이더리움 확장성 이슈 및 해결 방안 (샤딩, 플라즈마)

한겨레 kr8534@gmail.com

이종복 acejongbok@gmail.com

Agenda

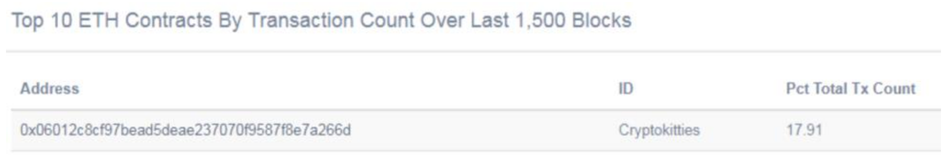
- Ethereum Scaling Challenges
- Ethereum Sharding
    - Block Proposer
    - Cross-shard Communication
    - (Stateless Client)
    - (Fork Choice Rule of Shard Chain)
- Etheruem Plasma

자기소개 슬라이드

Ethereum Scaling Challenges
Issues on Ethereum

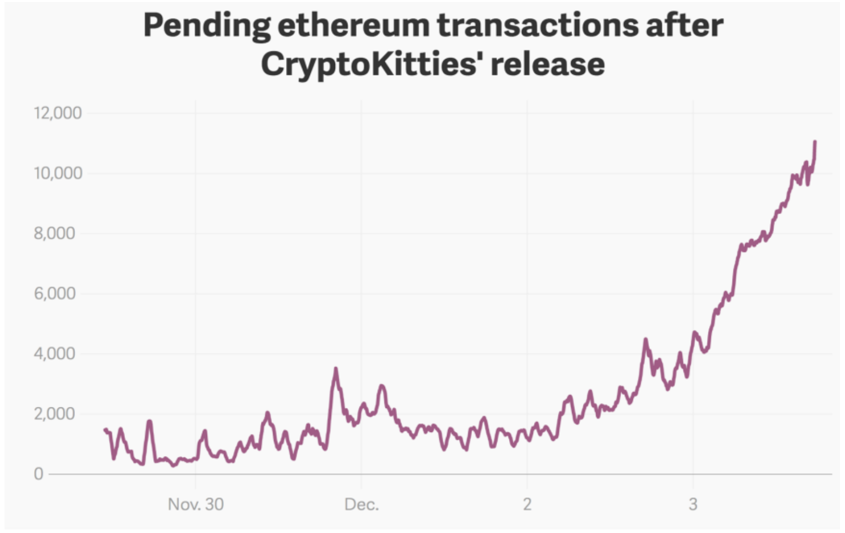
- 15~20 transactions per second
- Pending TX jumps on ICO

dApp | ICO

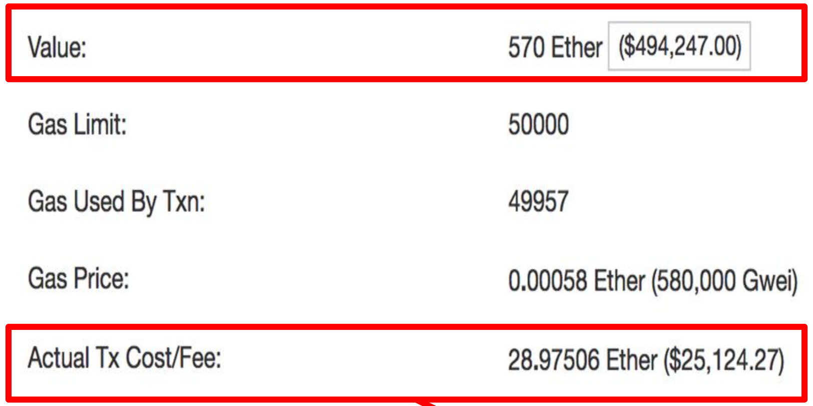


Top 10 ETH Contracts By Transaction Count Over Last 1,500 Blocks

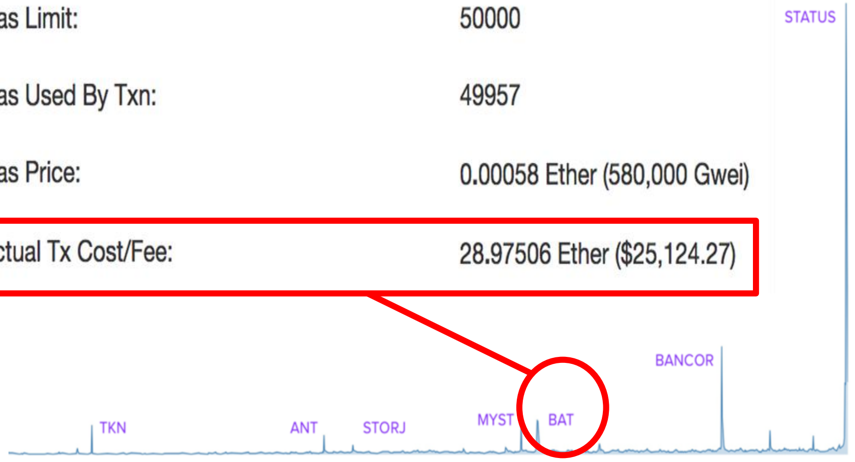
Address	ID	Pct Total Tx Count
0x06012c8cf97bead5deae237070f9587f8e7a266d	Cryptokitties	17.91



BAT ICO 당시
25000$ (총 value 의 약 5%)을 TX fee로 지불한 사례



Value:	570 Ether ($494,247.00)
Gas Limit:	50000
Gas Used By Txn:	49957
Gas Price:	0.00058 Ether (580,000 Gwei)
Actual Tx Cost/Fee:	28.97506 Ether ($25,124.27)



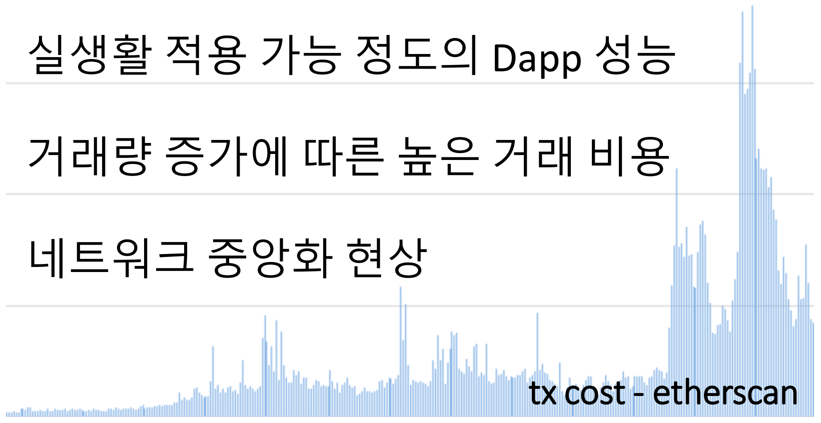
Ethereum Scaling Challenges

Importance of Scalability

실생활 적용 가능 정도의 Dapp 성능

거래량 증가에 따른 높은 거래 비용

네트워크 중앙화 현상


tx cost - etherscan

Computing

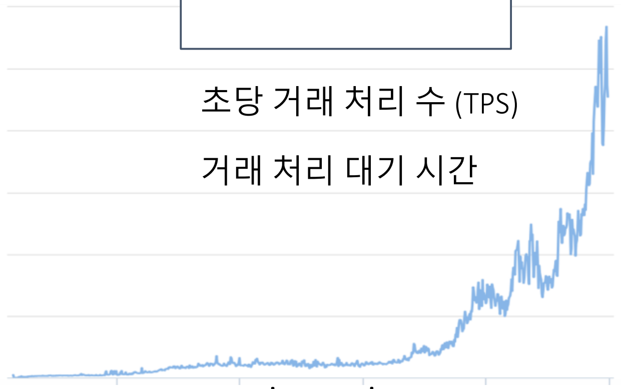
각 노드의 블록 검증 소요 시간

데이터 베이스 read/write 시간

Network

초당 거래 처리 수 (TPS)

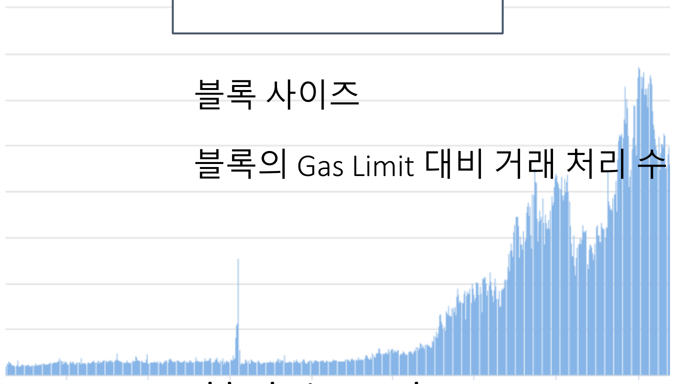
거래 처리 대기 시간


tx per day - etherscan

Storage

블록 사이즈

블록의 Gas Limit 대비 거래 처리 수

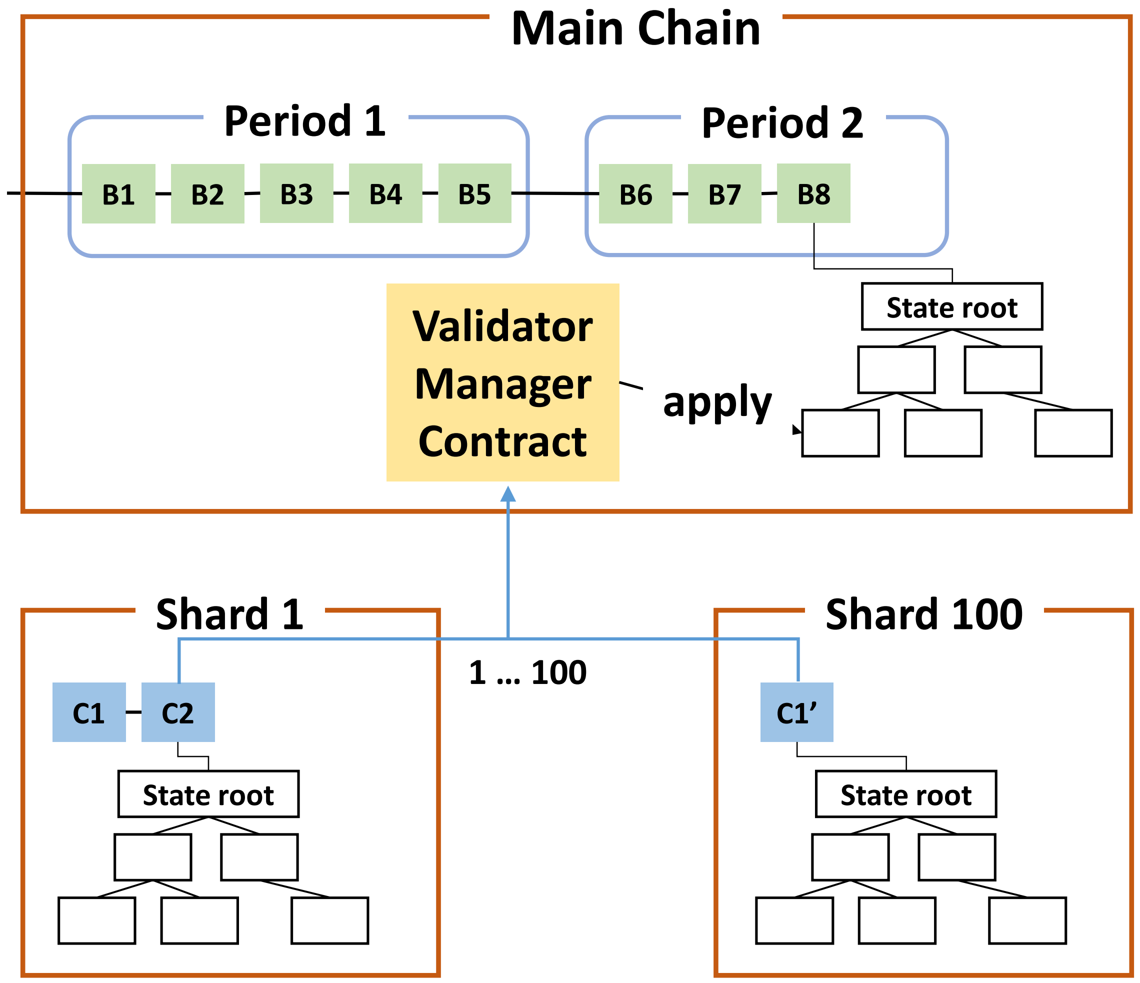

avg block size - etherscan

Ethereum Scaling Challenges

Current Attepts to Resolve

Name	Method	Description
Casper	**On**-chain	작업증명 방식에서 지분증명 방식으로 전환
Raiden Network	**Off**-chain, **Channel**-based	사용자간 처음과 마지막 거래만 기록하되, 중간 거래는 **Off**-chain의 채널에서 sign된 거래 교환으로 진행
Sharding	**On**-chain, **Chain**-based	검증자는 블록체인 state의 일부만 검증
Plasma	**Off**-chain, **Chain**-based	트리 구조의 다중 블록체인을 구축

Ethereum Sharding
Overview



- 포크가 필요없음

- **Validator Manager Contract** 을 통해 mainnet에 바로 적용 가능

- Phase 1 will provide
  - A set of shard validators
  - 100 Ethereum shards
  - Each shard will have "stateless clients", "account abstraction"

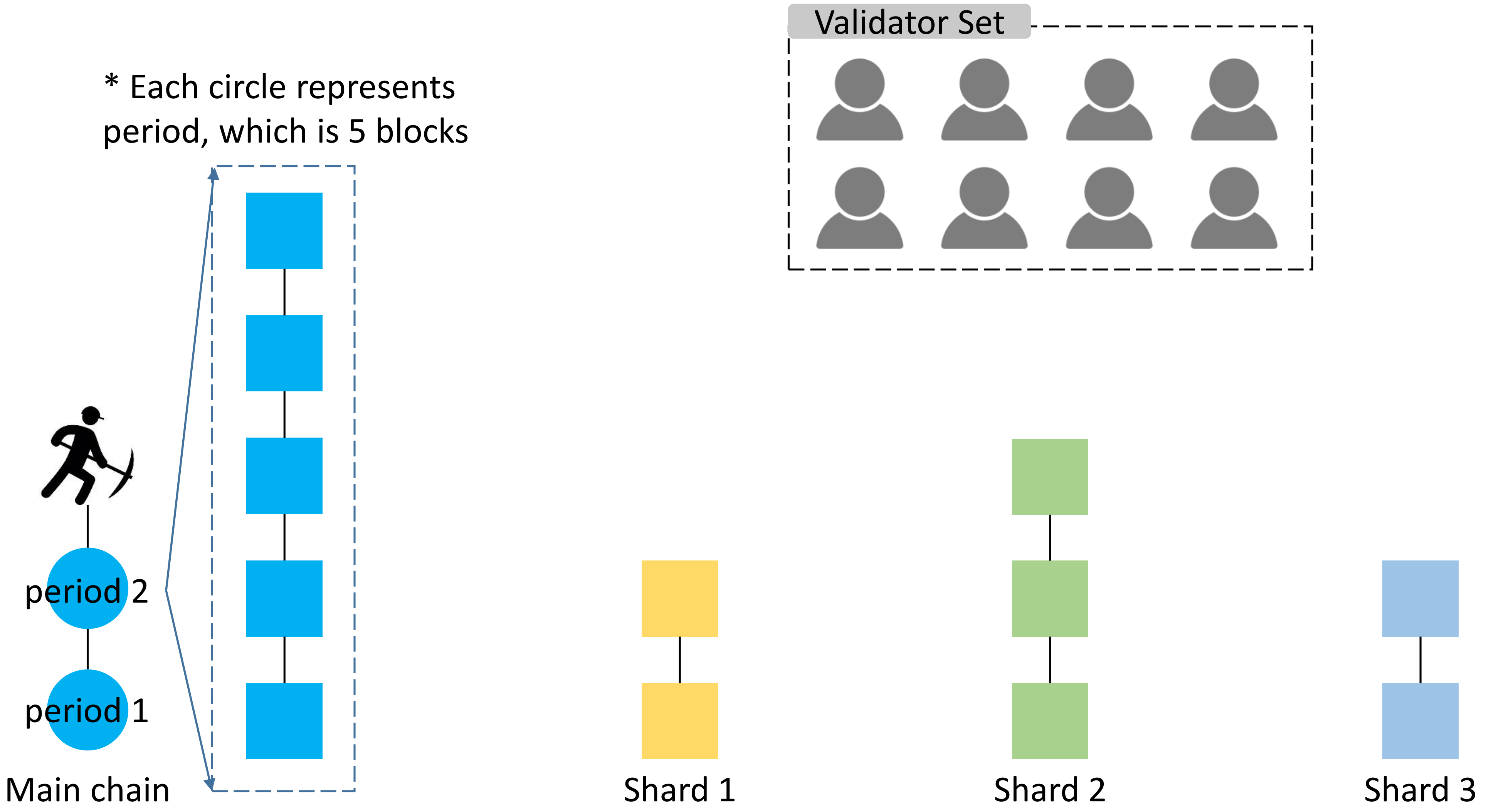
Ethereum Sharding
Block Proposal

- Validators are randomly sampled to propose new collation headers
- Collation headers are like block headers but for a particular shard

- 5개 블록마다, 새로운 **period** 시작
- One collation per shard, per period
- Validator는 LOOKAHEAD period 전에 random 하게 선택됨
  - Deposit을 건 validator set에서 매번 선정
- Validators verify block validity as far back as desired

Ethereum Sharding

Block Proposal



Ethereum Sharding
Block Proposal

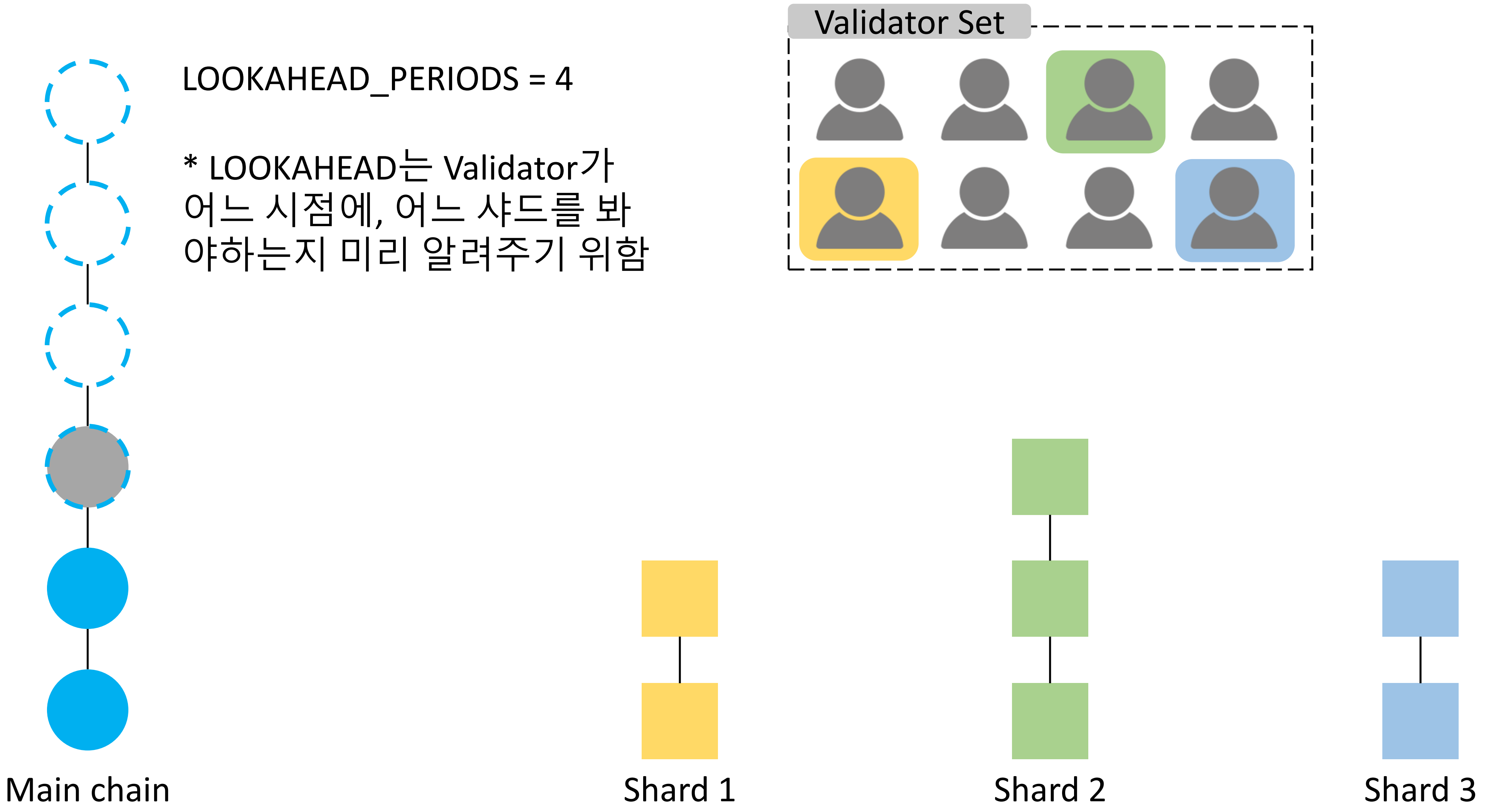
1. **Validators use LOOKAHEAD to check which shards they will be validating**
2. Validators download latest shard state
3. Validators verify blocks until some depth and pick head to build on
4. Client submits transaction with access list and witness
5. Validator pull relevant transactions from the mempool
6. Validators submit collation header to the root chain

Ethereum Sharding

Block Proposal

LOOKAHEAD_PERIODS = 4

* LOOKAHEAD는 Validator가
어느 시점에, 어느 샤드를 봐
야하는지 미리 알려주기 위함

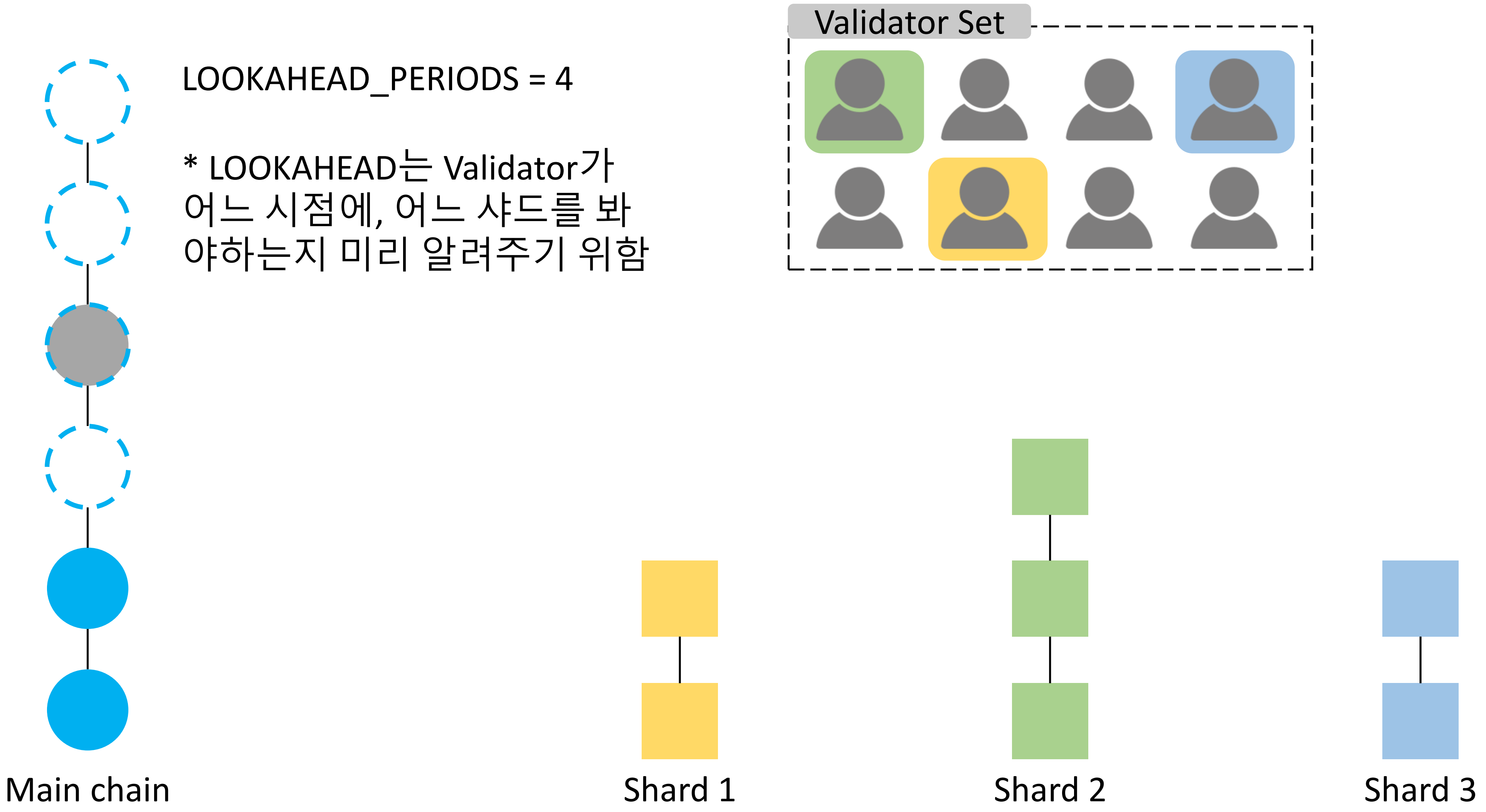


Ethereum Sharding

Block Proposal

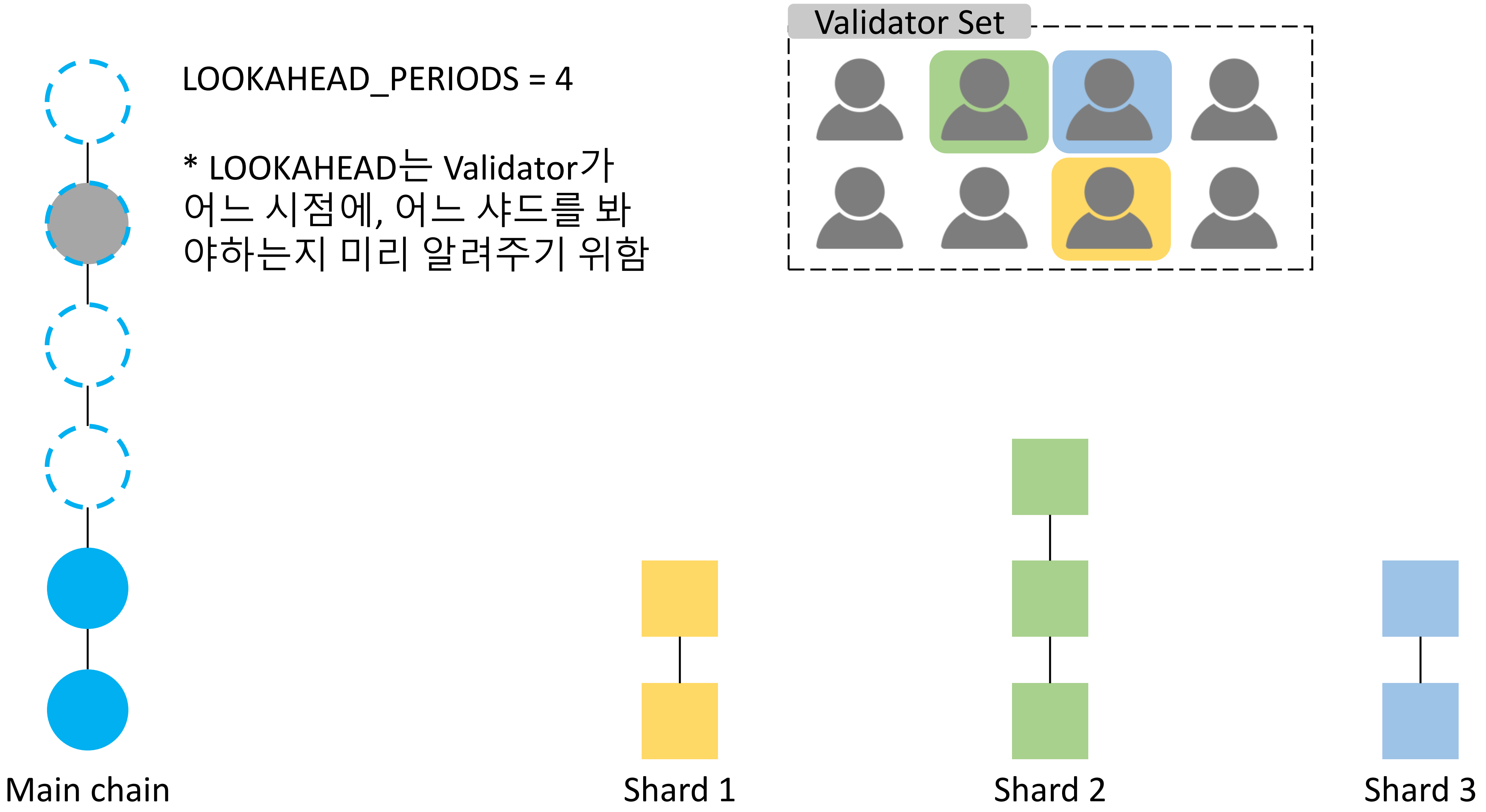
LOOKAHEAD_PERIODS = 4

* LOOKAHEAD는 Validator가 어느 시점에, 어느 샤드를 봐야하는지 미리 알려주기 위함



Ethereum Sharding

Block Proposal



LOOKAHEAD_PERIODS = 4

* LOOKAHEAD는 Validator가
어느 시점에, 어느 샤드를 봐
야하는지 미리 알려주기 위함

Ethereum Sharding

Block Proposal

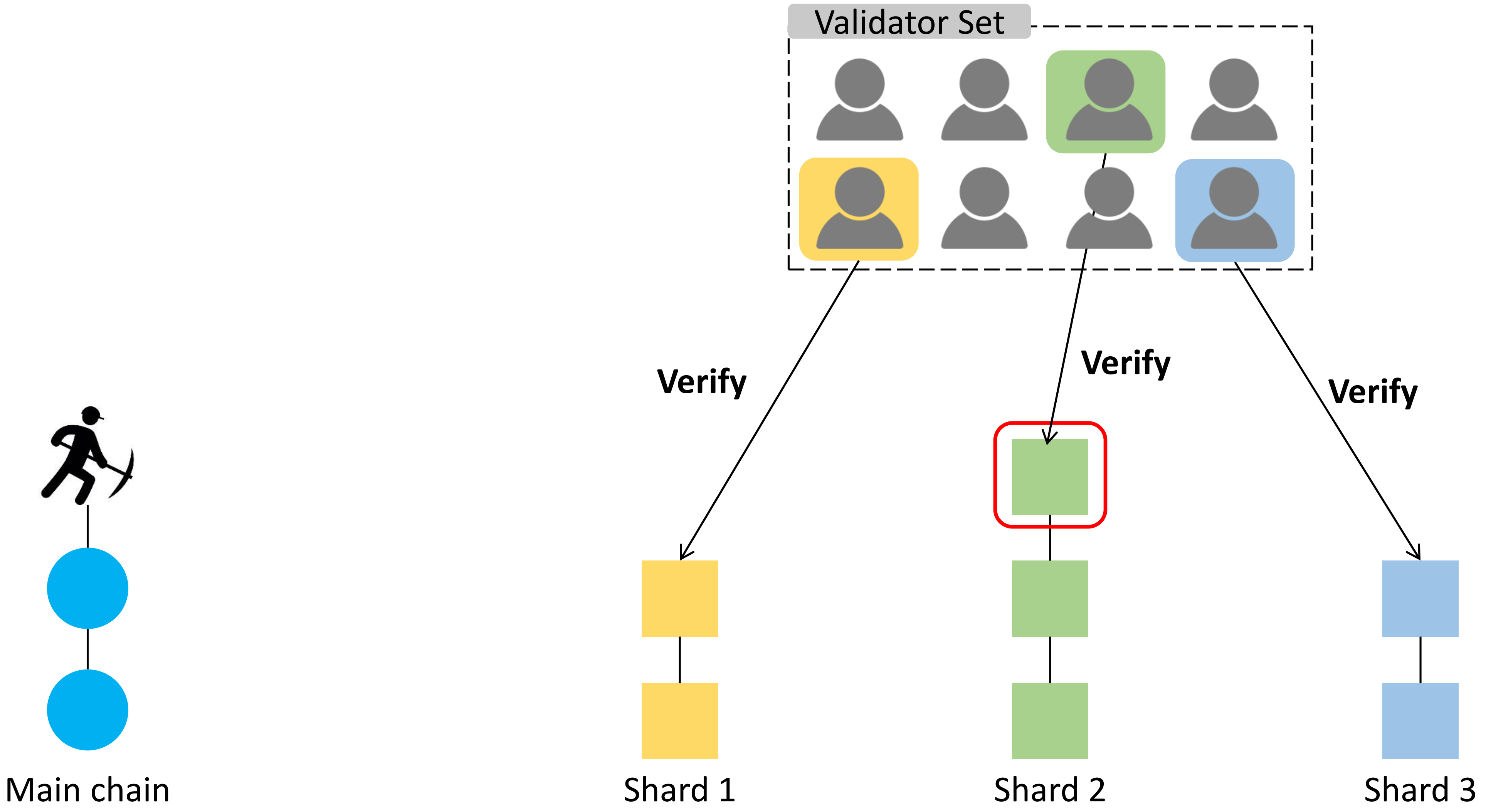
1. Validators use LOOKAHEAD to check which shards they will be validating
2. **Validators download latest shard state**
3. Validators verify blocks until some depth and pick head to build on
4. Client submits transaction with access list and witness
5. Validator pull relevant transactions from the mempool
6. Validators submit collation header to the root chain

Ethereum Sharding
Block Proposal

1. Validators use LOOKAHEAD to check which shards they will be validating
2. Validators download latest shard state
3. **Validators verify blocks until some depth and pick head to build on**
4. Client submits transaction with access list and witness
5. Validator pull relevant transactions from the mempool
6. Validators submit collation header to the root chain

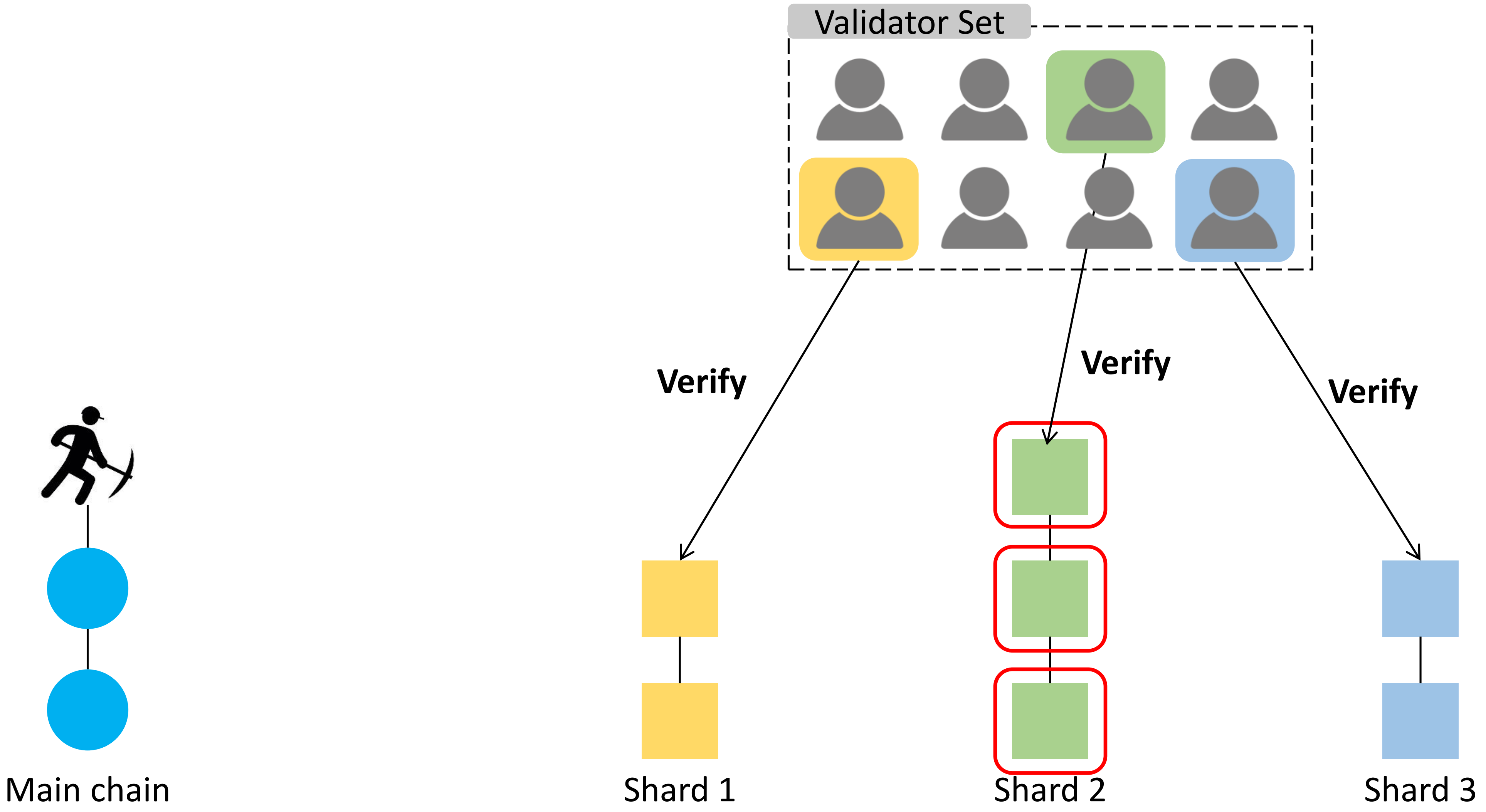
Ethereum Sharding

Block Proposal



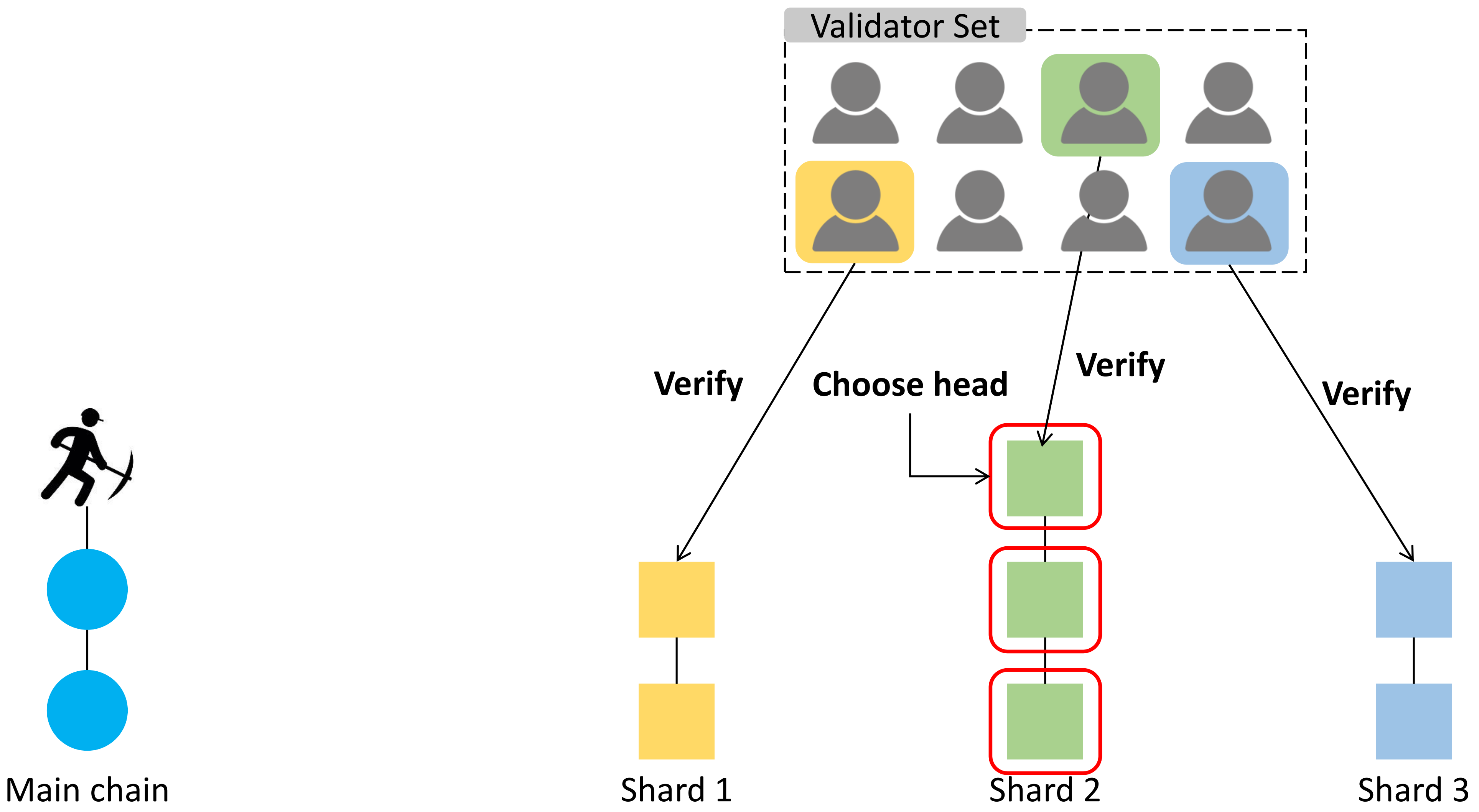
Ethereum Sharding

Block Proposal



Ethereum Sharding

Block Proposal



Ethereum Sharding
Block Proposal

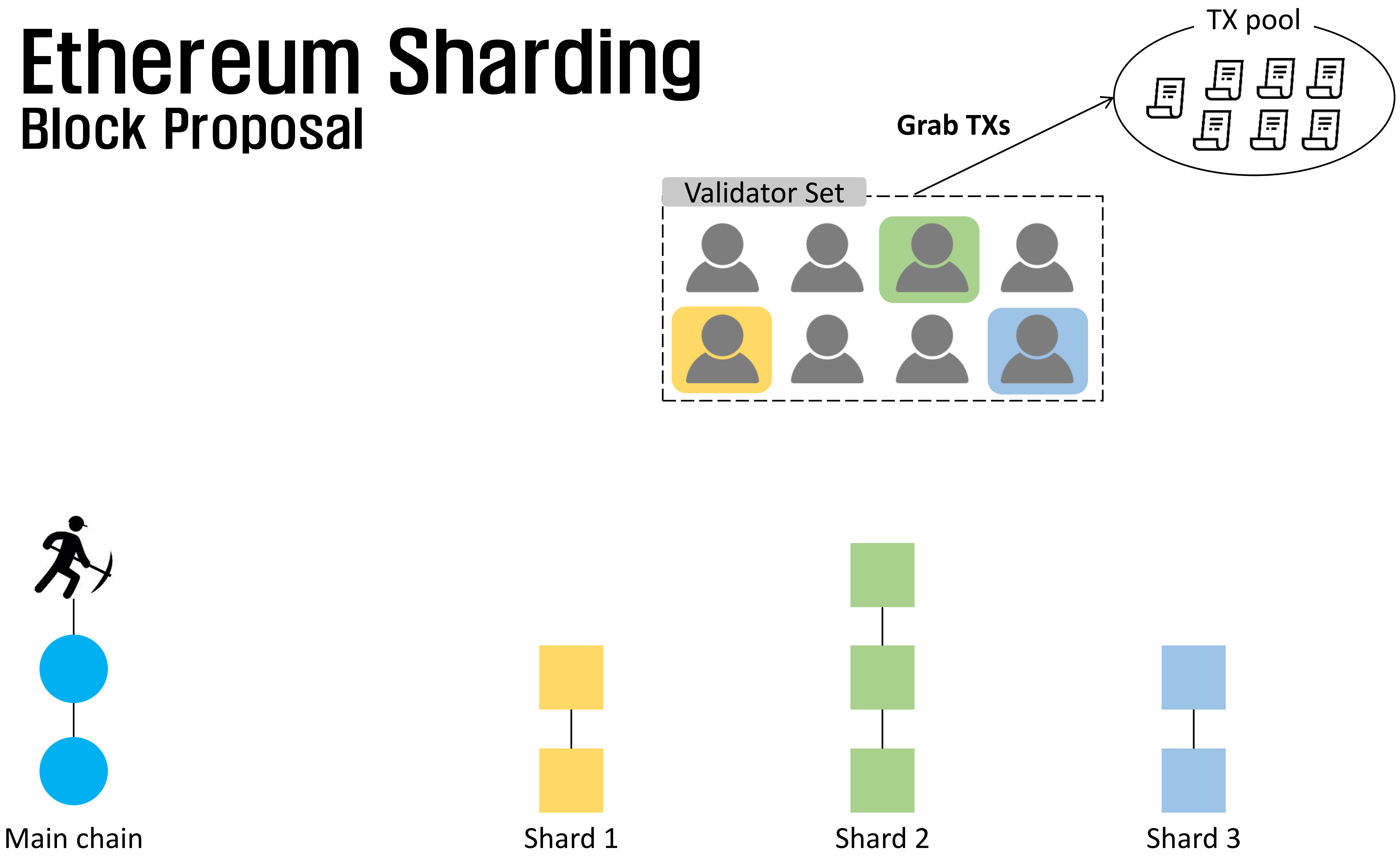
1. Validators use LOOKAHEAD to check which shards they will be validating
2. Validators download latest shard state
3. Validators verify blocks until some depth and pick head to build on
4. **Client submits transaction with access list and witness => covered later**
5. Validator pull relevant transactions from the mempool
6. Validators submit collation header to the root chain

Ethereum Sharding
Block Proposal

1. Validators use LOOKAHEAD to check which shards they will be validating
2. Validators download latest shard state
3. Validators verify blocks until some depth and pick head to build on
4. Client submits transaction with access list and witness => covered later
5. **Validator pull relevant transactions from the mempool**
6. Validators submit collation header to the root chain

Ethereum Sharding

Block Proposal

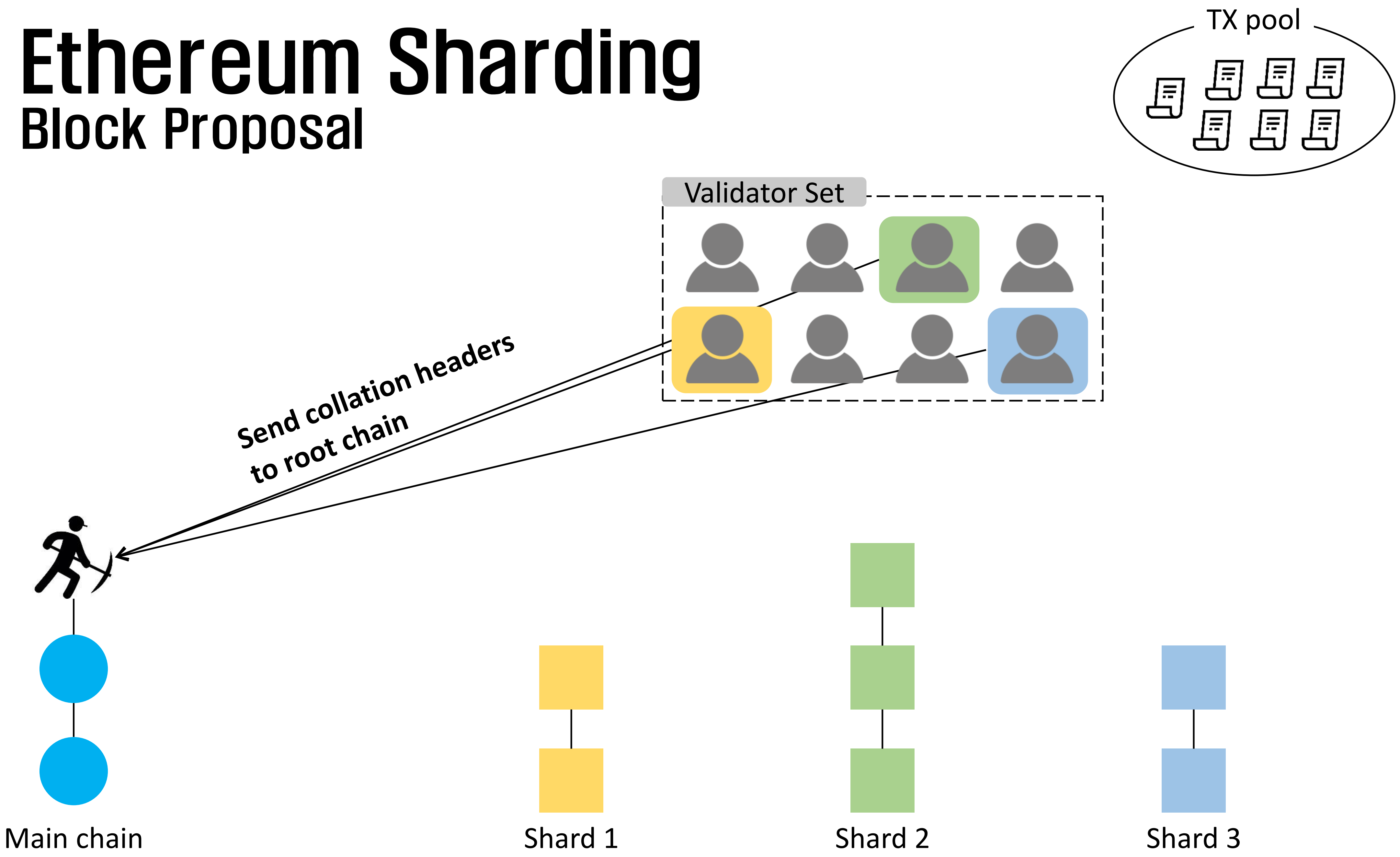


Ethereum Sharding

Block Proposal

1. Validators use LOOKAHEAD to check which shards they will be validating
2. Validators download latest shard state
3. Validators verify blocks until some depth and pick head to build on
4. Client submits transaction with access list and witness => covered later
5. Validator pull relevant transactions from the mempool
6. **Validators submit collation header to the root chain**

Ethereum Sharding
Block Proposal



Ethereum Sharding

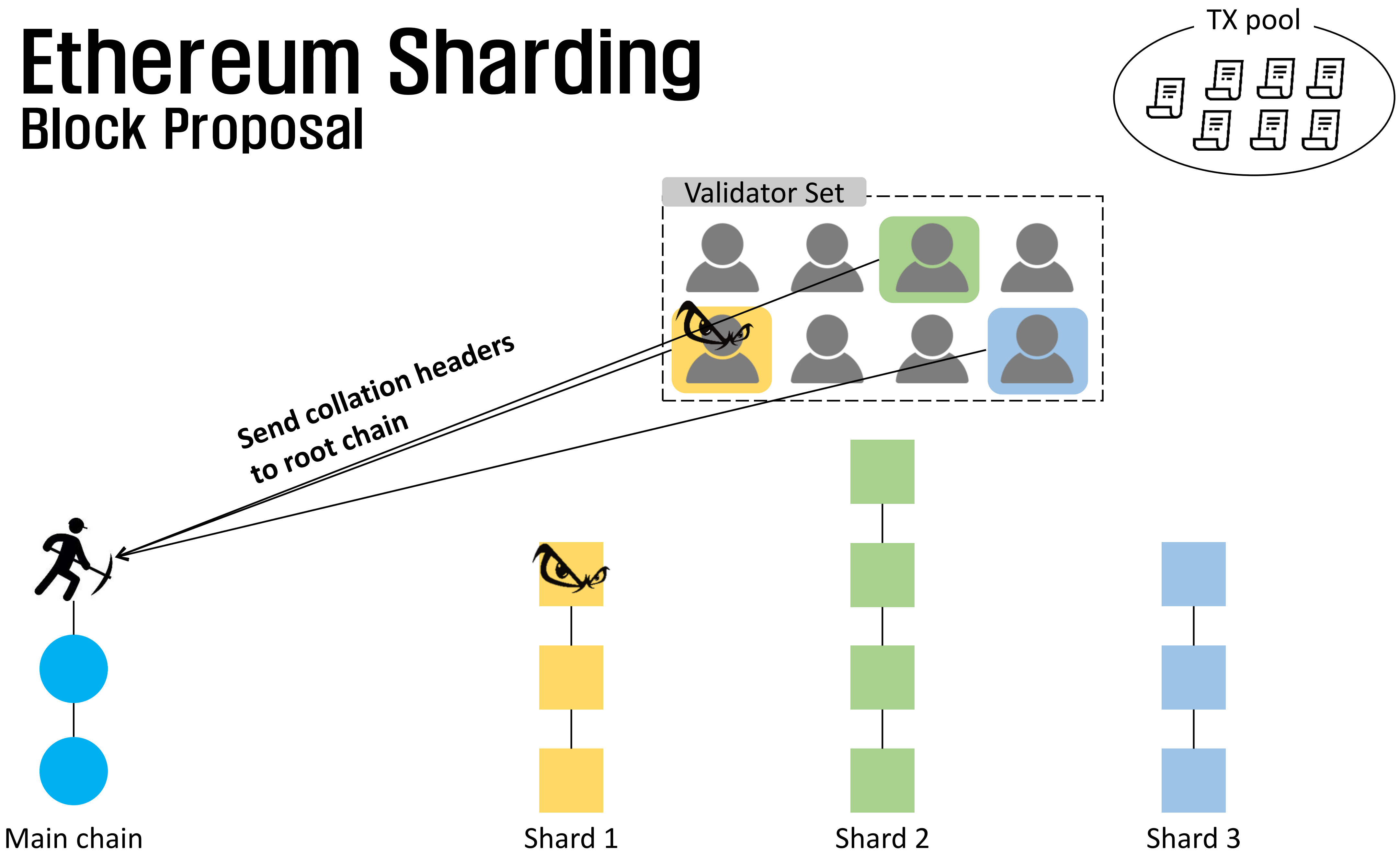
Block Proposal

1. Validators use LOOKAHEAD to check which shards they will be validating
2. Validators download latest shard state
3. Validators verify blocks until some depth and pick head to build on
4. Client submits transaction with access list and witness => covered later
5. Validator pull relevant transactions from the mempool
6. Validators submit collation header to the root chain

How to handle the case that evil validator submits invalid block?

Ethereum Sharding

Block Proposal

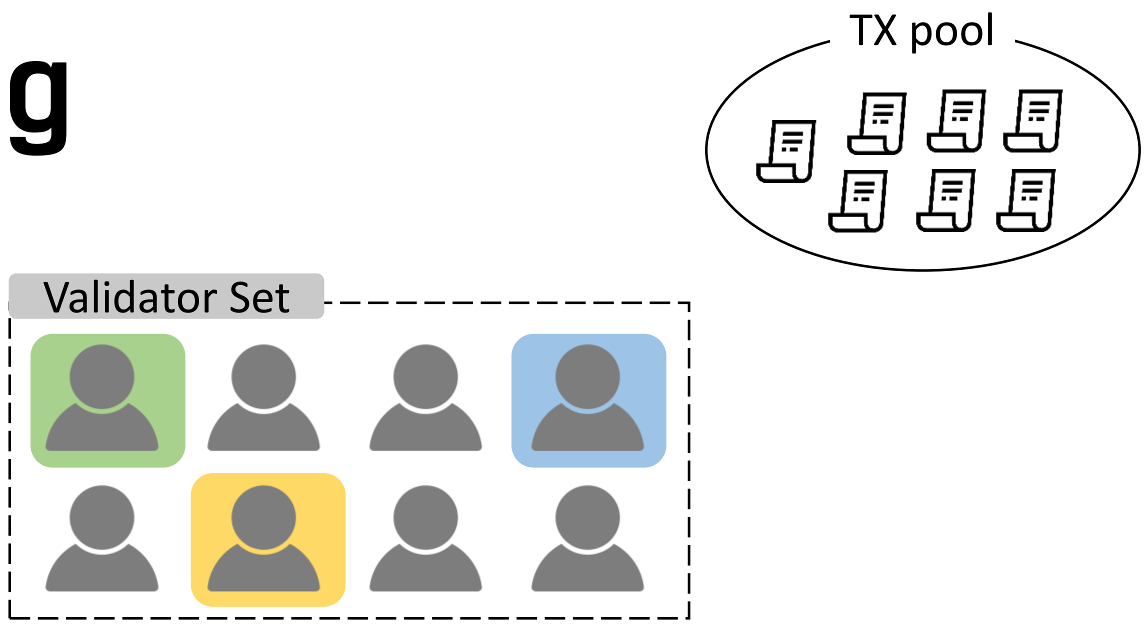


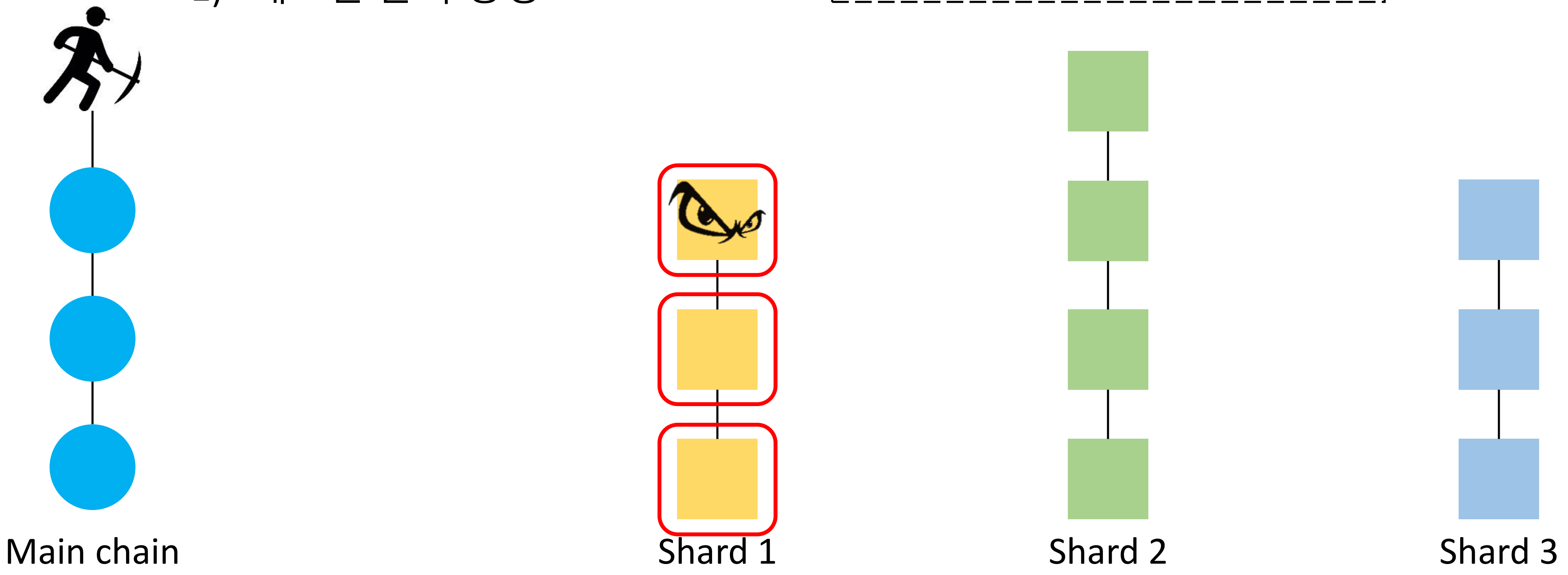
Ethereum Sharding

Block Proposal

1) Validation 단계에서, shard 1의 validator가 invalid collation을 notify

2) 새로운 분기 생성



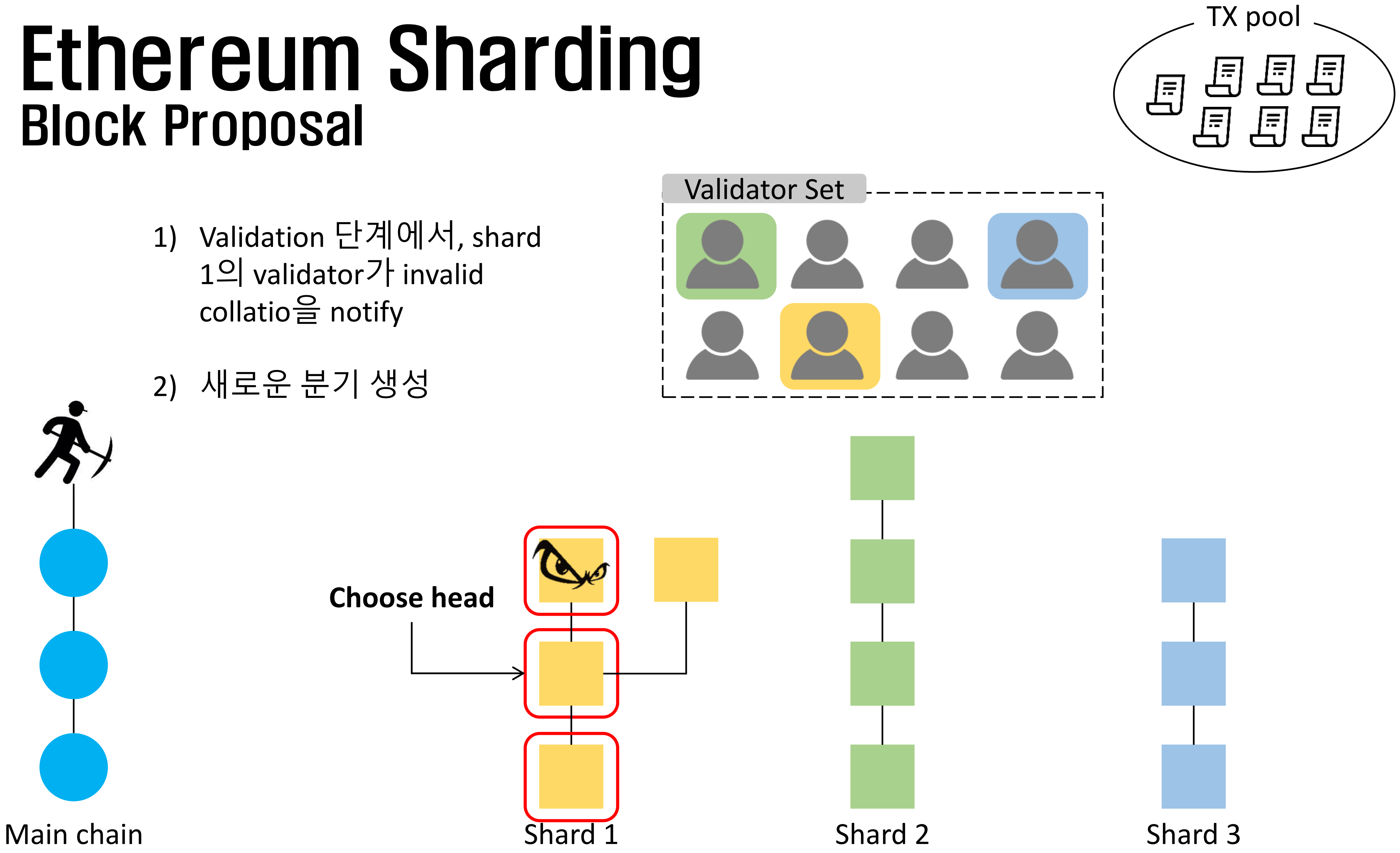


Ethereum Sharding

Block Proposal

1) Validation 단계에서, shard 1의 validator가 invalid collatio을 notify

2) 새로운 분기 생성



Ethereum Sharding

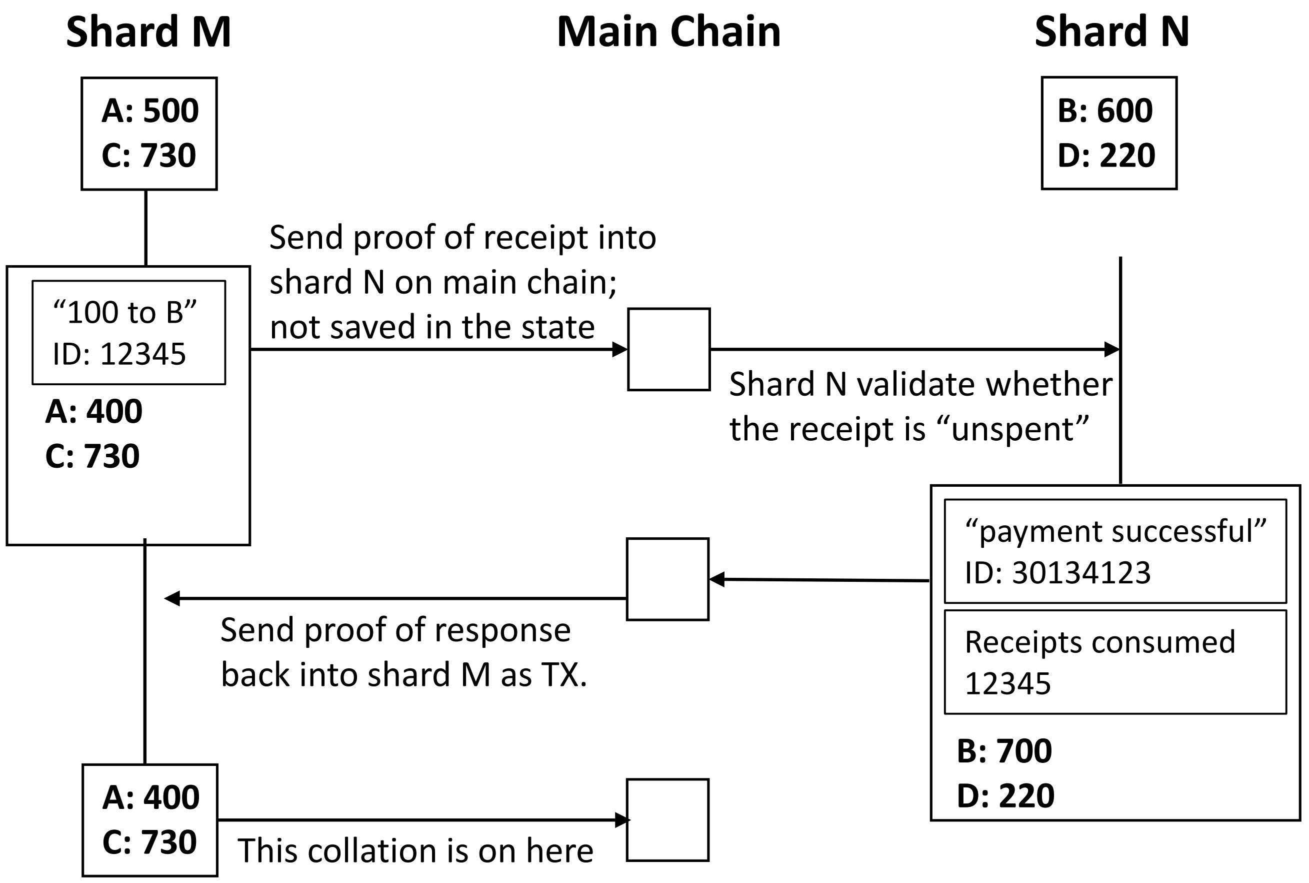
Block Proposal

1. Validators use LOOKAHEAD to check which shards they will be validating
2. Validators download latest shard state
3. Validators verify blocks until some depth and pick head to build on
4. Client submits transaction with access list and witness => covered later
5. Validator pull relevant transactions from the mempool
6. Validators submit collation header to the root chain

How to handle the case that evil validator submits invalid block? ➡ How Casper works on this?

Ethereum Sharding

Cross-shard Communication



Shard M
Main Chain
Shard N
A: 500
C: 730
B: 600
D: 220
"100 to B"
ID: 12345
A: 400
C: 730
Send proof of receipt into shard N on main chain; not saved in the state
Shard N validate whether the receipt is "unspent"
"payment successful"
ID: 30134123
Receipts consumed
12345
B: 700
D: 220
Send proof of response back into shard M as TX.
A: 400
C: 730
This collation is on here

Receipt-based communication
a.k.a "debit" or "credit" base.

1. Send a TX on Shard M and create a receipt
2. Wait for the first TX to be included
3. Send a TX on shard N which includes the proof of the receipt. (validate the receipt & increase appro-priate balance)

마무리지으며

- 샤딩은 현재도 활발히 논의되고 있고 결정된 spec이 없음
  (4단계의 로드맵이 있지만 phase 1을 제외하고는 정보가 거의 없음)
- Whitepaper, yellowpaper 와 같이 명시적으로 설명하고 있는 document가 매우 부족
  (현재는 github sharding document, medium post, Youtube clip)
- 아직까지는 Python clien에서만 test되고 있는 단계
- 추가필요

References

- https://medium.com/@icebearhww/ethereum-sharding-and-finality-65248951f649
- https://github.com/ethereum/sharding/blob/develop/docs/doc.md
- https://github.com/ethereum/wiki/wiki/Sharding-FAQ
- https://github.com/ethereum/wiki/wiki/chain-fibers-redux
- https://github.com/ethereum/sharding/tree/develop/sharding
- https://medium.com/l4-media/making-sense-of-ethereums-layer-2-scaling-solutions-state-channels-plasma-and-truebit-22cb40dcc2f4
- https://www.youtube.com/channel/UC7tELjcjz84KlbQJf0t-euQ
- Ethresear.ch

감사합니다

발표를 도와주신 이더리움 연구회 2기, Alex Daniel 찬혁 윤님
Shout out to Vitalik Buterin, Hsiao-Wei Wang, Karl Floersch

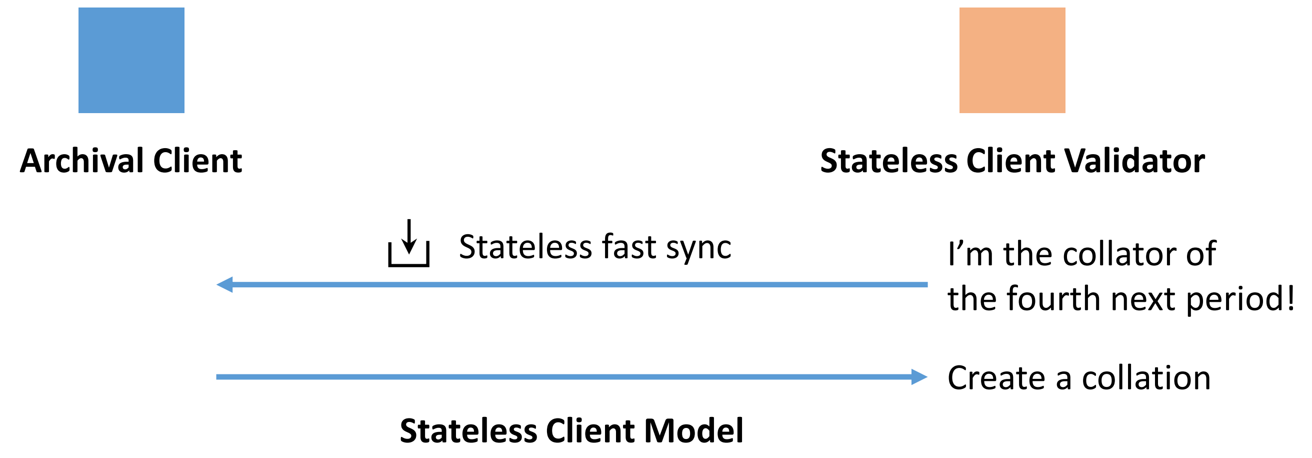
Backup

Stateless Client

Fork-Choice Rule of Shard Chain

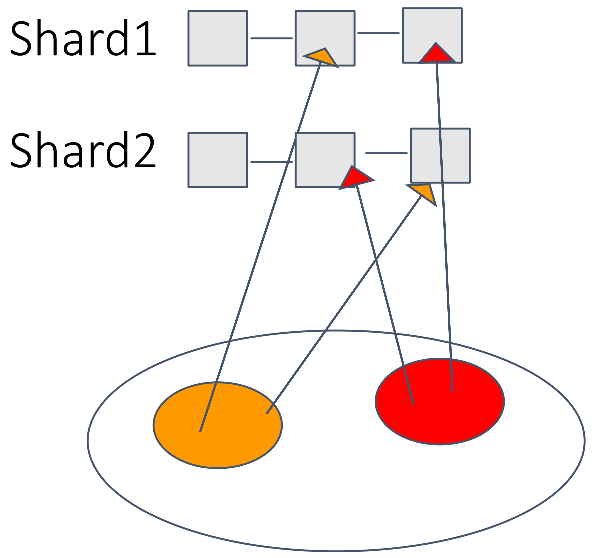
Ethereum Sharding
Stateless Client

- Basic Concept
  - Stateless clients only store the state trie root
  - The archival clients store the full state trie and provide the Merkle branches that the given collation needs
  - With these branches, the stateless clients are able to build partial state trie and verify
  - Validator only have to validate the recent collations to sync with the shard



Ethereum Sharding
Stateless Client

Synchronizing state incurs overhead



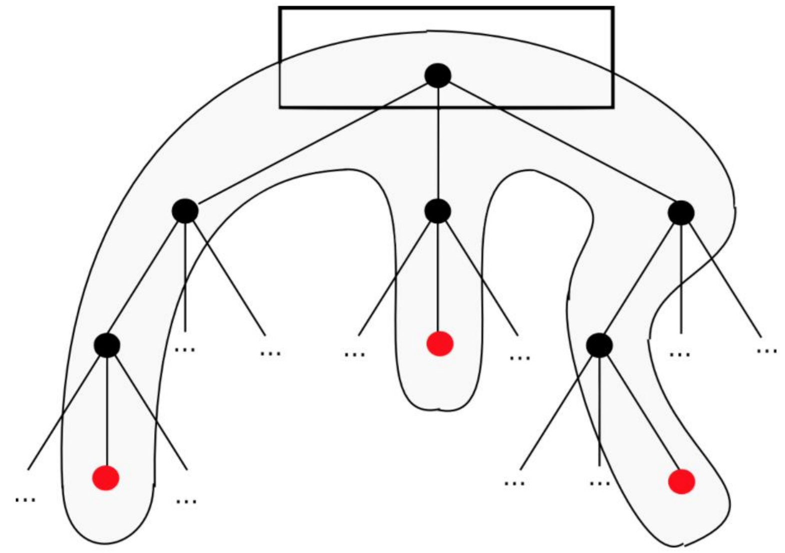
```
NewState = StateFunction(State, Block)

                    ⇩

NewStateObj = ApplyStateFunction(
            StateObj,
            Witness,
            Block
)
```

Ethereum Sharding
Stateless Client

```
NewStateObj = ASF(
            StateObj,
        Witness,
        Block
)
```

tuple containing the **state root** and ot
her state data (gas used, receipts, blo
om filter, etc)

RLP-encoded list of Merkle tree nodes t
hat provides the **portions of the state** pr
ovided by **TX Sender**



Ethereum Sharding

Transaction / Collation Header Format

```
TX :

  [
      [nonce, acct, data....],    # transaction body
      [node1, node2, node3....]   # witness
  ]

Collation Header:

  [
      [shard_id, ... , sig],   # header
      [tx1, tx2 ...],          # transaction list
      [node1, node2, node3...] # witness
  ]
```

Ethereum Sharding

AddHeader

previous state root: 0x12bc57,

Merkle root of the transactions: 0x3f98ea,

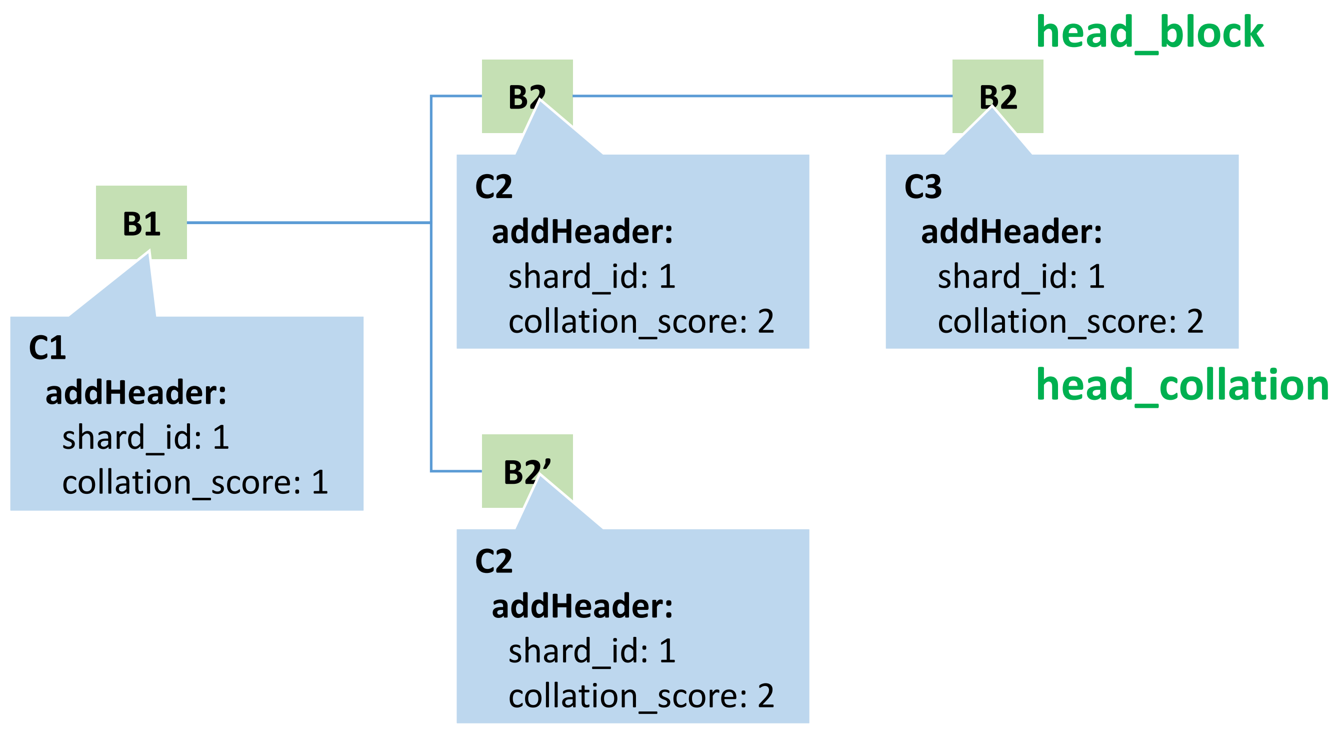
state root after processing transactions: 0x5d0cc1.

Signed, collators #1, 2, 4, 5, 6, 8, 11, 13 ... 98, 99 (⅔)

$$\begin{tikzcd}
\text{rlp(header)} \arrow[d] \\
\text{AddHeader(shard\_id, collation\_header)} \arrow[d] \\
\text{VMC}
\end{tikzcd}$$

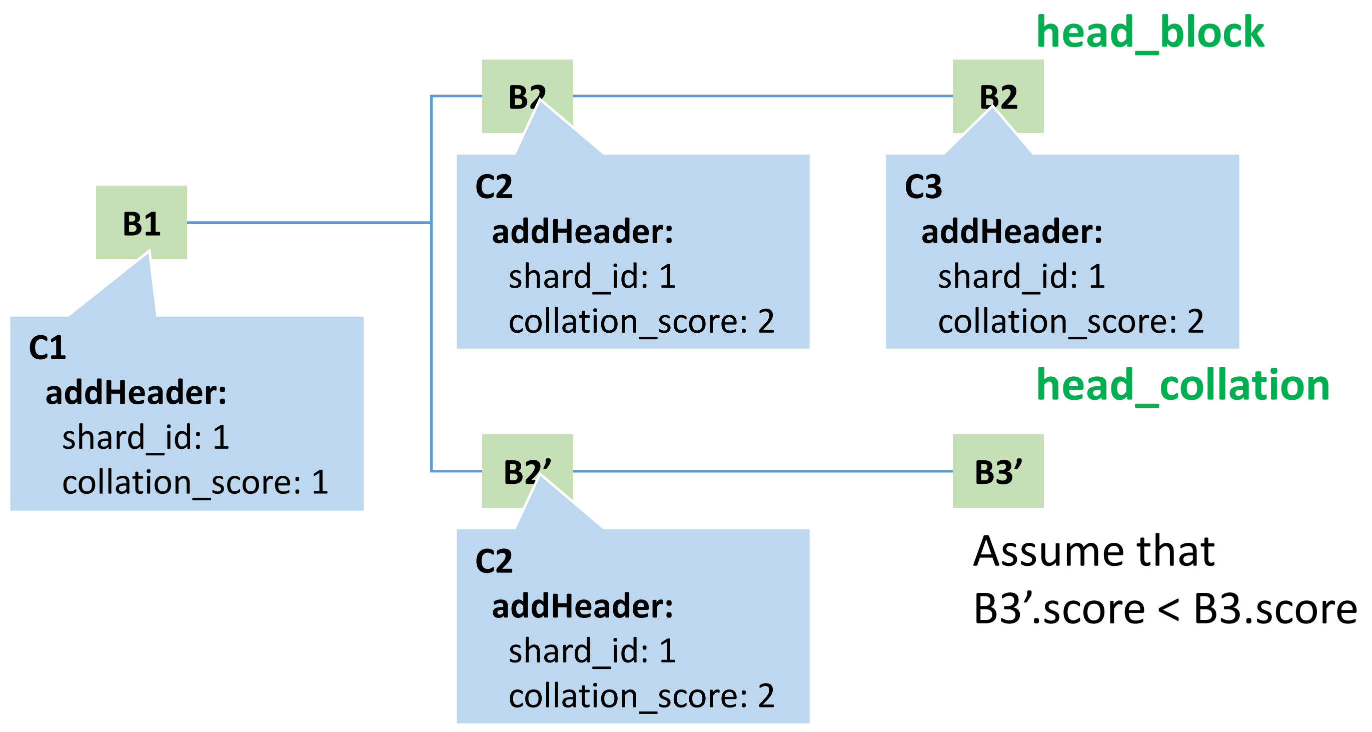
Ethereum Sharding
Fork Choice Rule of Shard Chain

- The fork choice rule depends on the longest main chain.
- Not simply the head collation of “longest valid shard chain”, but
  “The longest valid shard chain within the longest valid main chain”


head_block
B2
B2
B1
C2 addHeader: shard_id: 1 collation_score: 2
C3 addHeader: shard_id: 1 collation_score: 2
C1 addHeader: shard_id: 1 collation_score: 1
head_collation
B2’
C2 addHeader: shard_id: 1 collation_score: 2

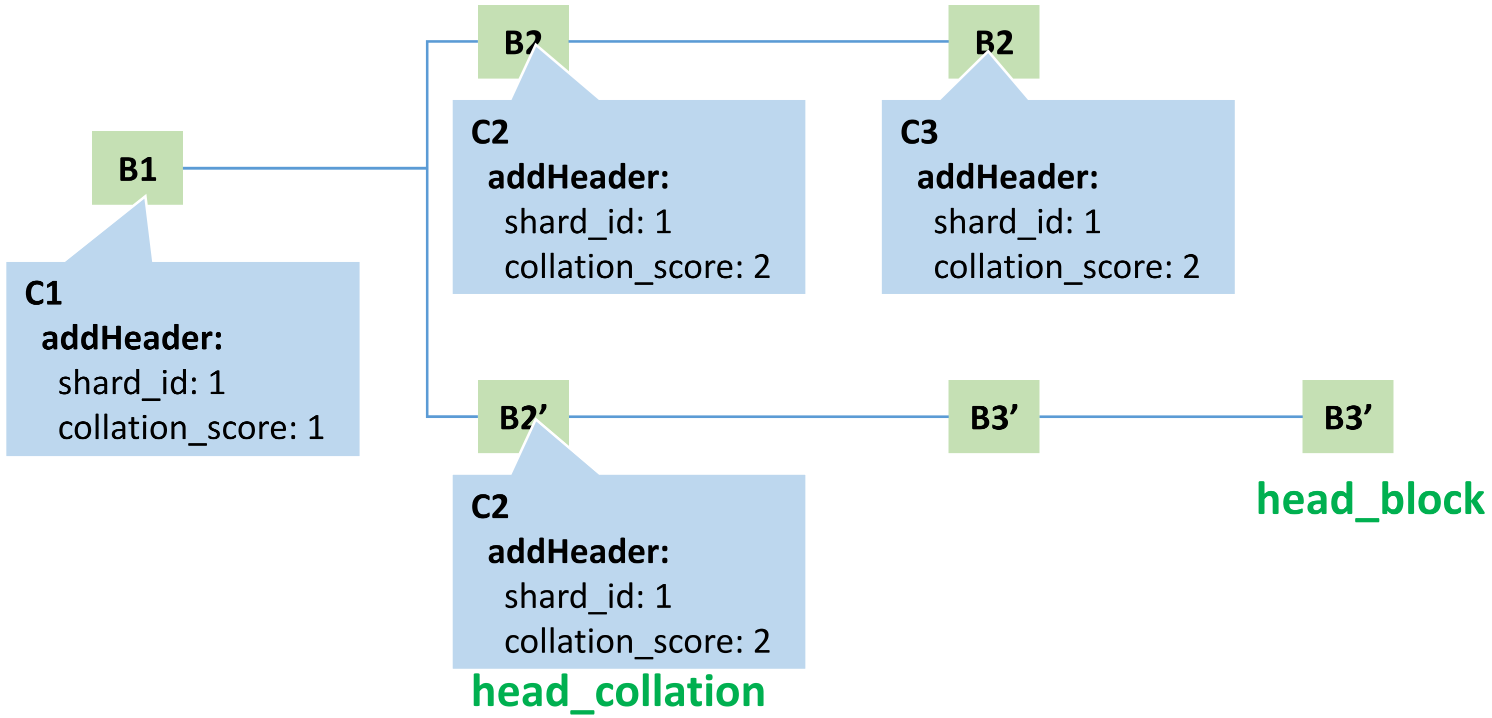
Ethereum Sharding
Fork Choice Rule of Shard Chain

- The fork choice rule depends on the longest main chain.
- Not simply the head collation of "longest valid shard chain", but **"The longest valid shard chain within the longest valid main chain"**



Ethereum Sharding
Fork Choice Rule of Shard Chain

- The fork choice rule depends on the longest main chain.
- Not simply the head collation of "longest valid shard chain", but **"The longest valid shard chain within the longest valid main chain"**


B1
C1
addHeader:
shard_id: 1
collation_score: 1
B2
C2
addHeader:
shard_id: 1
collation_score: 2
B2
C3
addHeader:
shard_id: 1
collation_score: 2
B2'
C2
addHeader:
shard_id: 1
collation_score: 2
head_collation
B3'
B3'
head_block