Ethereum Sharding Concept
以太坊分片概说

Asia-Pacific Ethereum Community Meetup @ Shenzhen
Dec 3, 2017

Ethereum Research
Hsiao-Wei Wang (王筱維)

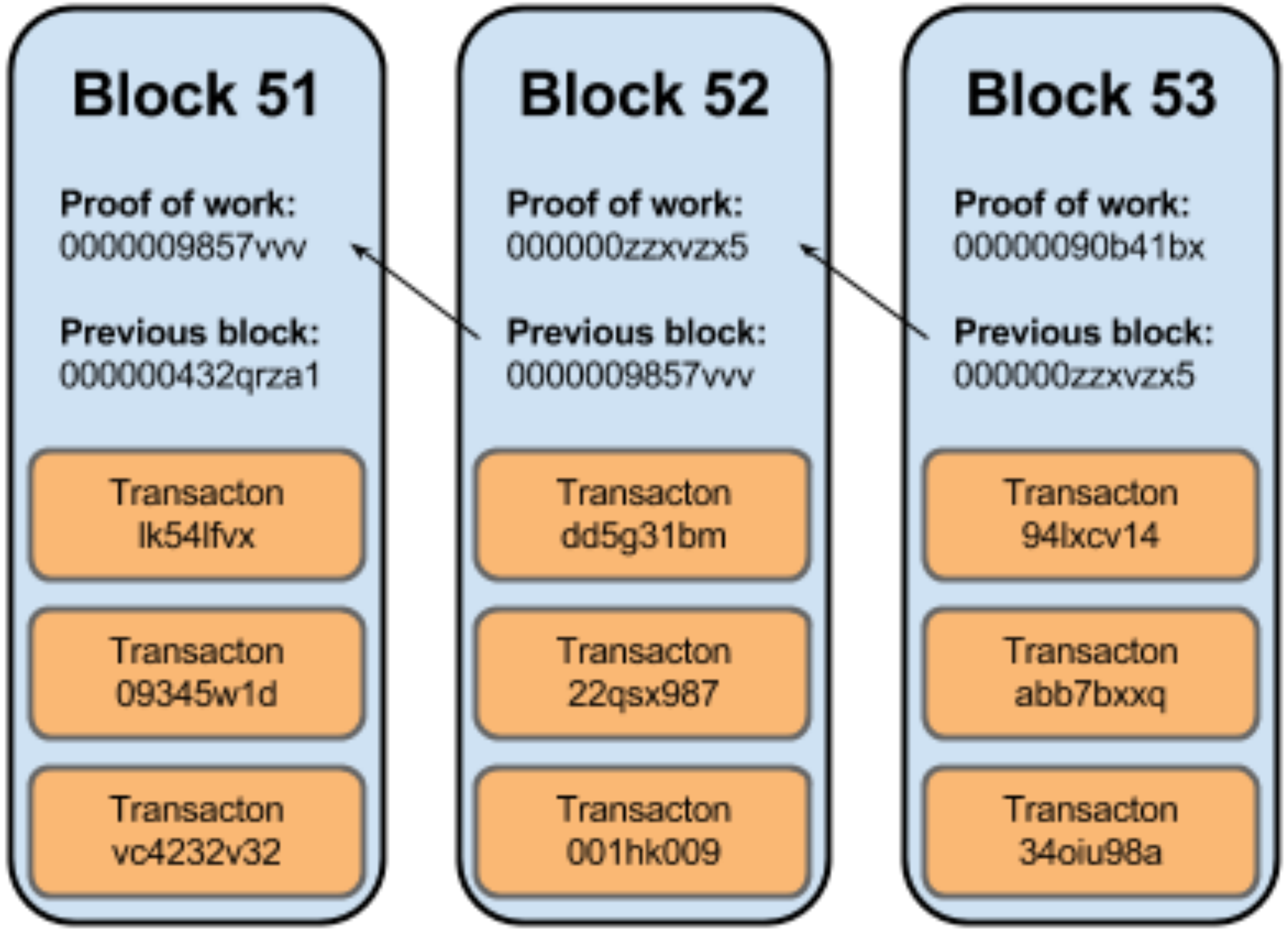
Outline

- **Ethereum 1.0 node**, 以太坊 1.0 节点
- **Scalability issue of Blockchain**, 区块链的可扩展性问题
- **Sharding**, 分片
- **What’s new?** 分片上的新设计

If I am an Ethereum 1.0 full node

以太坊 1.0 节点

Ethereum is a blockchain system



Block 51

Proof of work:
0000009857vvv

Previous block:
000000432qrza1

Transacton
lk54lfvx

Transacton
09345w1d

Transacton
vc4232v32

Block 52

Proof of work:
000000zzxvzx5

Previous block:
0000009857vvv

Transacton
dd5g31bm

Transacton
22qsx987

Transacton
001hk009

Block 53

Proof of work:
00000090b41bx

Previous block:
000000zzxvzx5

Transacton
94lxcv14

Transacton
abb7bxxq

Transacton
34oiu98a

P2P Network

- Receive / Broadcast transactions and blocks
  接收 / 广播交易与区块

- full sync / fast sync (geth) / warp sync (parity)

- Mainnet / Testnet (ROPSTEN, KOVAN, RINKEBY…)
  主链 / 测试链

Verification

- Execute EVM (Ethereum Virtual Machine) bytecode
  执行 EVM bytecode

State Transition

```
state_transition_function(state, block) → state′
```

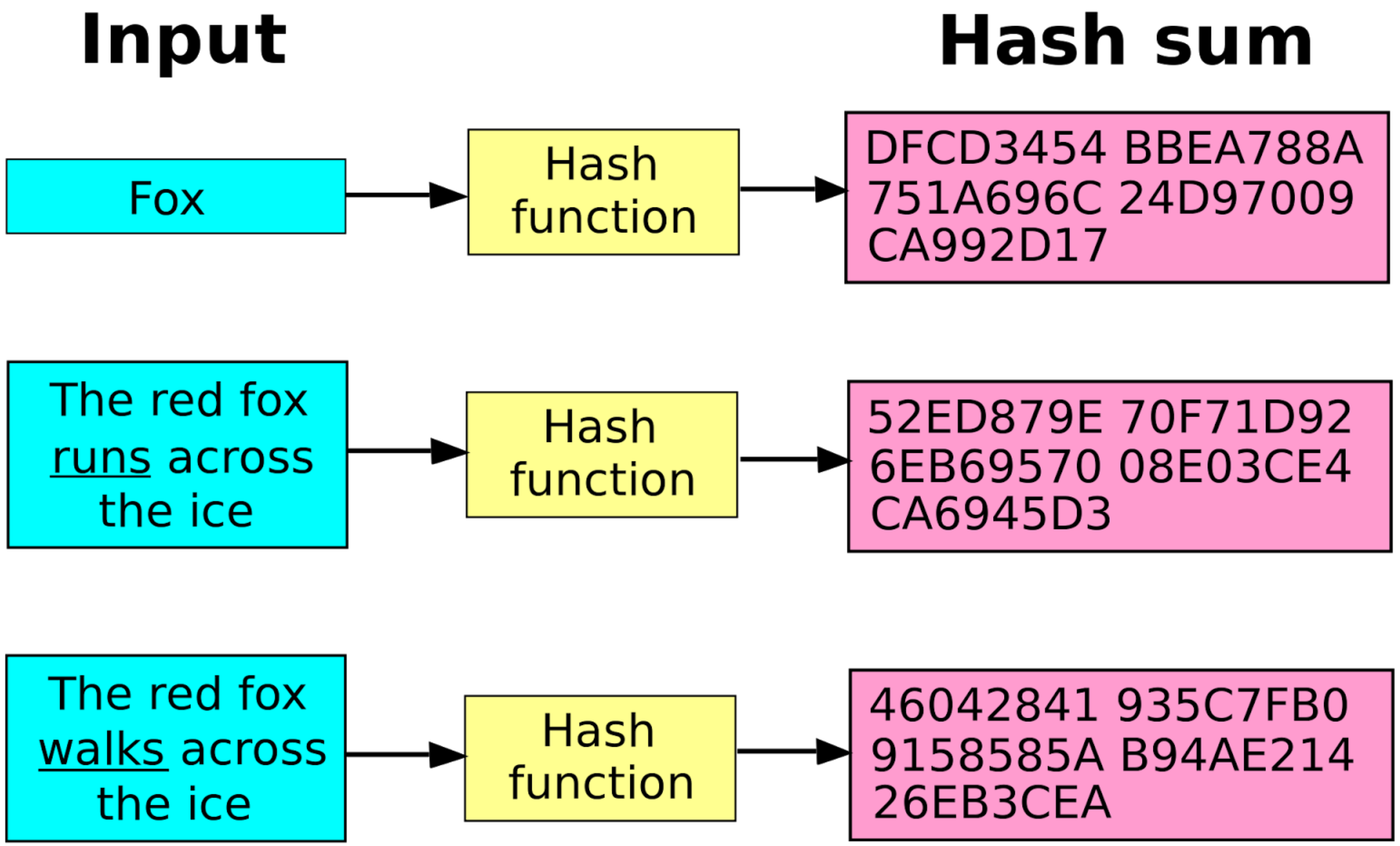
- Access the tx-related accounts
- Computation
- Update/Write the state

Verification

- Execute EVM (Ethereum Virtual Machine) bytecode
  执行 EVM bytecode

- Verify the **merkle proofs**
  验证 **merkle proofs**

Hash Function



Input

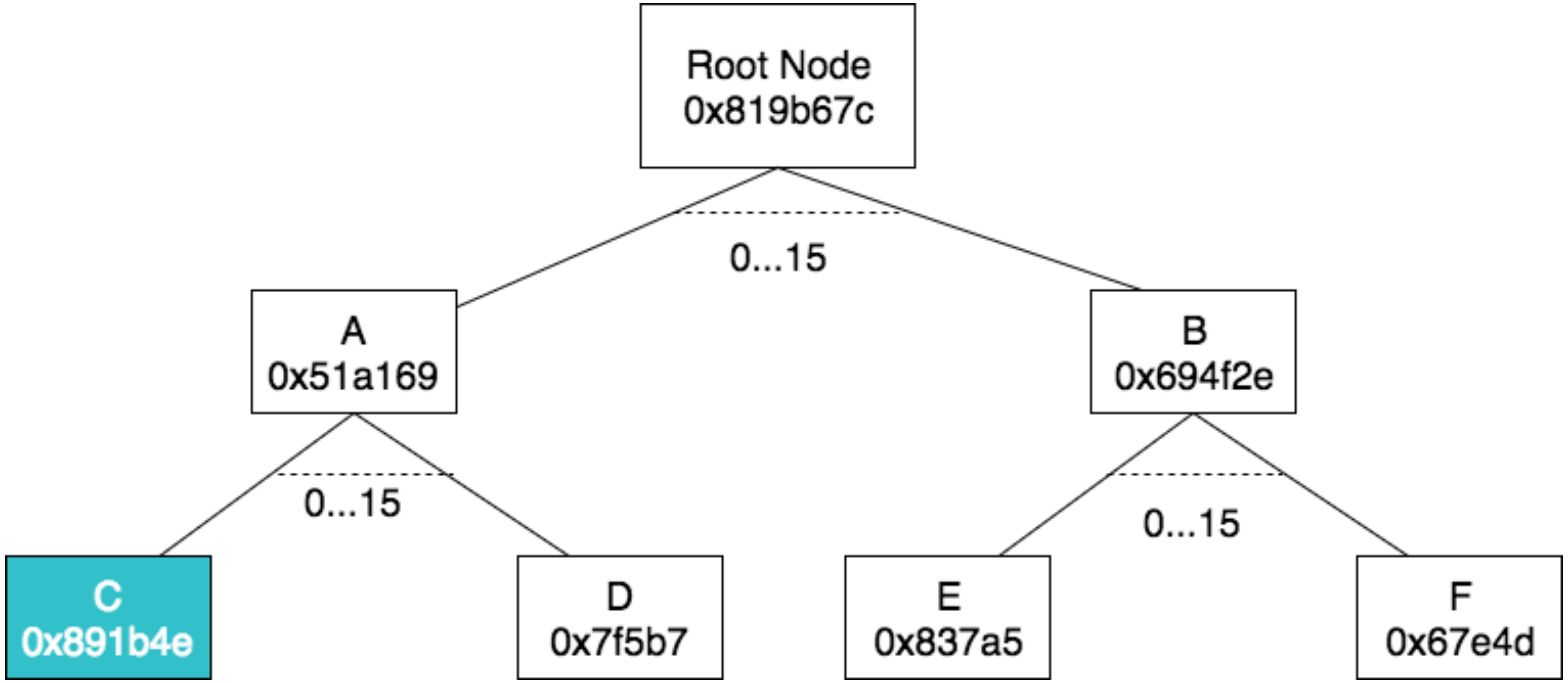
Hash sum

Fox → Hash function → DFCD3454 BBEA788A 751A696C 24D97009 CA992D17

The red fox runs across the ice → Hash function → 52ED879E 70F71D92 6EB69570 08E03CE4 CA6945D3

The red fox walks across the ice → Hash function → 46042841 935C7FB0 9158585A B94AE214 26EB3CEA

State Trie and Merkle Proof



balance + nonce + codehash + storage

if eth_mining

- Collect transactions from tx mempool
  从交易池中选出交易
- Execute EVM (Ethereum Virtual Machine) code
  执行 EVM bytecode
- Create merkle proofs
  建立 merkle proofs
- Run Ethash PoW algorithm
  运行 Ethash 工作量证明演算法

Scalability Issues

可扩展性问题

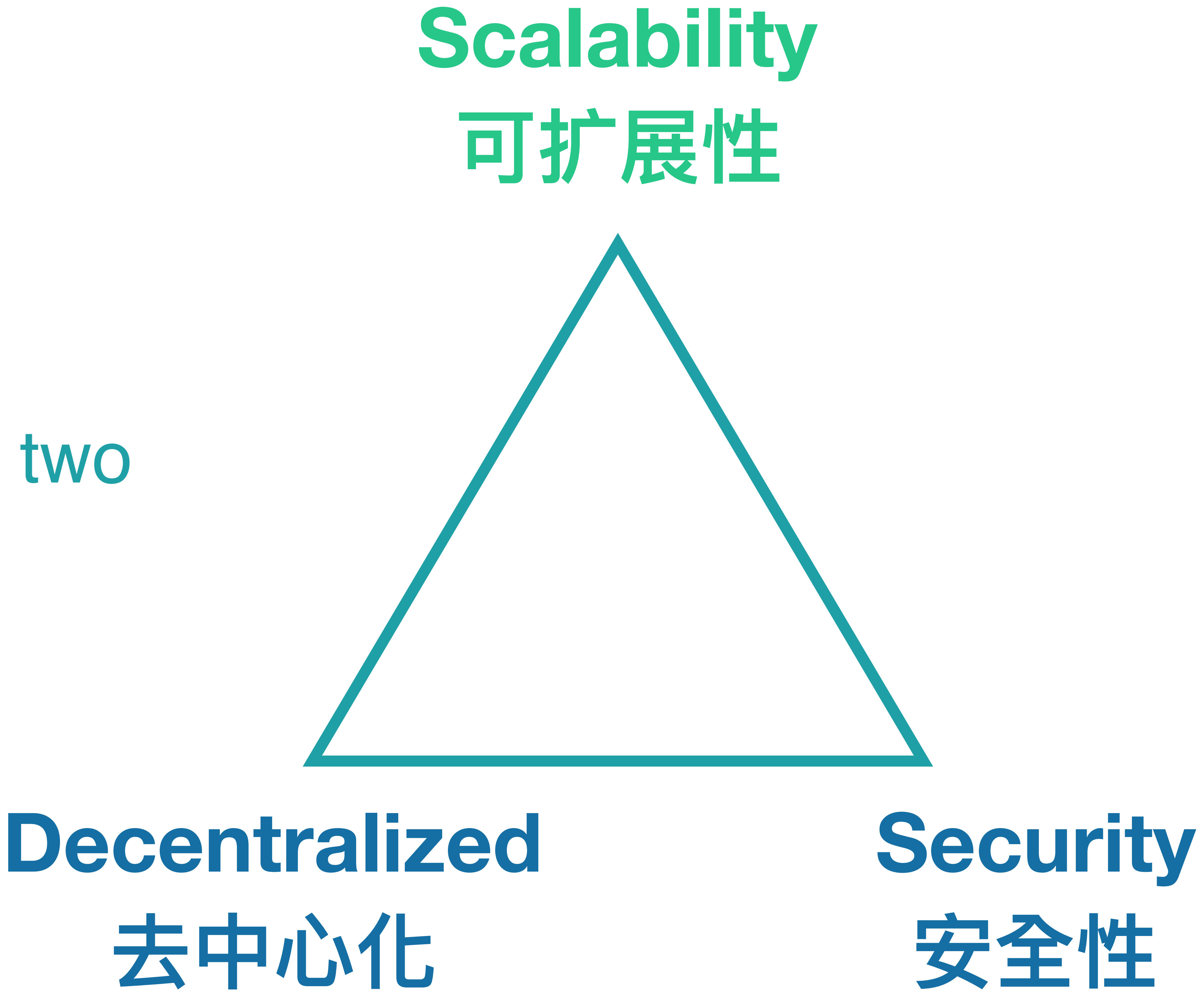
Scalability Issues

- Every full node executes each transaction and store the whole (or pruned) state trie for security and decentralized
  为了安全性與去中心化，每个全节点都执行每一笔交易，并储存整个 (或修整过的) state trie

- Parallelizability of EVM execution
  EVM 的平行化执行

Blockchain Trilemma

> “ blockchain systems can only at most have two of the following three properties
>
> **- Vitalik Buterin, Sharding FAQ**
> https://github.com/ethereum/wiki/wiki/Sharding-FAQ ”



Solutions

- State channels
  状态通道

- Plasma chain
  Plasma 链

- Interactive verification for scalable computation
  交互式验证

Solutions

- State channels
  状态通道
- Plasma chain
  Plasma 链
- Interactive verification for scalable computation
  交互式验证
- **Sharding**
  分片

Sharding

The brand new chains!

Sharding in Blockchain

- Create many new shard chains
  创建许多的新的分片链

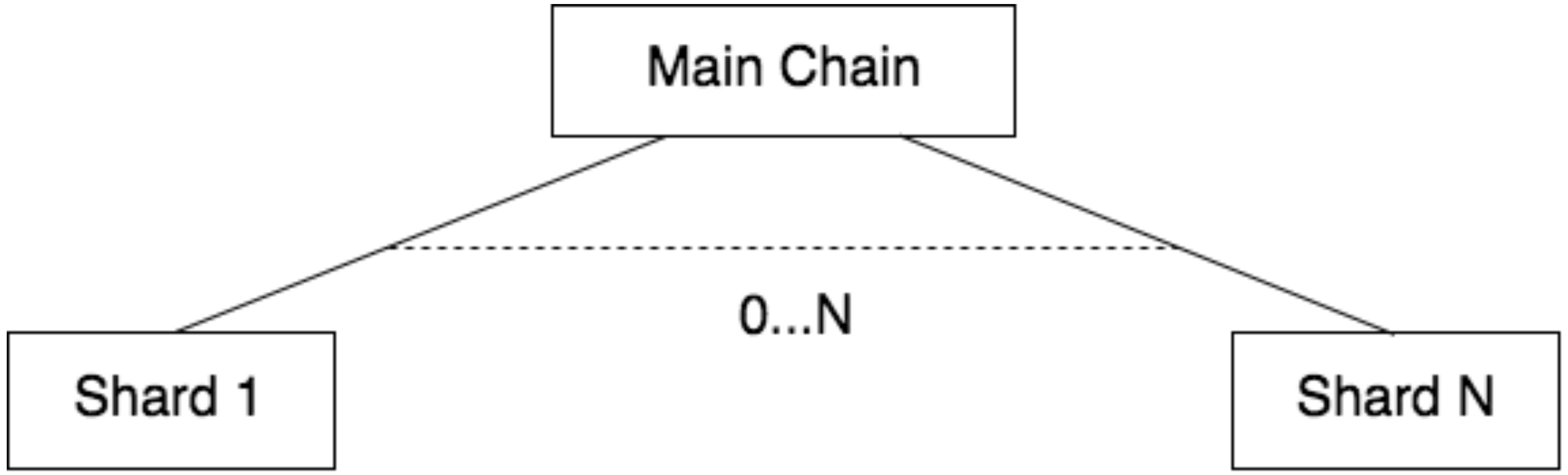
- Each shard chain is a new galaxy
  每个分片都是一个新的小星系

- The fork choice rule of shard chain is based on main chain (Ethereum Mainnet)
  分片上的分岔选择规则是根据主链上的分岔状况

Main Chain <—> Shard Chain

Main Chain	Shard Chain
Block BlockHeader	**Collation** CollationHeader
Block Proposer (or Miner in PoW chain)	Collator
Ethash (PoW) Casper (PoS)	Via **validator manager contract on main chain**

Basic Sharding - Quadratic 二次分片



Basic Sharding - Tracking on Main Chain

Validator Manager Contract

- `deposit`
- `withdraw`
- `get_eligible_proposer / sample`
- `add_header`

https://github.com/ethereum/sharding/blob/develop/sharding/contracts/validator_manager.v.py

Basic Sharding

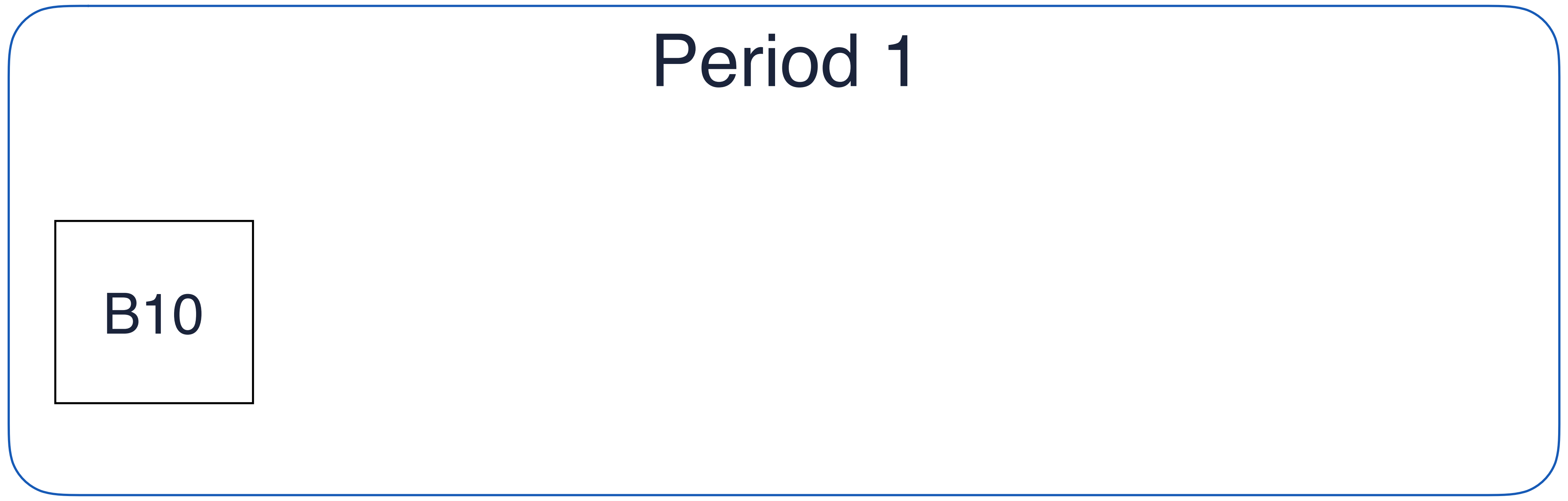
Main chain

Shard 1

Shard 2

Basic Sharding

Main chain



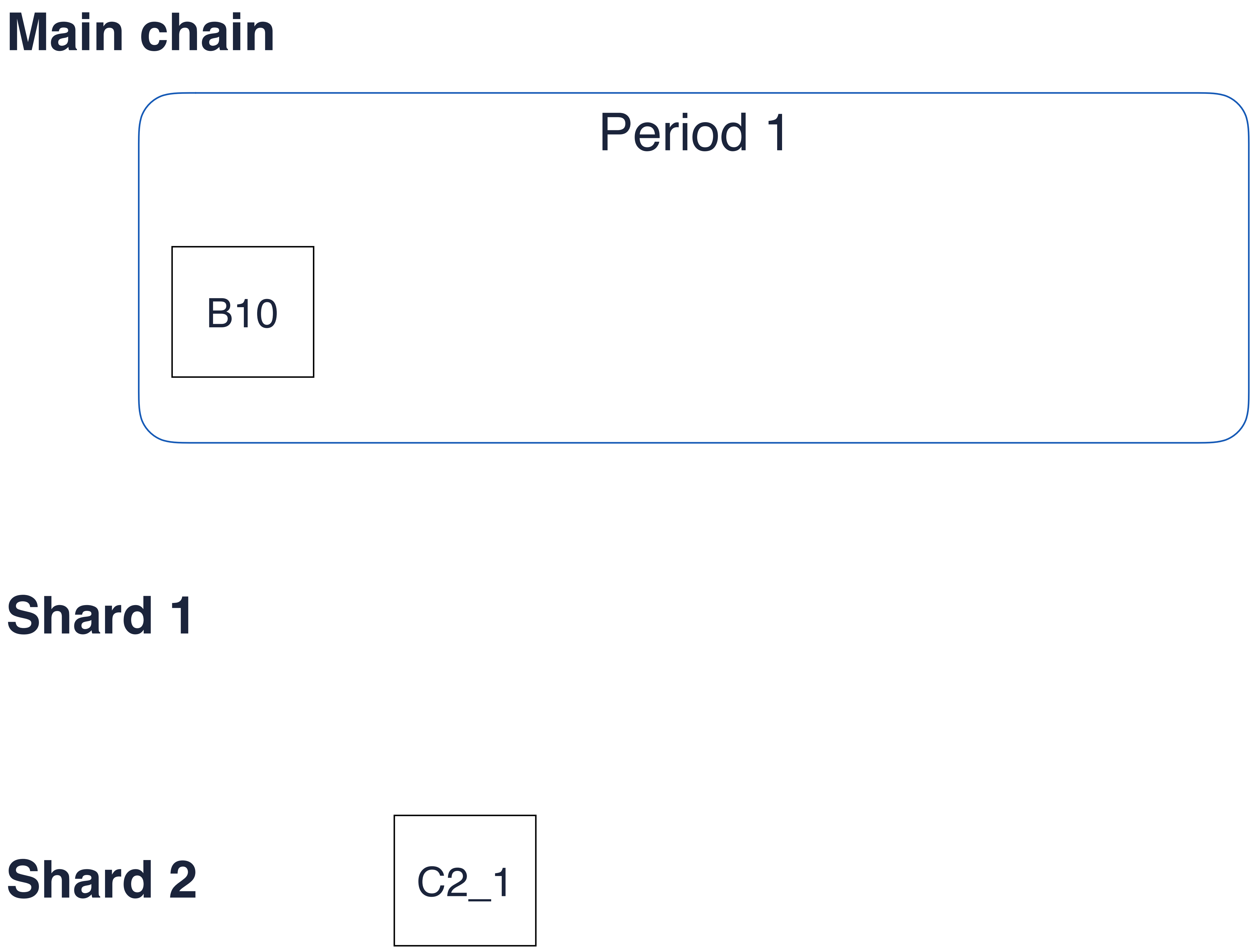
Period 1

B10

Shard 1

Shard 2

Basic Sharding



Main chain

Period 1

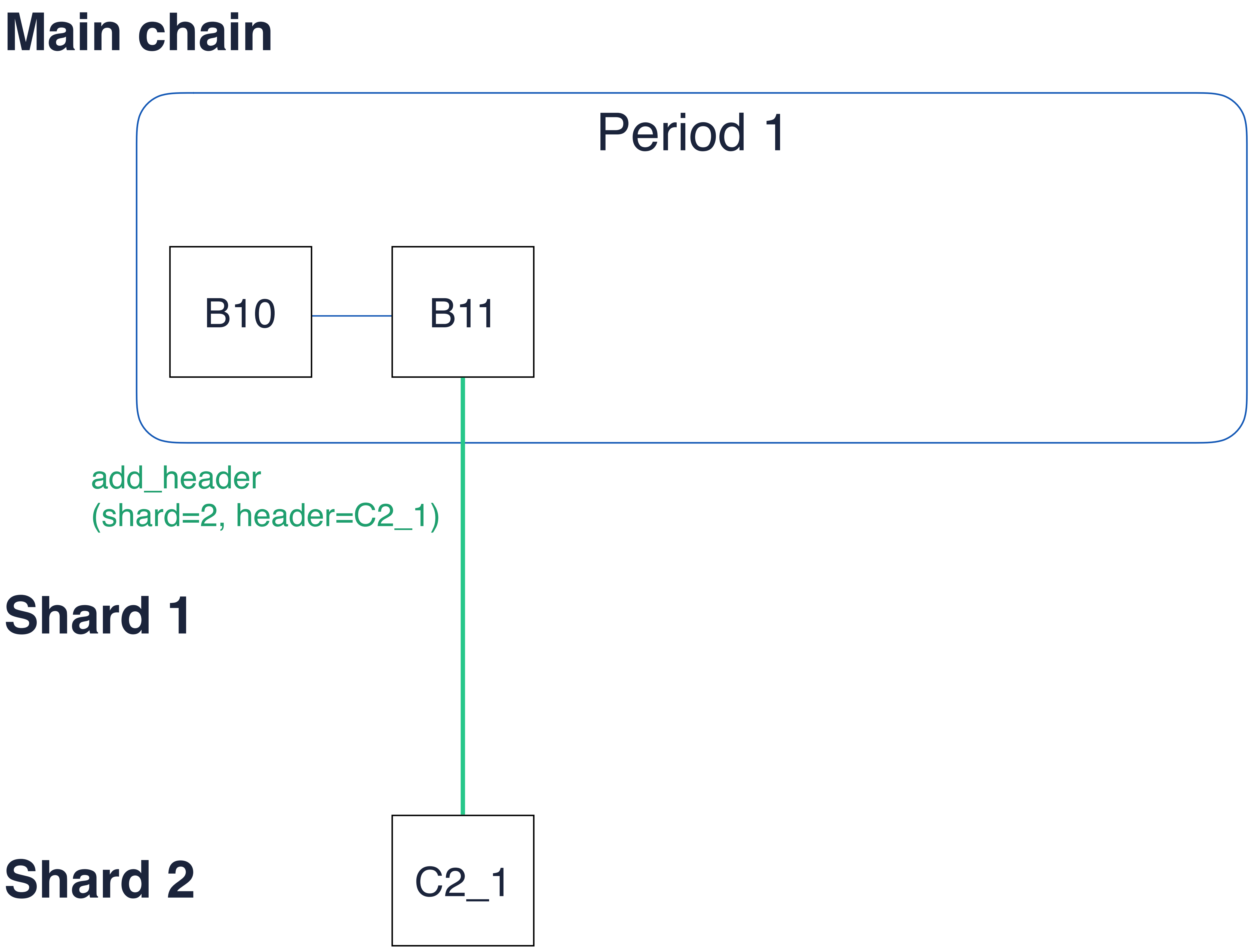
B10

Shard 1

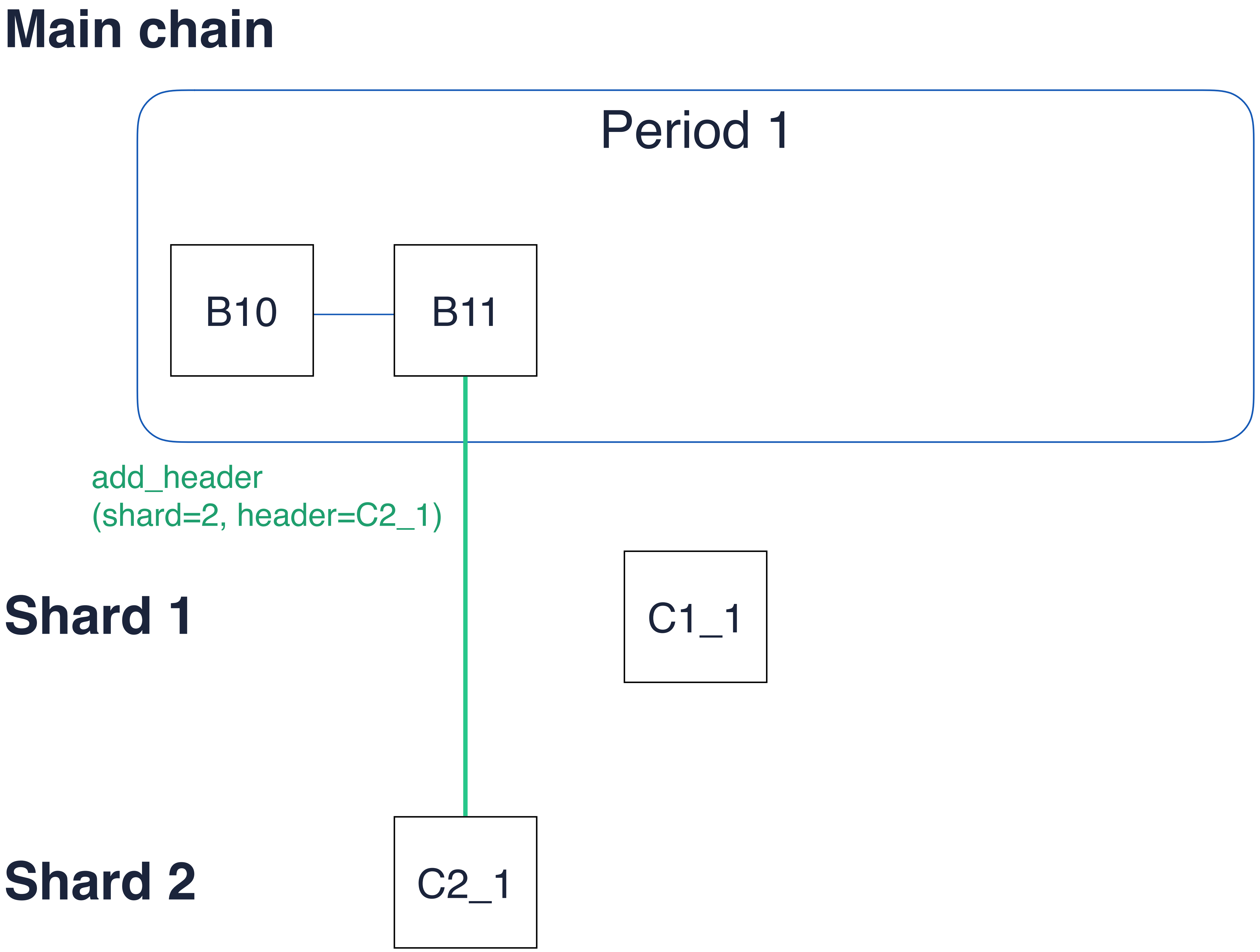
Shard 2

C2_1

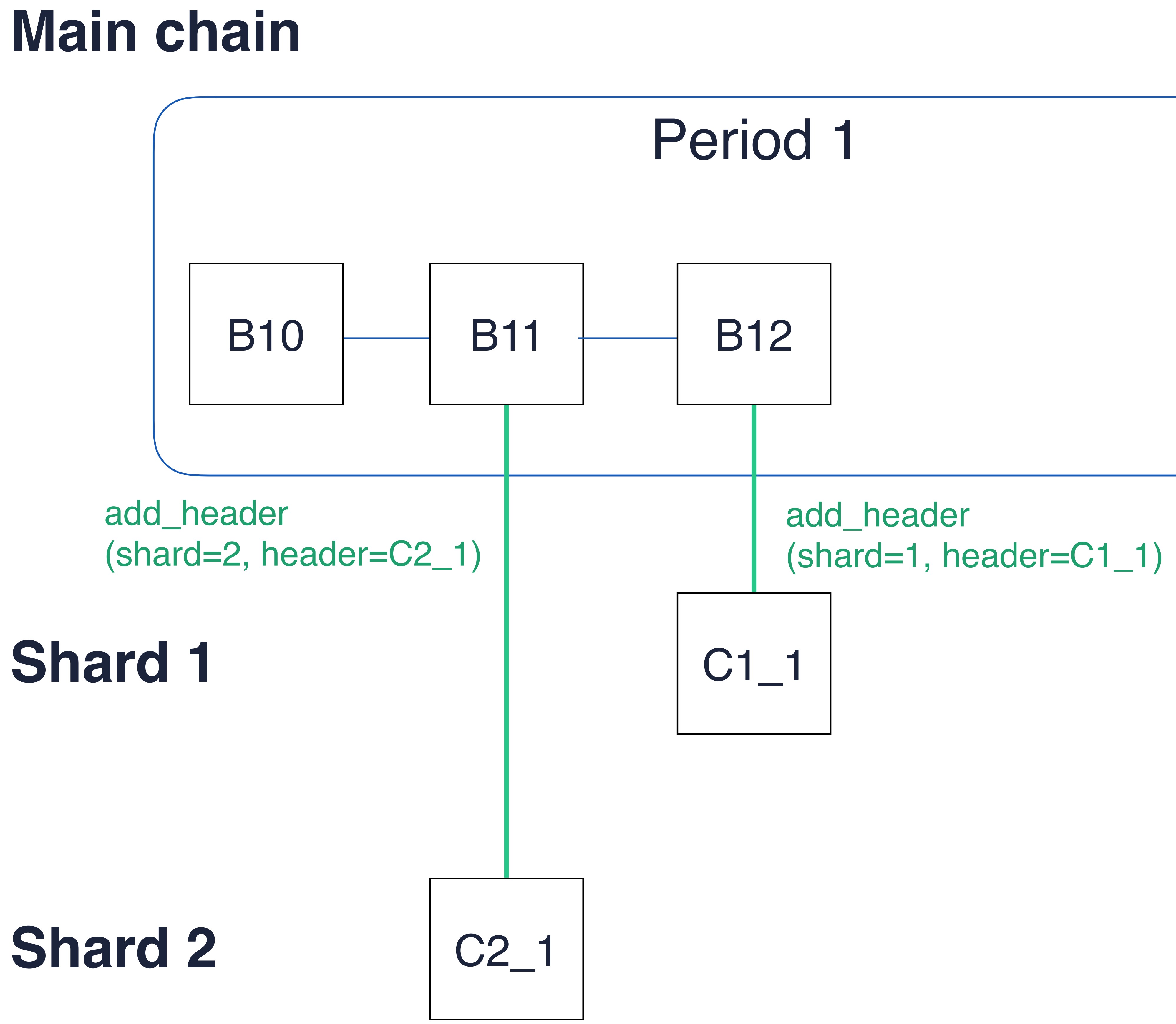
Basic Sharding



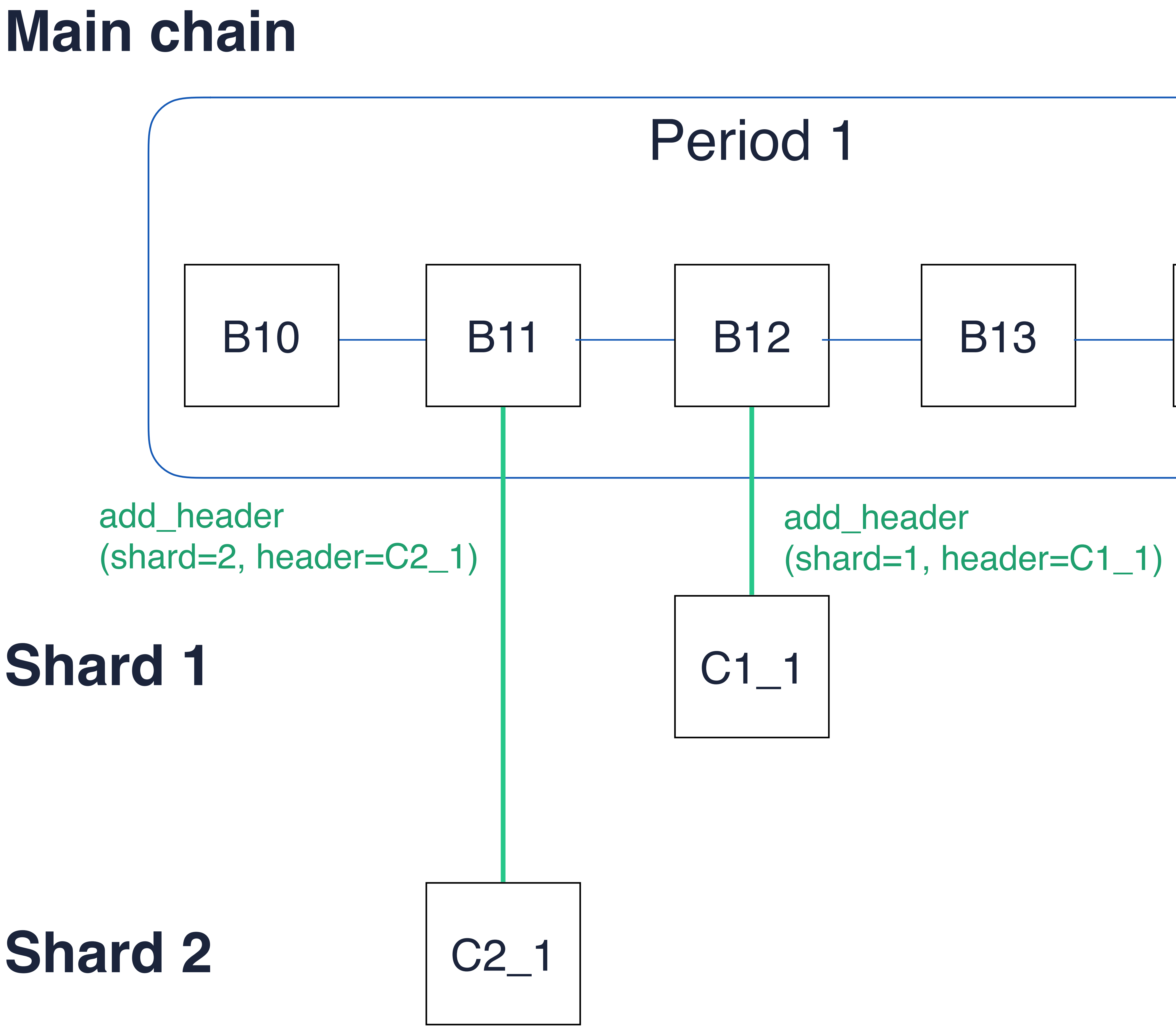
Basic Sharding



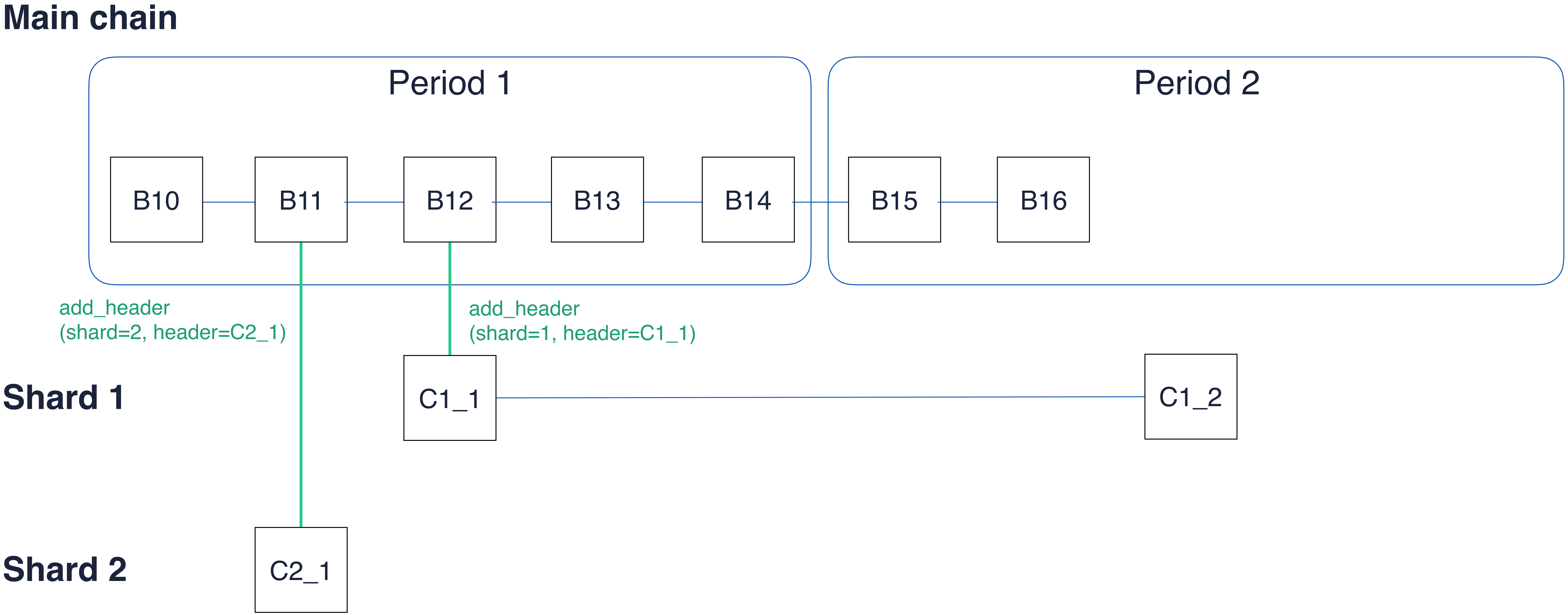
Basic Sharding



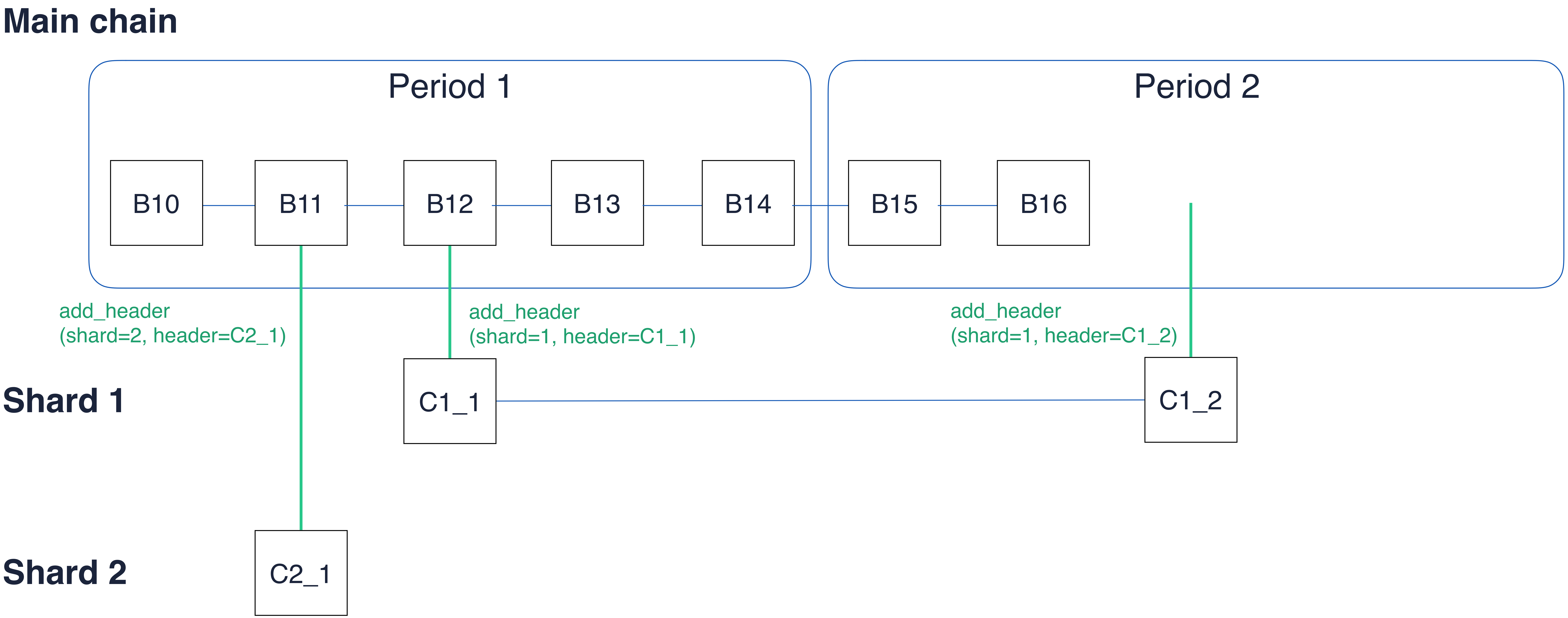
Basic Sharding



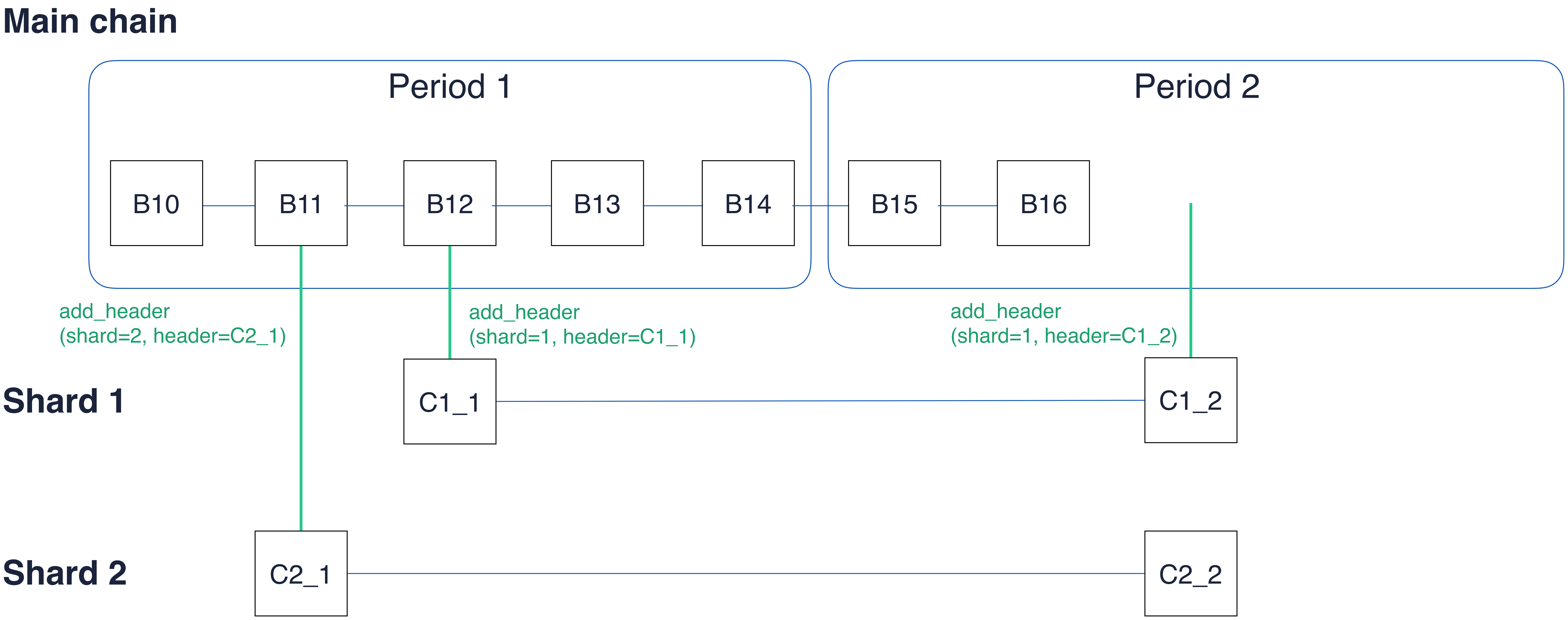
Basic Sharding



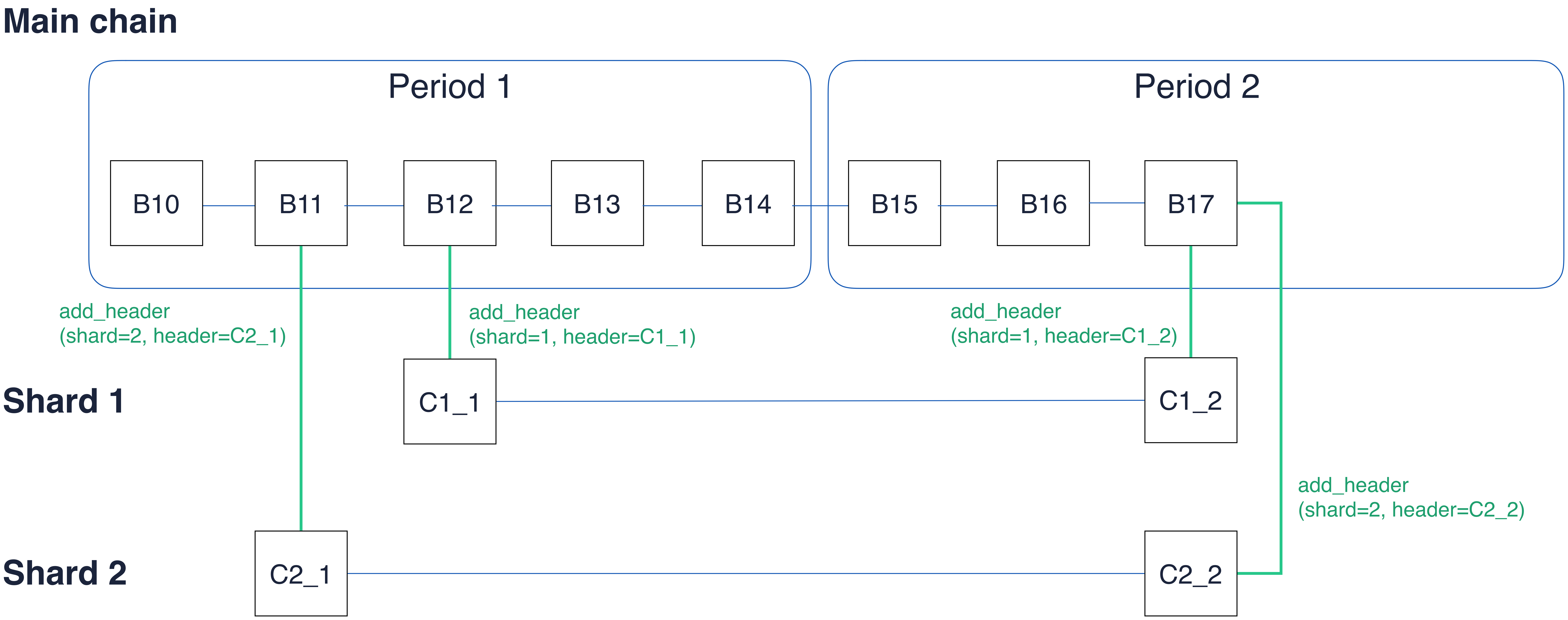
Basic Sharding



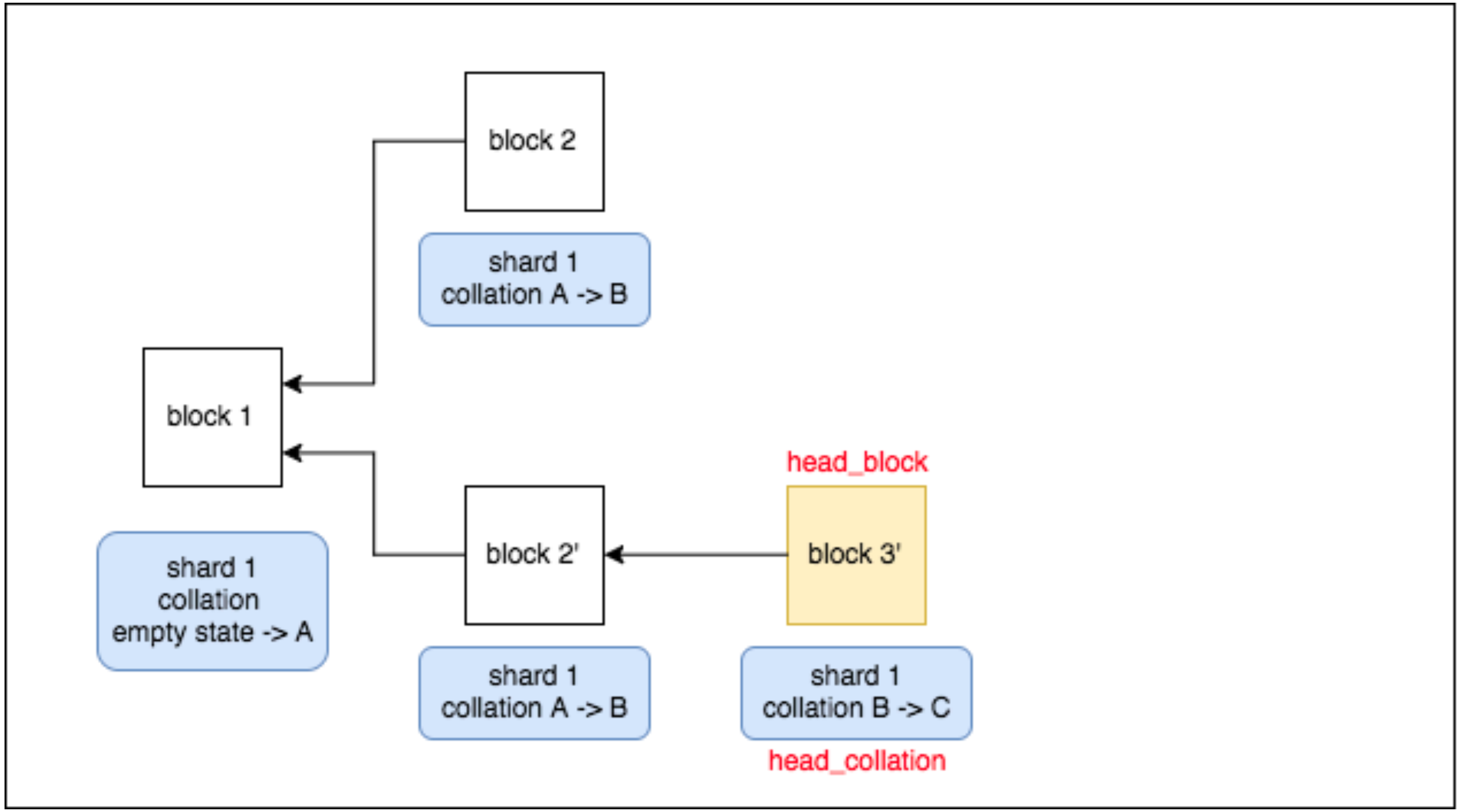
Basic Sharding



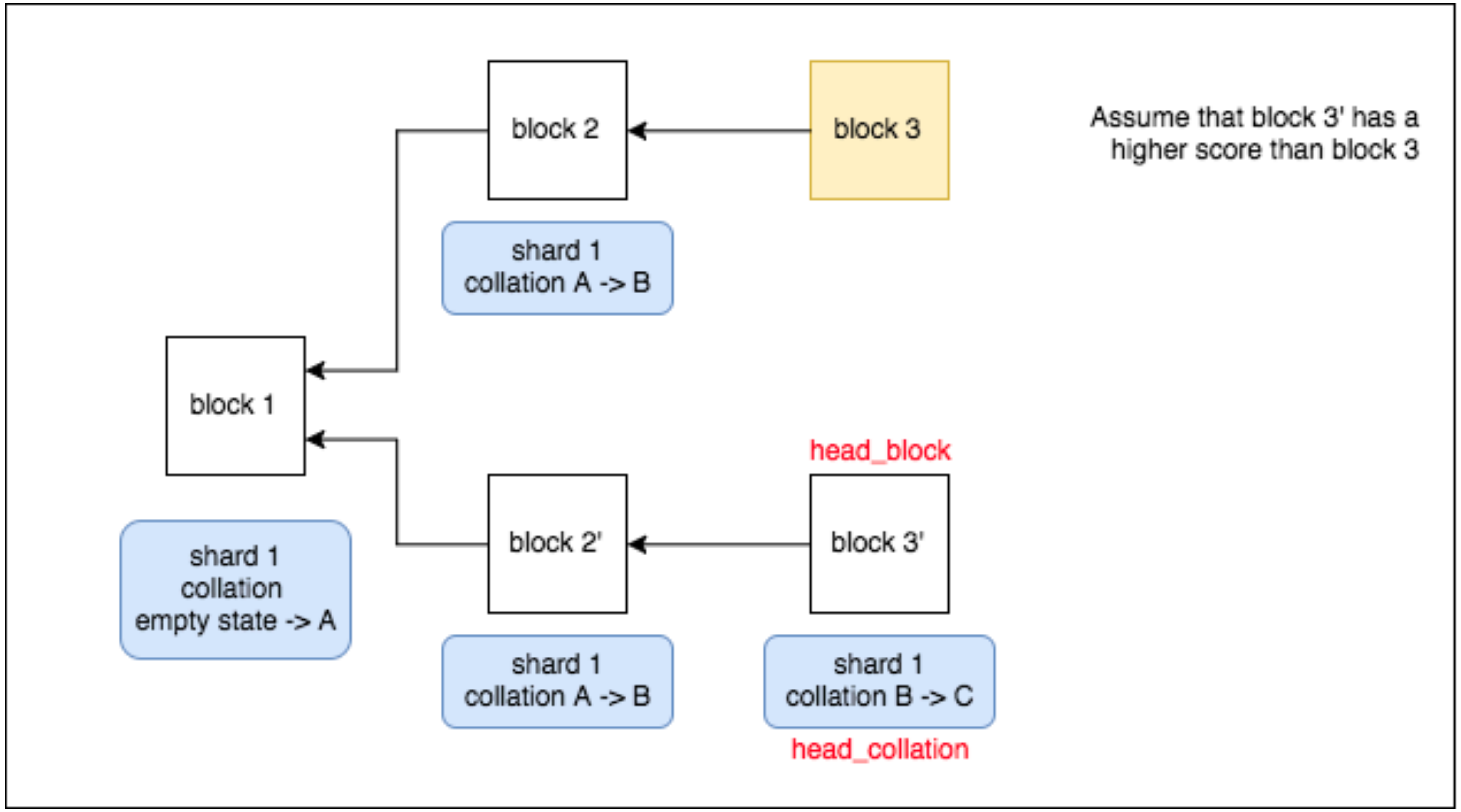
Basic Sharding



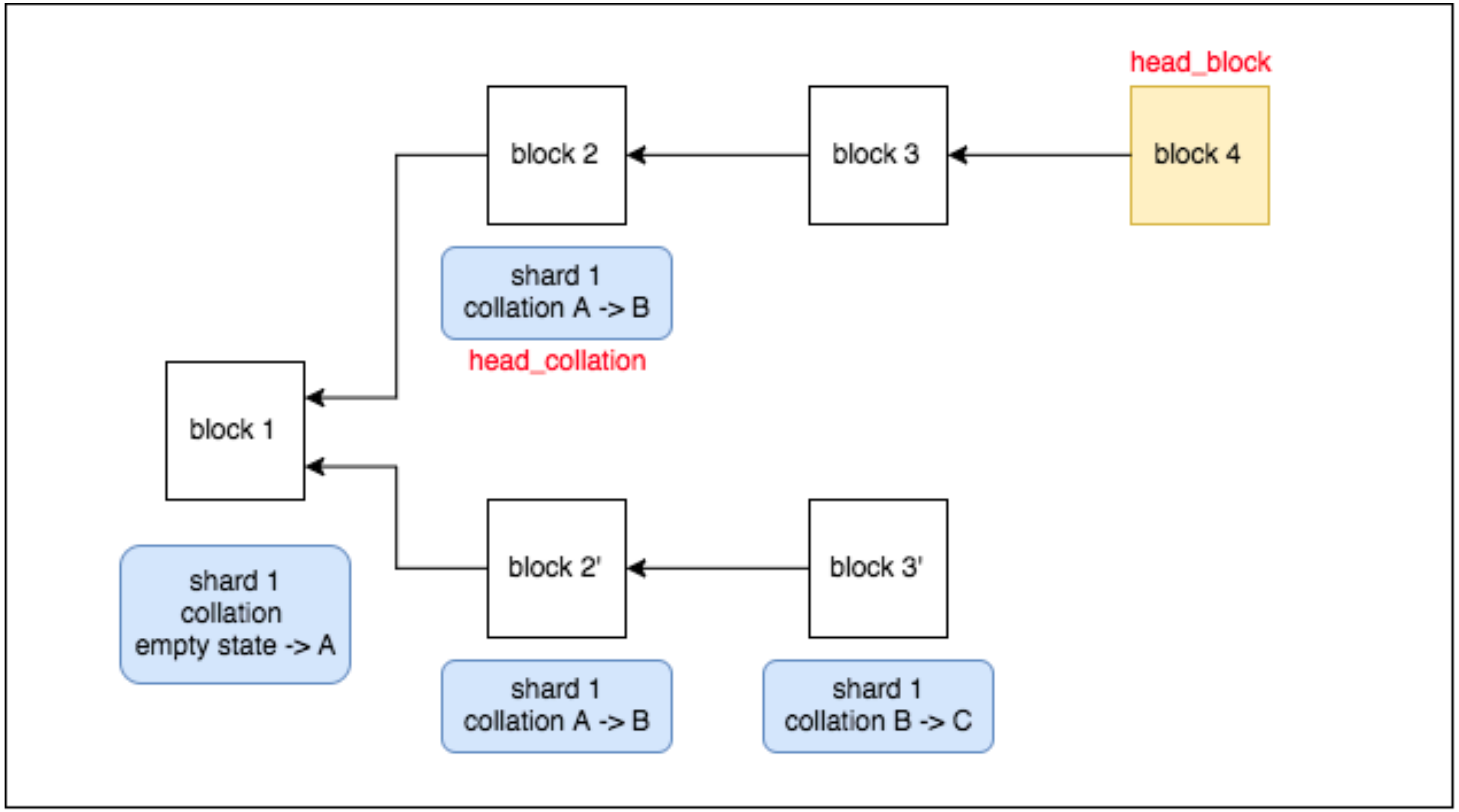
Basic Sharding - Fork Choice Rule



Basic Sharding - Fork Choice Rule

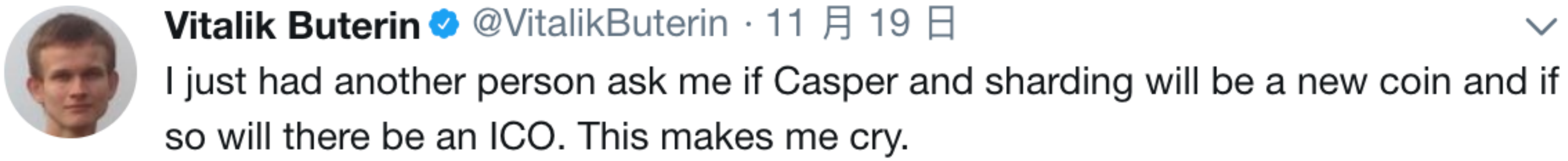


Basic Sharding - Fork Choice Rule



ICO?

“There’s NO ShardCoin ICO!”
没有 ShardCoin ICO!



Vitalik Buterin @VitalikButerin · 11 月 19 日

I just had another person ask me if Casper and sharding will be a new coin and if so will there be an ICO. This makes me cry.

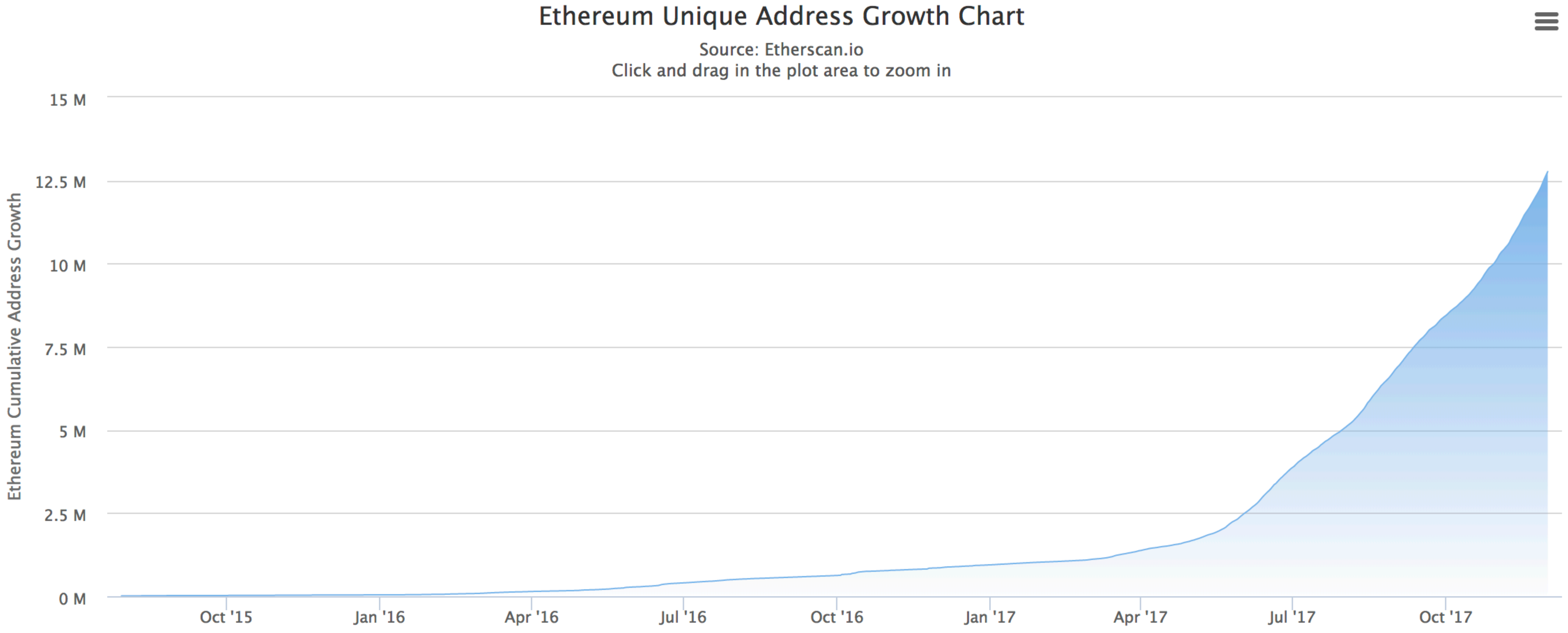
We can try something new design
in the new shards!

我们可以在新的分片上
尝试一些新的设计

Make the client “stateless”

无状态客户端

Unique Address Growth Chart



Ethereum Unique Address Growth Chart

Source: Etherscan.io

Click and drag in the plot area to zoom in

Some Numbers

~12.7 Millions

Distinct Addresses

~104,123

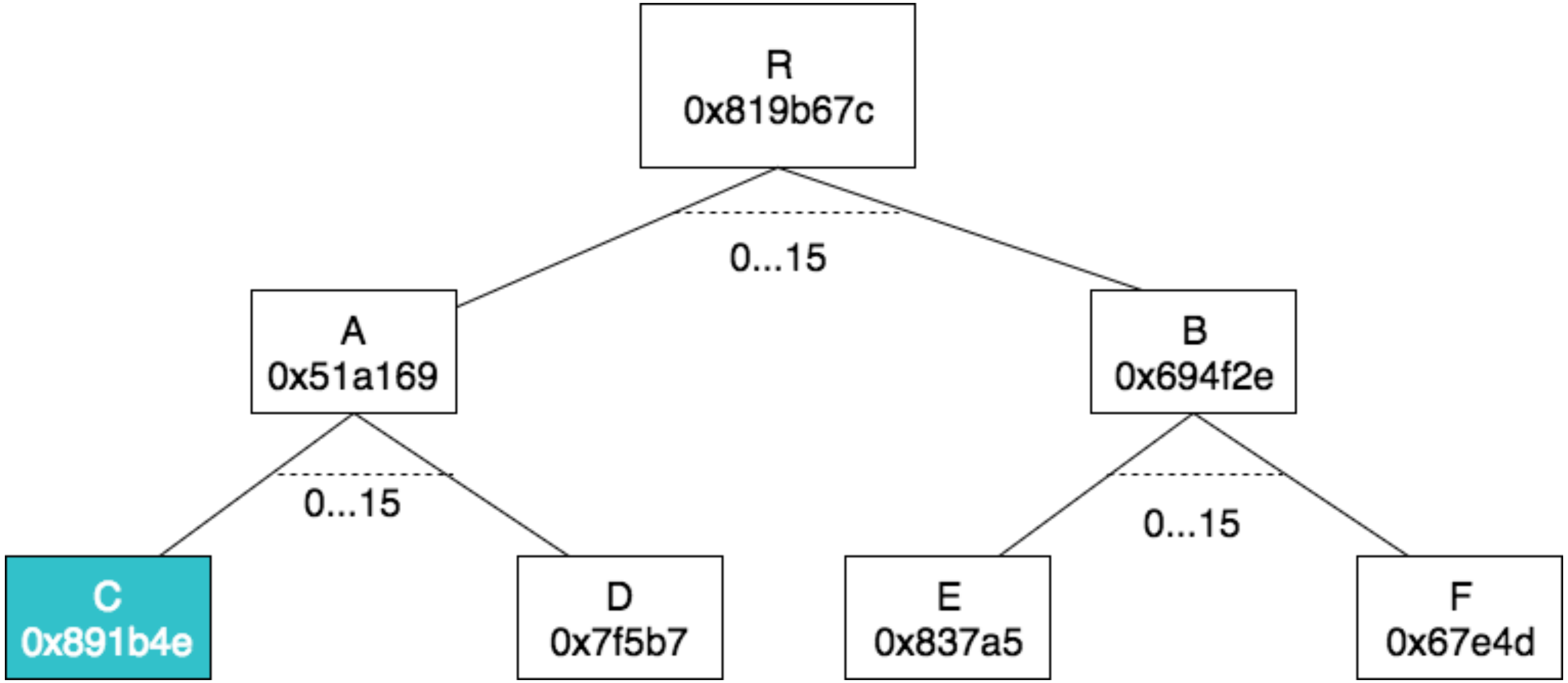
New Address/Day

30.8 GB

Geth w/ FAST Sync

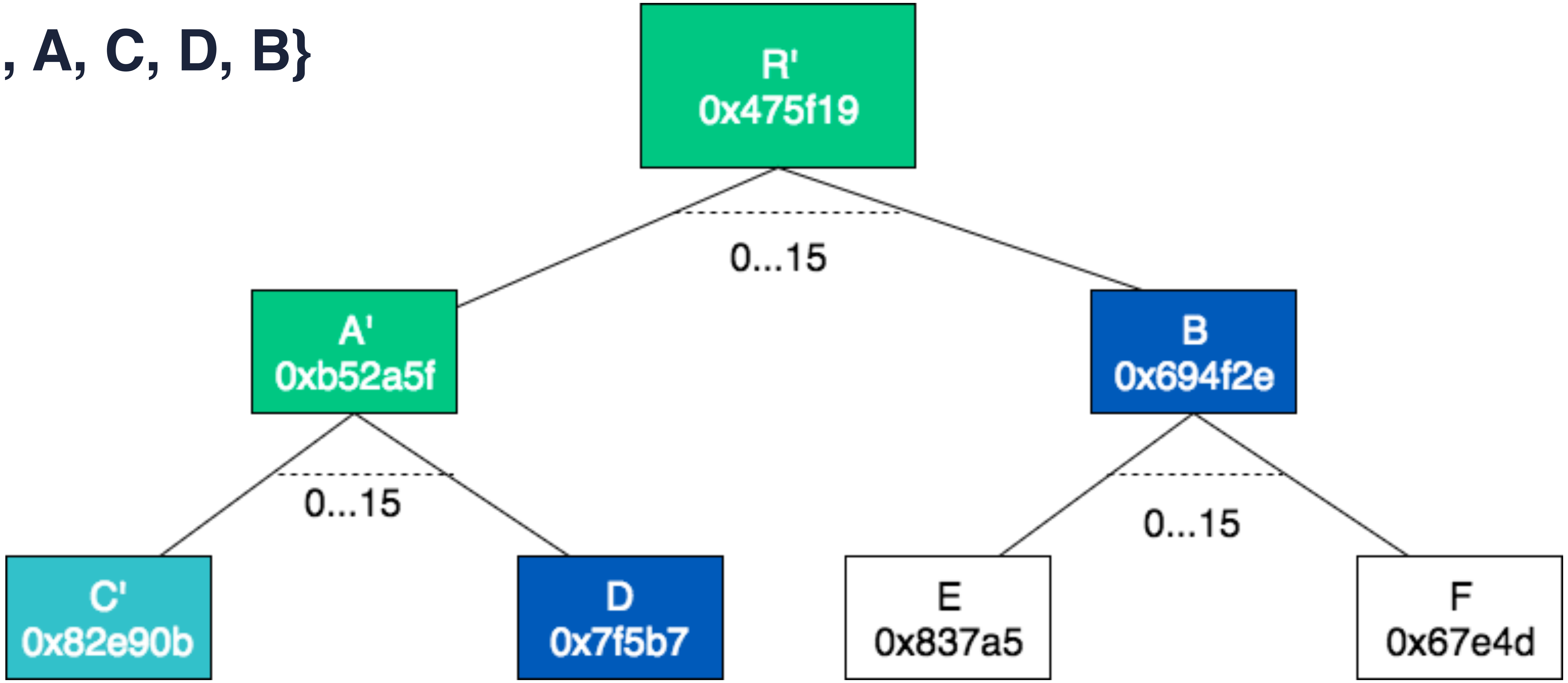
Source: etherscan.io
Dec 1st, 2017

Pre-state


R
0x819b67c
0...15
A
0x51a169
B
0x694f2e
0...15
0...15
C
0x891b4e
D
0x7f5b7
E
0x837a5
F
0x67e4d

Post-state

Witness: {R, A, C, D, B}



State Transition

```
state_transition_function(state_root, collation, witness)
        → state_root′, read_set, write_set
```

- Senders provide transaction witness
  送出交易者提供 transaction witness

- Archival node provide collation witness
  全状态节点提供 collation witness

- Stateless full node only have to store state roots
  无状态全节点只需存 state roots

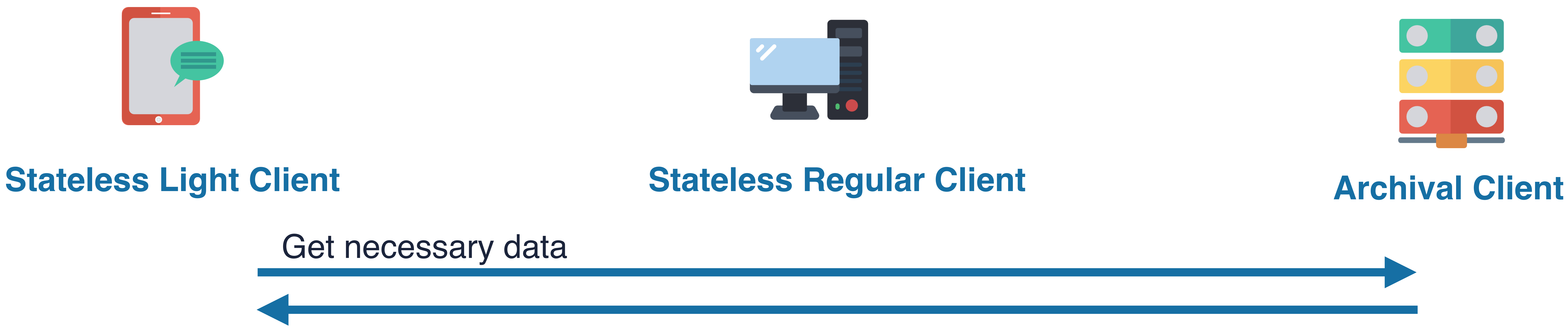
Basic Sharding - A possible scenario

Stateless Light Client

Stateless Regular Client

Archival Client

Basic Sharding -
A possible scenario



Stateless Light Client

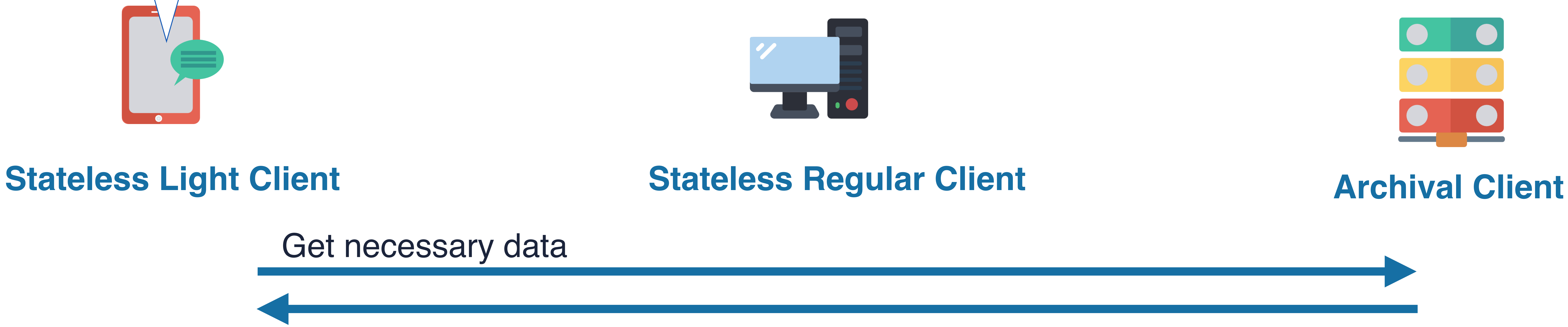
Stateless Regular Client

Archival Client

Get necessary data

Basic Sharding - A possible scenario

```
tx = [
    version_num,
    chain_id,
    shard_id,
    account,
    gas,
    data
]
```



Stateless Light Client

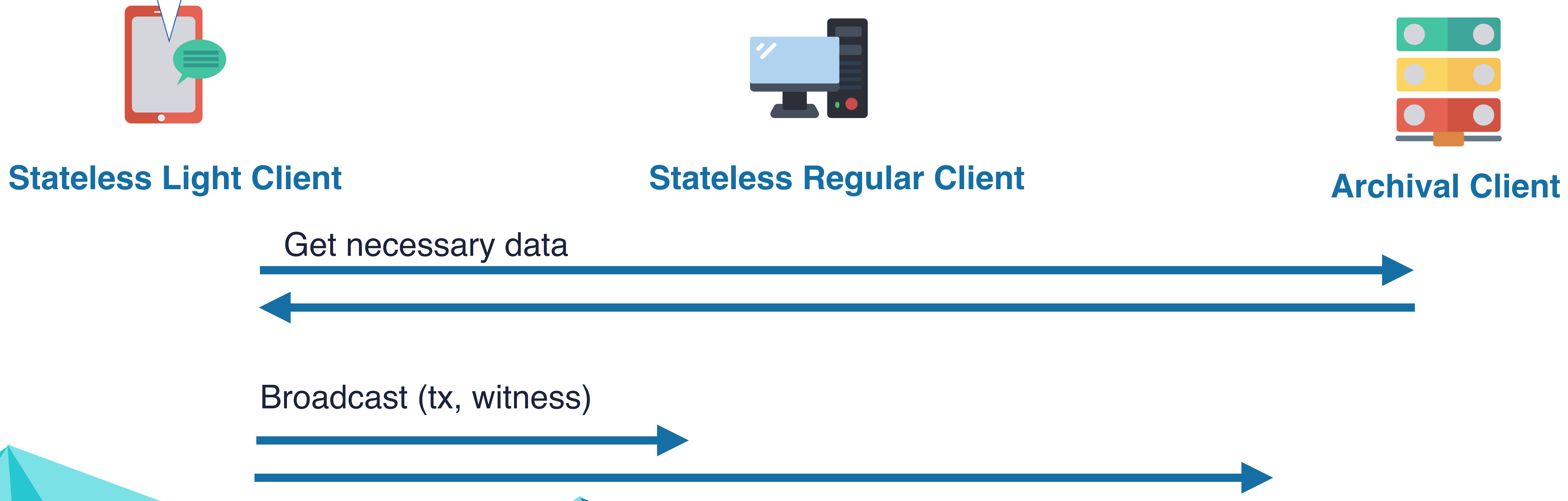
Stateless Regular Client

Archival Client

Get necessary data

Basic Sharding - A possible scenario

```
tx = [
    version_num,
    chain_id,
    shard_id,
    account,
    gas,
    data
]
```



Stateless Light Client

Stateless Regular Client

Archival Client

Get necessary data

Broadcast (tx, witness)

Basic Sharding - Create a collation

Archival Client

Stateless Client Validator

Basic Sharding - Create a collation

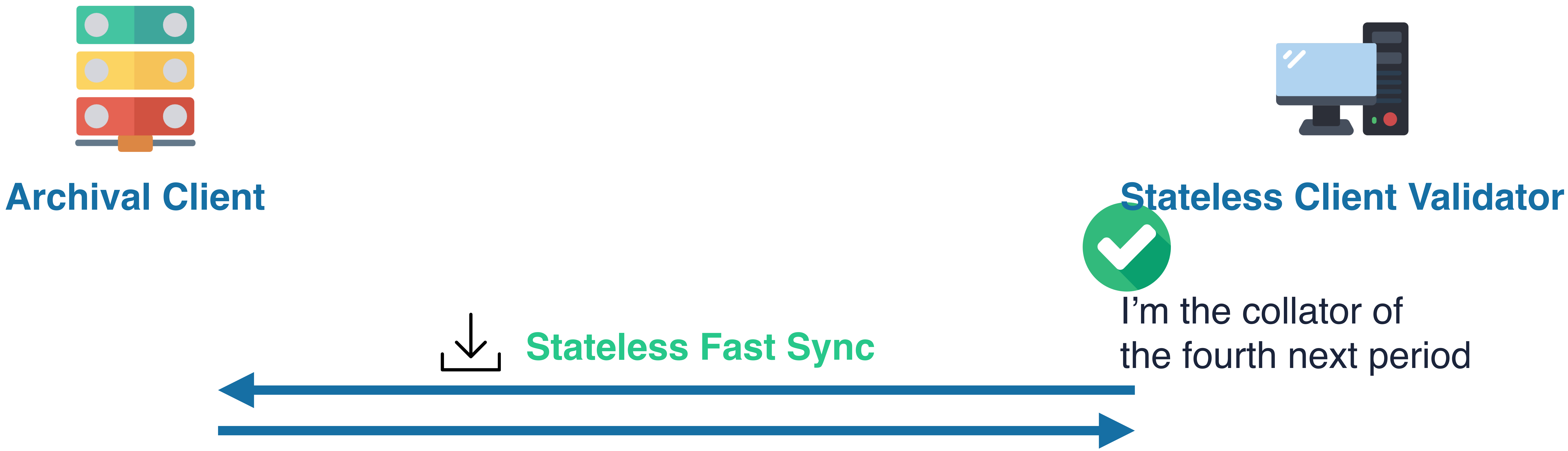
Archival Client

Stateless Client Validator

✓

I’m the collator of
the fourth next period

Basic Sharding - Create a collation



Archival Client

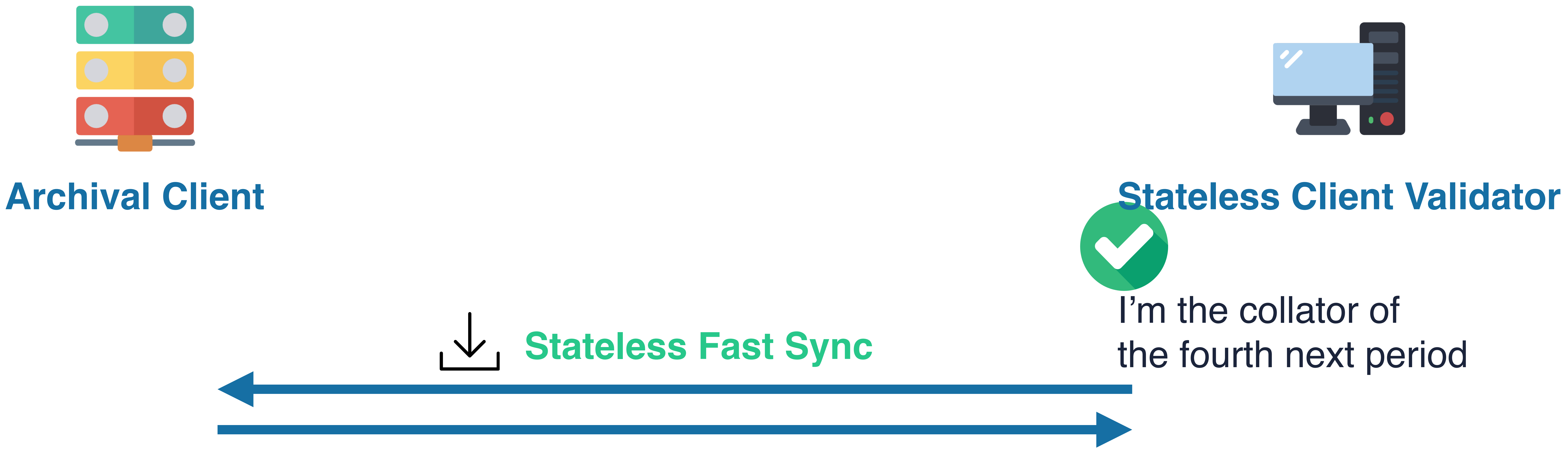
Stateless Client Validator

✓

I’m the collator of
the fourth next period

Stateless Fast Sync

Basic Sharding - Create a collation



Archival Client

Stateless Client Validator

✓

I’m the collator of
the fourth next period

Stateless Fast Sync

Create a collation!

Research Topics for Optimization

- Stateless client
- Account redesign
- Account abstraction
- Binary state trie
- Parallelizability
- EVM 2.0
- ….etc.

Research Topics of Hard Problems

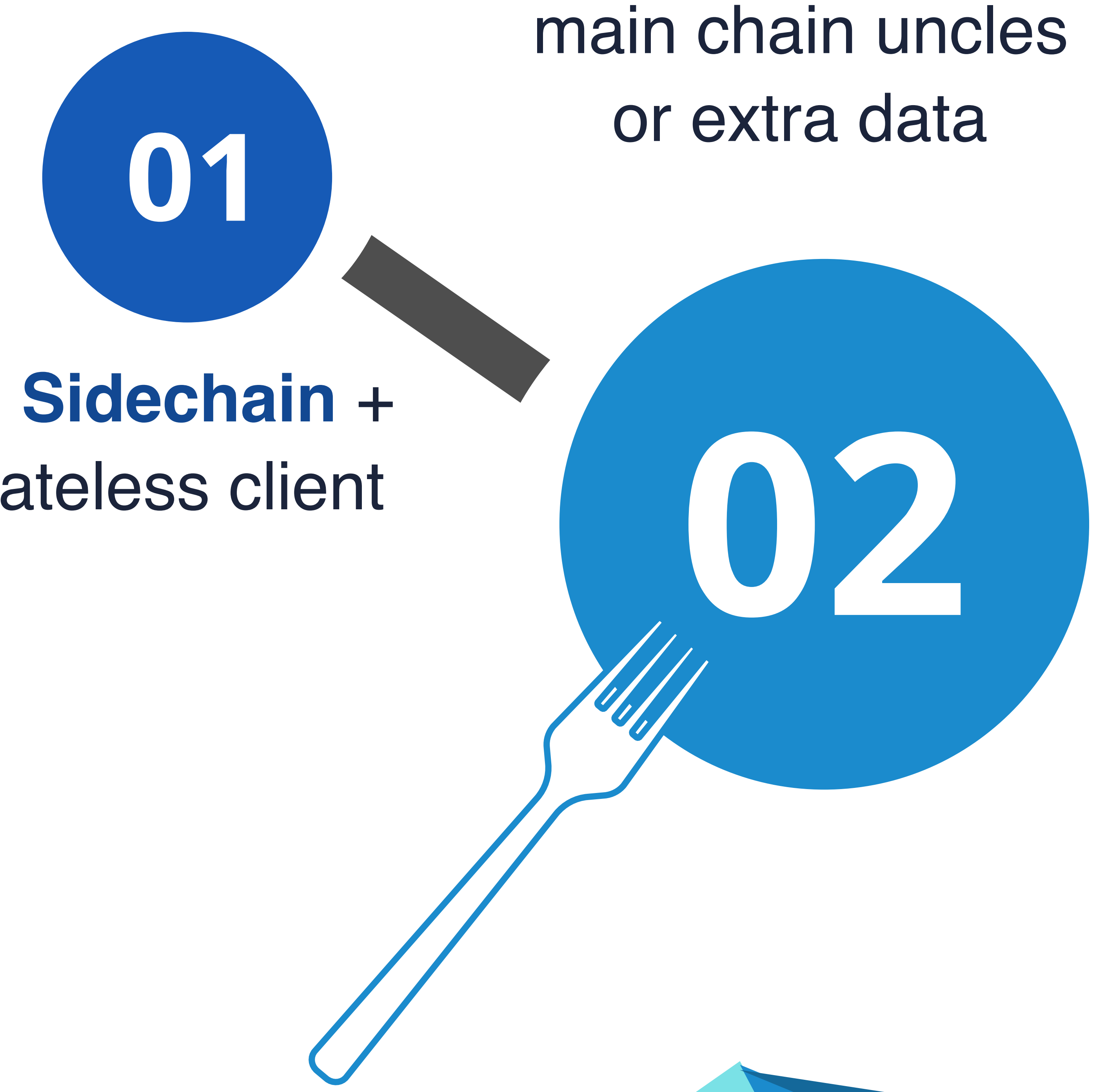
- Data availability
- Guaranteed scheduled call (atomic transaction)
- 1% attack problem
- Censorship resistance
- Partition state
- Cryptoeconomics
- ….etc.

Roadmap

01

L2 Sidechain +
Stateless client

Roadmap



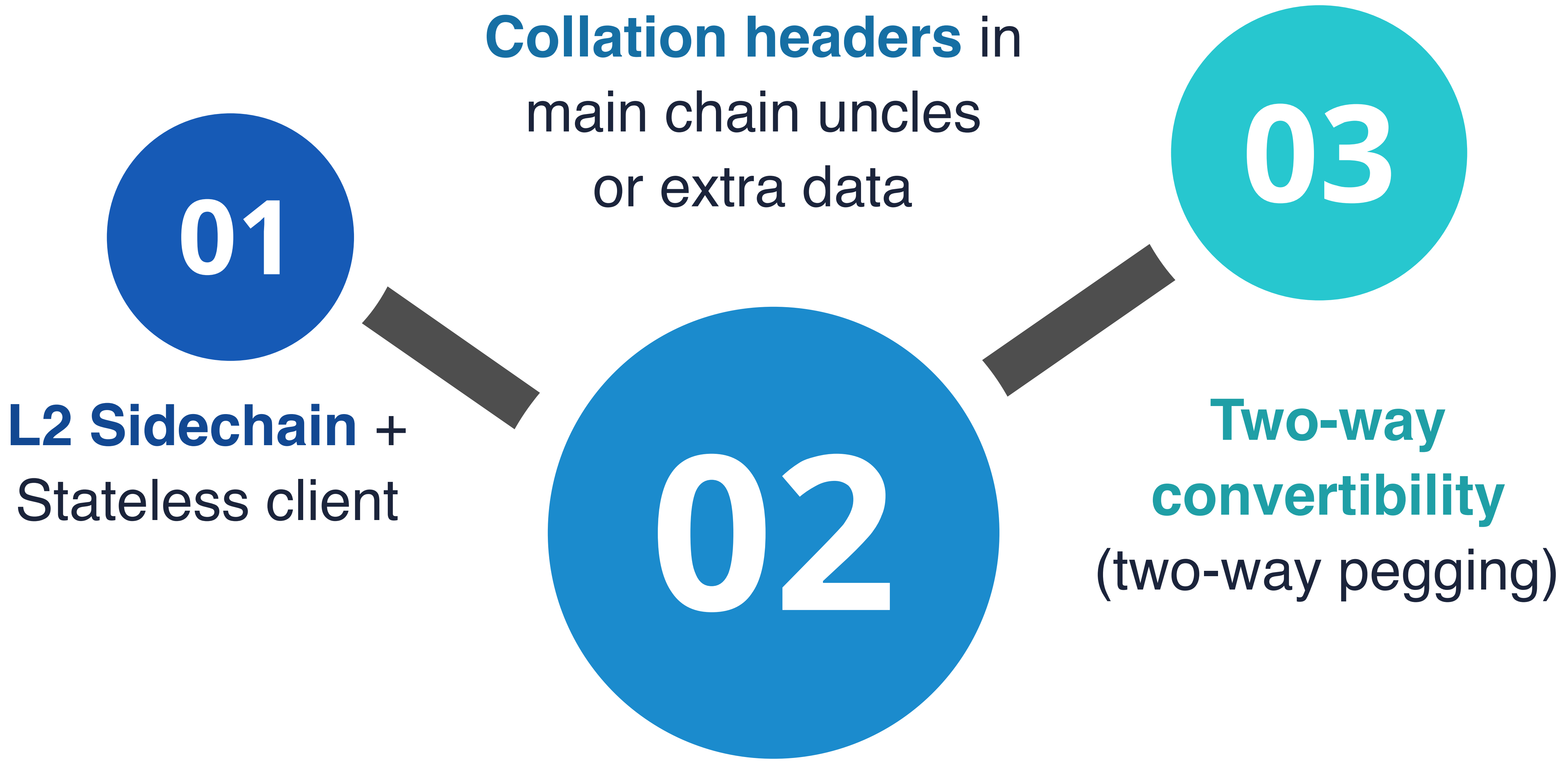
01

L2 Sidechain +
Stateless client

02

Collation headers in
main chain uncles
or extra data

Roadmap



01

L2 Sidechain + Stateless client

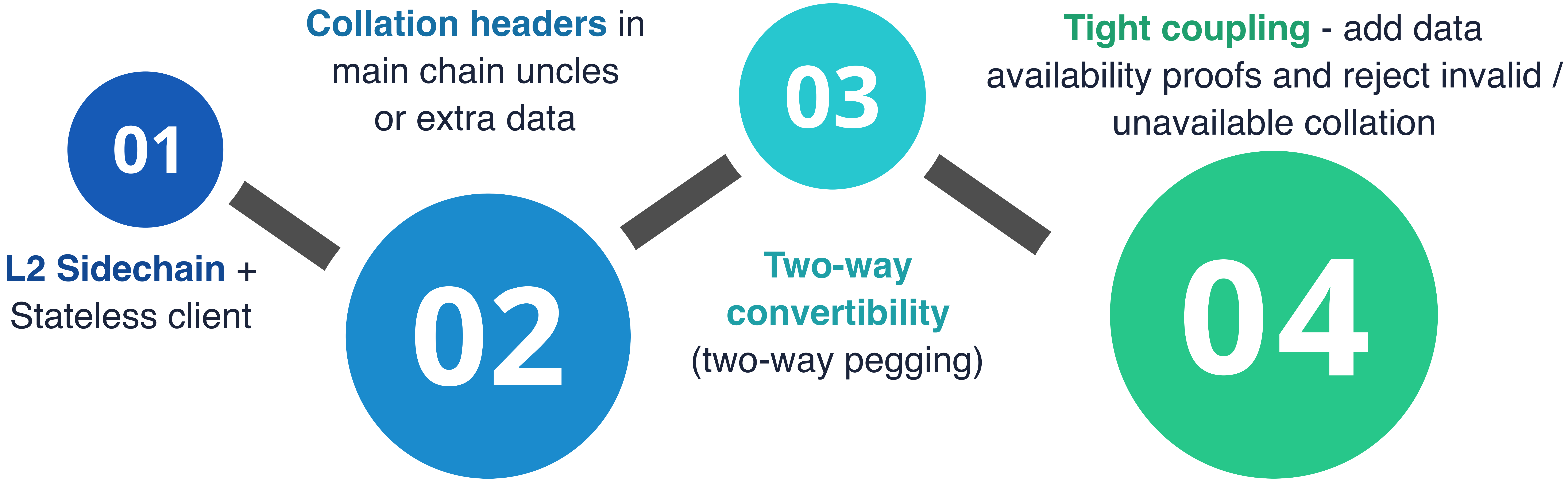
02

Collation headers in main chain uncles or extra data

03

Two-way convertibility (two-way pegging)

Roadmap



01

L2 Sidechain + Stateless client

02

Collation headers in main chain uncles or extra data

03

Two-way convertibility (two-way pegging)

04

Tight coupling - add data availability proofs and reject invalid / unavailable collation

Conclusion

结论

Conclusion

- The scalability problems will be improved with multiple phases
  以太坊可扩展性问题将由不同解决方案与多个阶段逐步改善

- In the new shards, we will have opportunities to try some revolutionary cool ideas
  在新的分片，我们有机会尝试各种大幅度的的強化

Resource and Acknowledgements

- **Sharding FAQ**
  https://github.com/ethereum/wiki/wiki/Sharding-FAQ
- **Ethereum Research**
  https://ethresear.ch/c/sharding
- **Sharding PoC**
  https://github.com/ethereum/sharding/
- **gitter ethereum/casper-scaling-and-protocol-economics channel**
  https://gitter.im/ethereum/casper-scaling-and-protocol-economics
- **Vitalik Buterin, Mai-Hsuan (Kevin) Chia, Nicholas Lin, Lane Rettig**

Thanks!

You can find me on gitter: @hwwhww