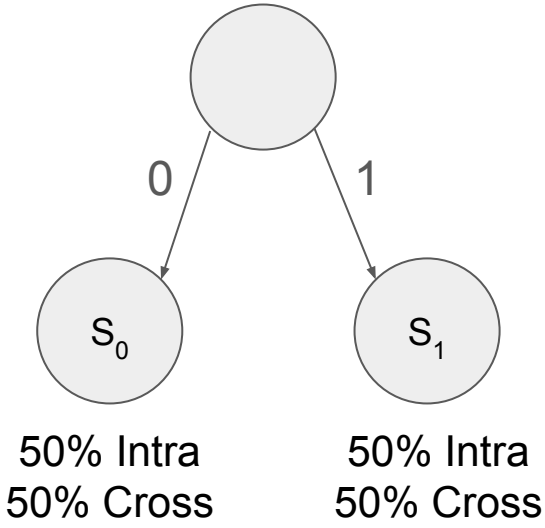
- ➔ Shard Re-Adjustment or any other method must be used to distribute the cross shard transactions to more shards.

Increase in Percentage of Cross-Shard with Increase in Shard

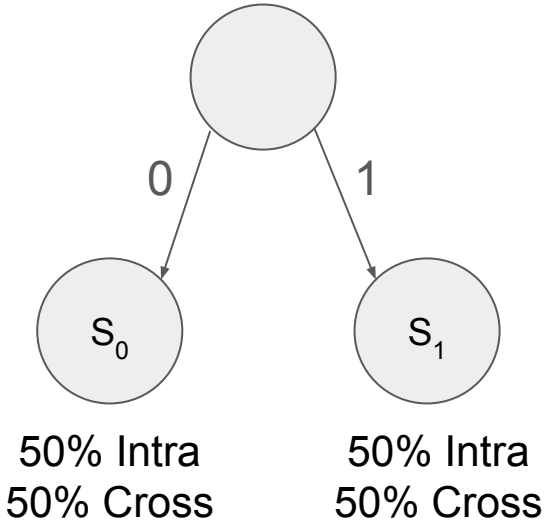
As we increase the number of shards,
what is the percentage increase in cross-shard txn?

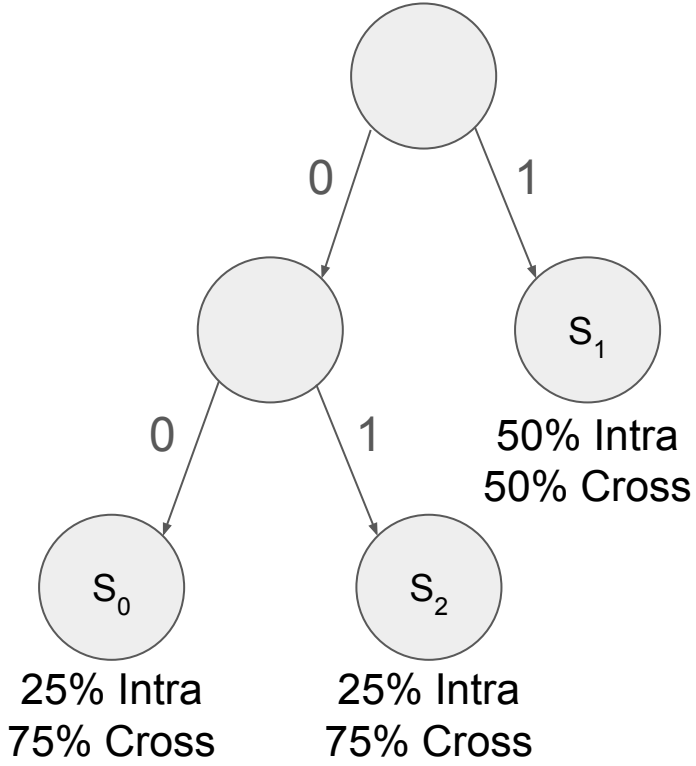
We experiment for a fixed period of 5 epochs and 1000 accounts.
The accounts for transactions are uniformly picked up from all the shards.

Increase in Percentage of Cross-Shard with Increase in Shard

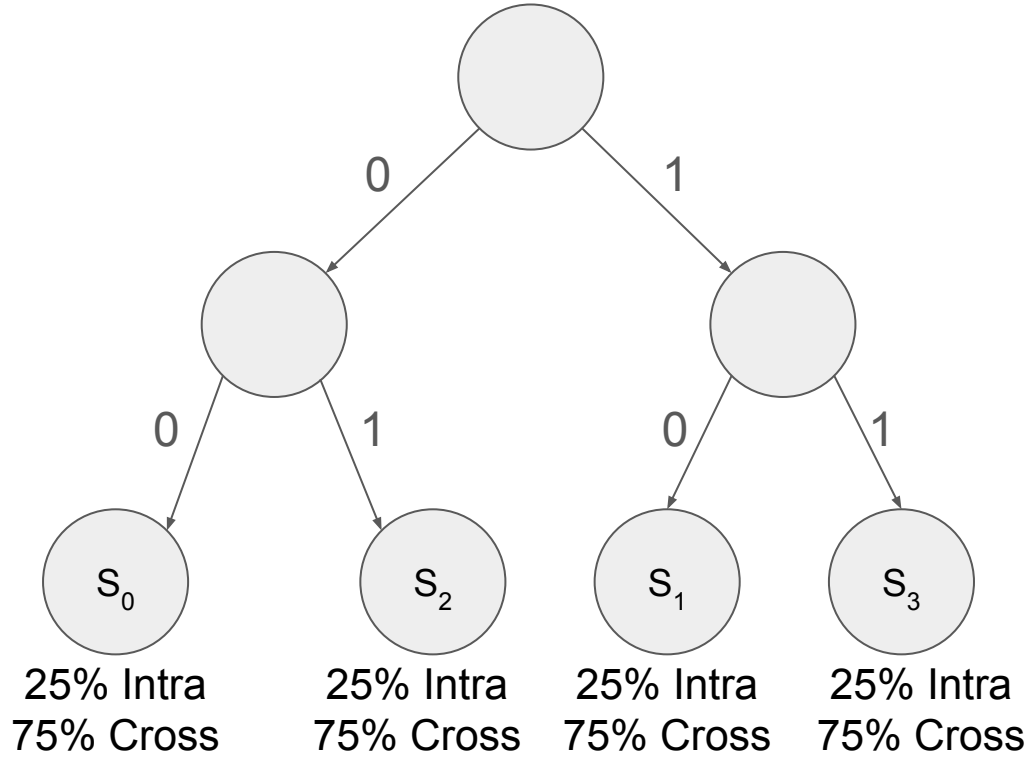


Increase in Percentage of Cross-Shard with Increase in Shard

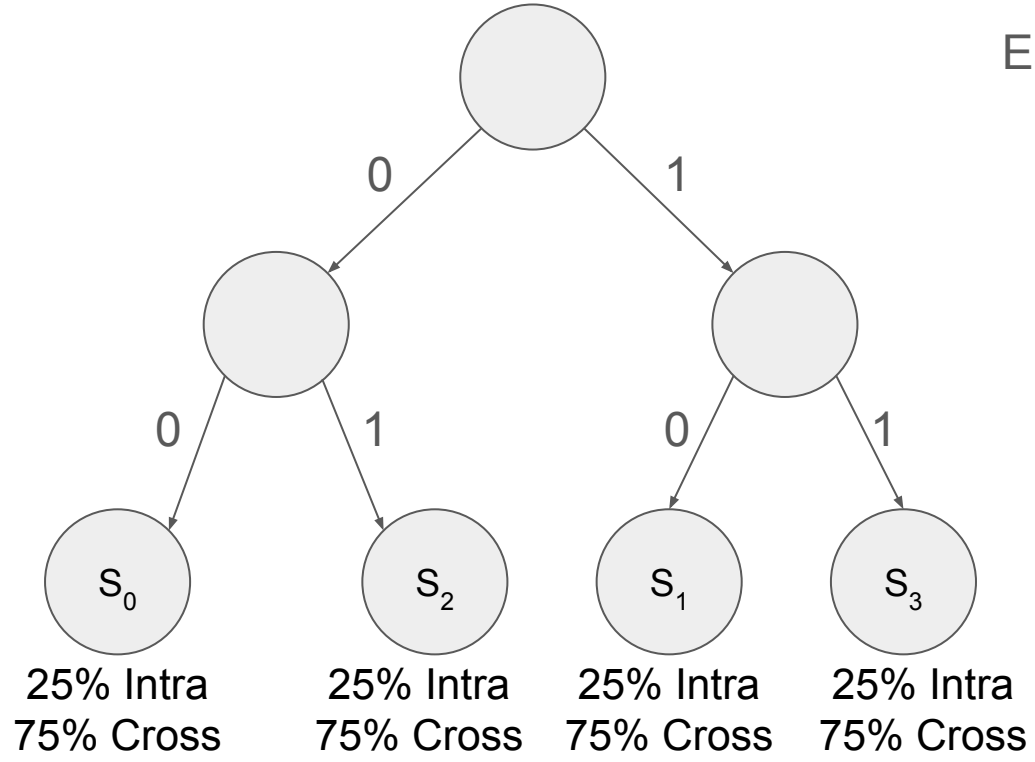




Increase in Percentage of Cross-Shard with Increase in Shard

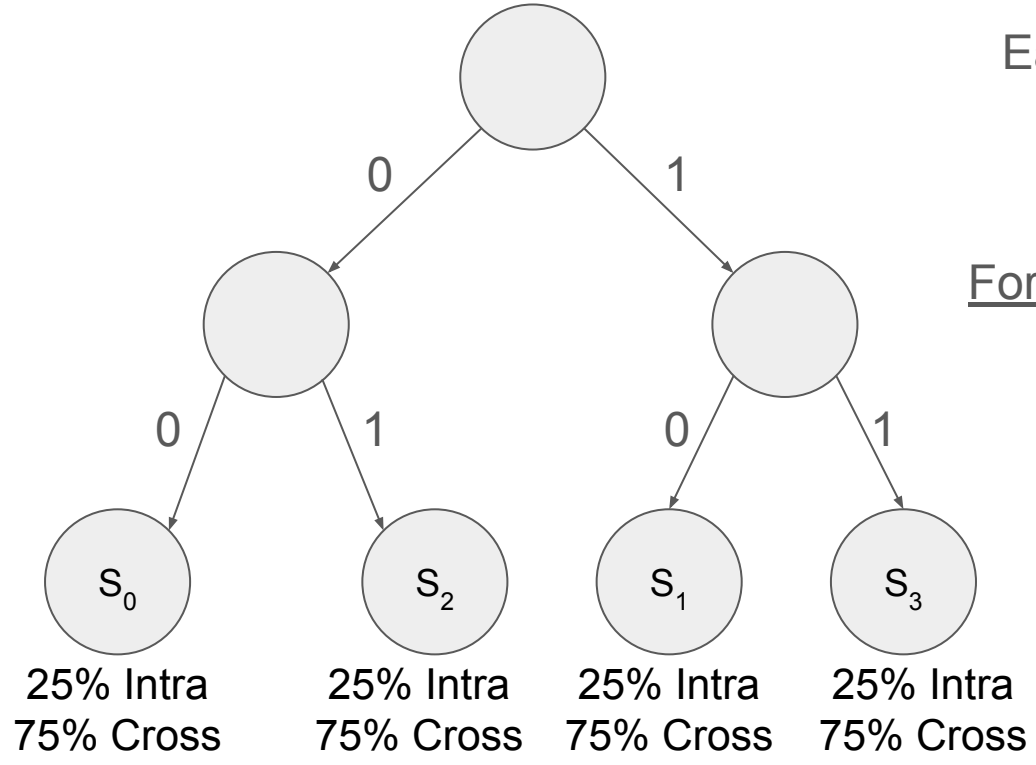


Increase in Percentage of Cross-Shard with Increase in Shard



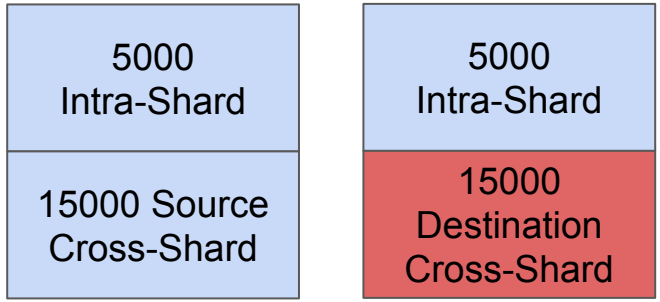
Each Block in a Round (20k Txns)
Intra-Shard: 5000 Txns
Cross-Shard: 15000 Txns

Increase in Percentage of Cross-Shard with Increase in Shard



Each Block in a Round (20k Txns)
Intra-Shard: 5000 Txns
Cross-Shard: 15000 Txns

For consecutive 8 Rounds of a Shard



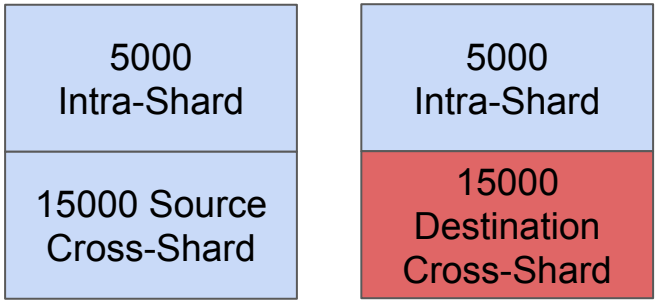
First 4 Rounds

Next 4 Rounds

Increase in Percentage of Cross-Shard with Increase in Shard

Each Block in a Round (20k Txns)
Intra-Shard: 5000 Txns
Cross-Shard: 15000 Txns

For consecutive 8 Rounds of a Shard

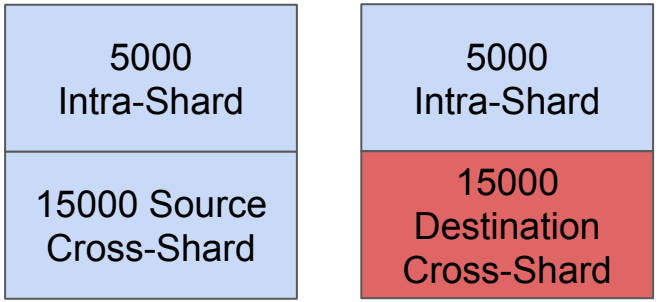


For 1 Shard:
#Intra = 5000 x 8 = 40000
#Cross = 15000 x 4 = 60000

Increase in Percentage of Cross-Shard with Increase in Shard

Each Block in a Round (20k Txns)
Intra-Shard: 5000 Txns
Cross-Shard: 15000 Txns

For consecutive 8 Rounds of a Shard



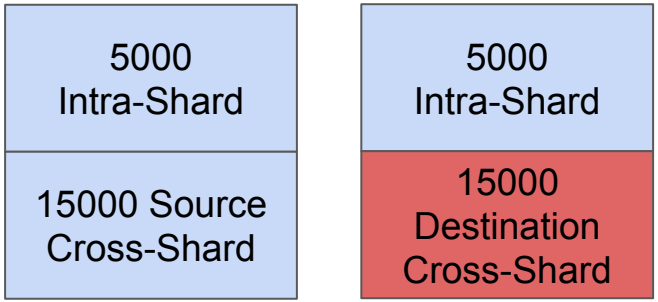
For 1 Shard:
#Intra = 5000 x 8 = 40000
#Cross = 15000 x 4 = 60000

For 4 Shards:
#Intra = 40000 x 4 = 160000
#Cross = 60000 x 4 = 240000

Increase in Percentage of Cross-Shard with Increase in Shard

Each Block in a Round (20k Txns)
Intra-Shard: 5000 Txns
Cross-Shard: 15000 Txns

For consecutive 8 Rounds of a Shard



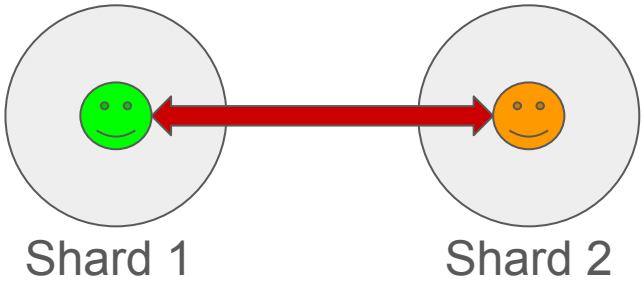
For 1 Shard:
#Intra = 5000 x 8 = 40000
#Cross = 15000 x 4 = 60000

For 4 Shards:
#Intra = 40000 x 4 = 160000
#Cross = 60000 x 4 = 240000

#Percentage of Cross-Shard = 60%
(Same as received in Experiment)

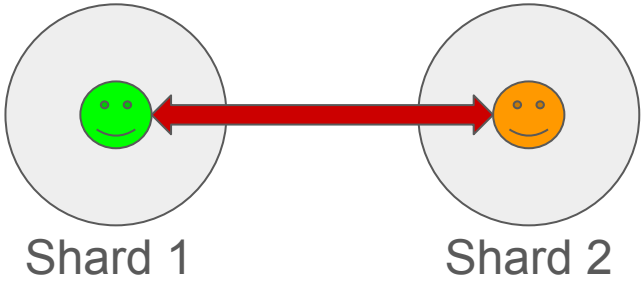
Increase in Percentage of Cross-Shard with Increase in Shard

Minimising Cross-Shard by Opening Additional Accounts

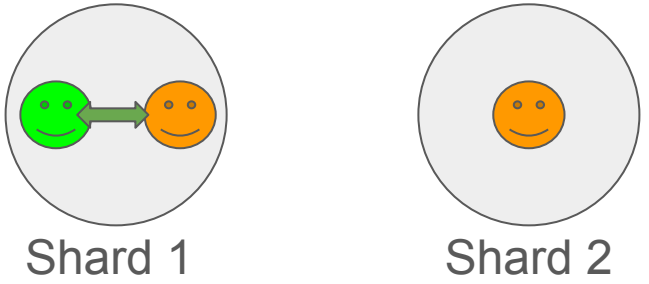


Lots of Transactions between these two accounts.

Minimising Cross-Shard by Opening Additional Accounts

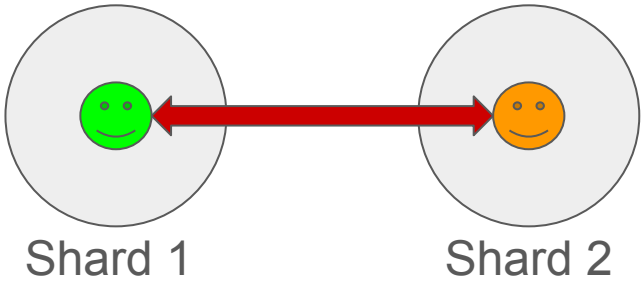


Lots of Transactions between these two accounts.

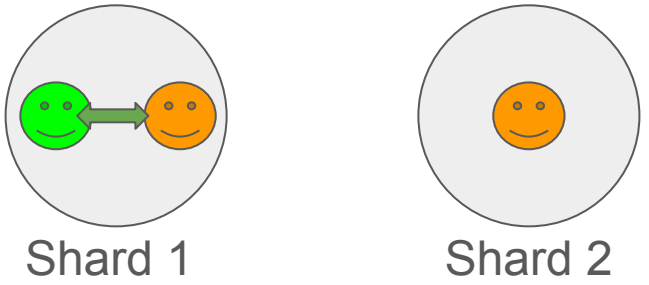


Create new wallet for one of the account in the other shard.

Minimising Cross-Shard by Opening Additional Accounts



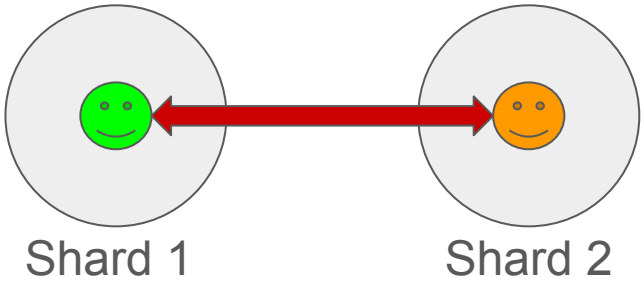
Lots of Transactions between these two accounts.



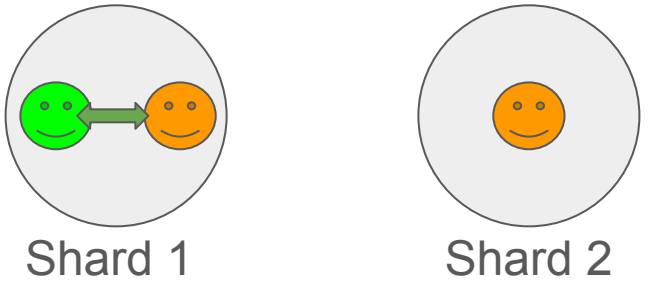
Create new wallet for one of the account in the other shard.

Transaction can be processed 2x speed

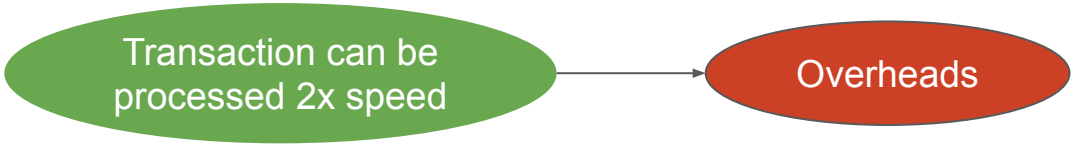
Minimising Cross-Shard by Opening Additional Accounts



Lots of Transactions between these two accounts.



Create new wallet for one of the account in the other shard.



Minimising Cross-Shard by Opening Additional Accounts

- ➔ Accounts with more than 3 transactions between them in a round are optimized with this technique.
- ➔ With 4 shards, and 1000 accounts, and blocks full of transactions, the percentage of Cross-Shard could be reduced to around 48% from 60%.

Conclusion

- MultiVersX successfully implemented sharding and currently running with 3 shards on the main network.
- With almost 3.4M user accounts in the mainnet, it comes out as one of the most successful sharded system currently.
- Their VM supports writing of Smart Contracts with Rust as it runs on Web Assembly.
- Bottlenecks created by Cross-Shard, Meta-Chain and Security Issues need to be overcome to make sharding a goto idea for scalability.
- With this R&D, learnt about practical implementation of sharding along with how to do analysis of sharded system.

References

[1] Multiversx whitepaper.

[2] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In OsDI, volume 99, 1999.

[3] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper, 19(1), August 2012.

[4] Paradigm Research. Elrond: High-throughput public blockchain with adaptive state sharding, 2019.

[5] A. Secure. The zilliqa project: A secure, scalable blockchain platform. 2018.

[6] Alex Skidanov and Illia Polosukhin. Nightshade: Near protocol sharding design. 2019.

[7] Dmitry Tanana. Avalanche blockchain protocol for distributed computing security. In 2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). IEEE, 2019.

[8] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. White paper, 21(2327):4662, 2016.