Front-running Attack in Sharded Blockchains and Fair Cross-shard Consensus

Jianting Zhang[1], Wuhui Chen[2], Sifu Luo[2], Tiantian Gong[1], Zicong Hong[3], Aniket Kate[1,4]

[1]Purdue University, [2]Sun Yat-sen University
[3]The Hong Kong Polytechnic University, [4]Supra Research

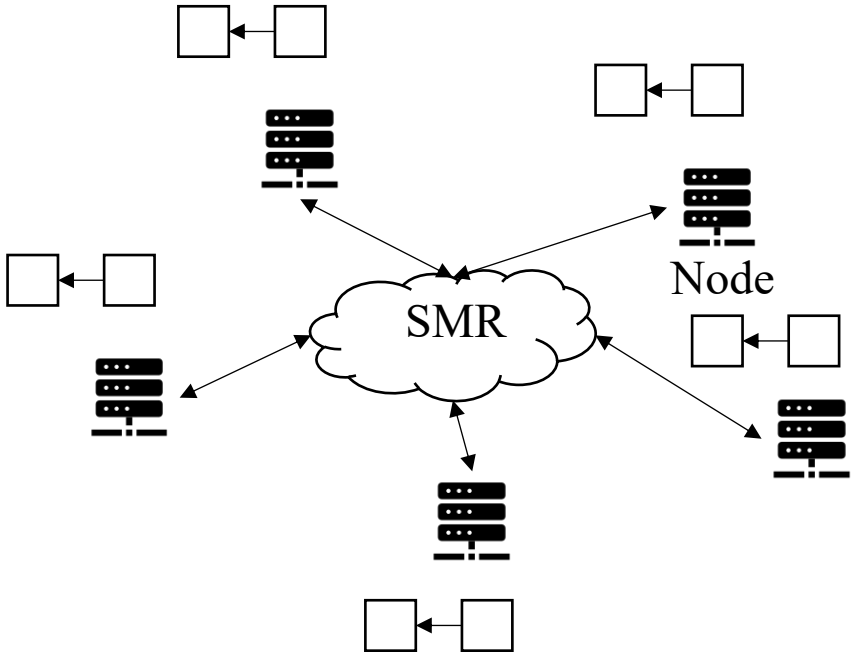
NDSS SYMPOSIUM/2024

Presented by Internet Society

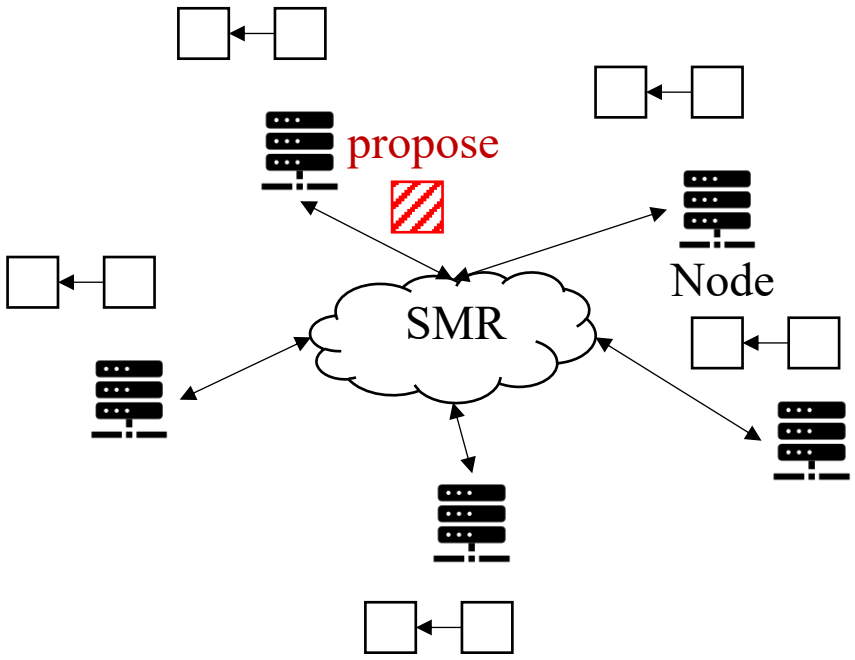
#NDSSSymposium2024

Blockchain

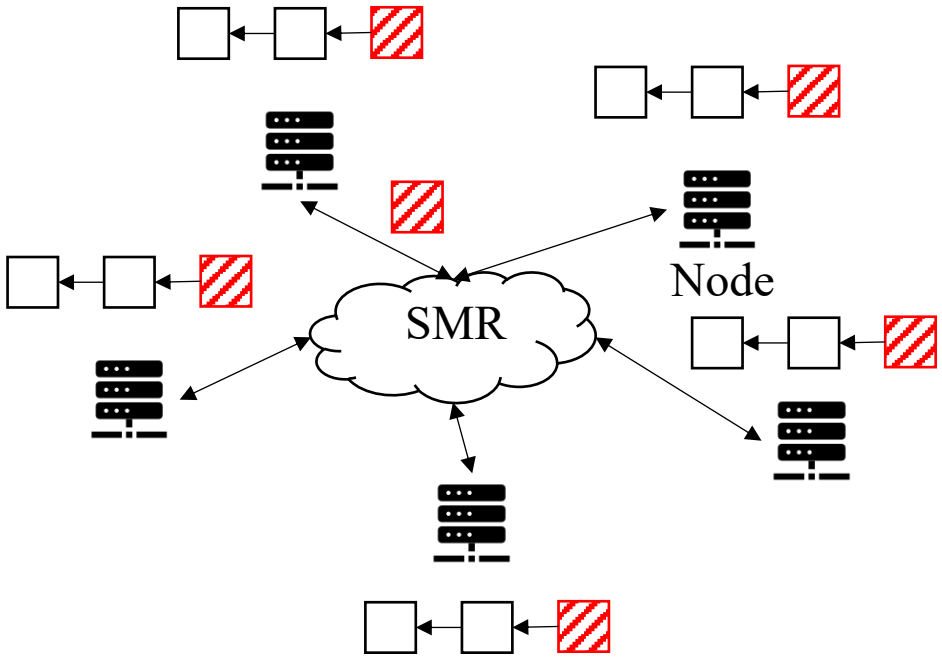
Blockchain: a State Machine Replication (SMR)



Blockchain



Blockchain

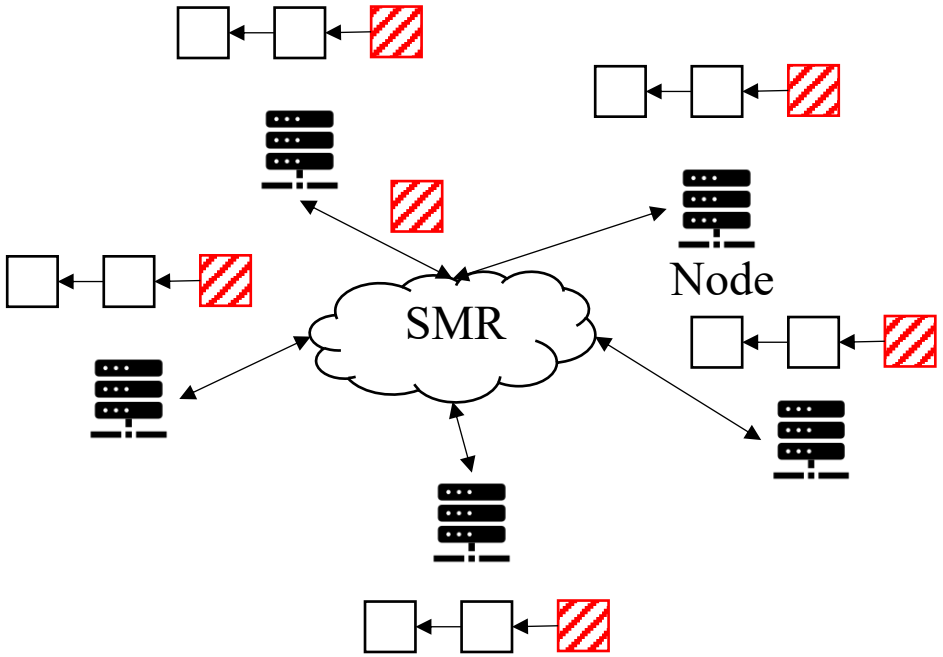


Scalability problem

Ideal scalability: more nodes, higher throughput

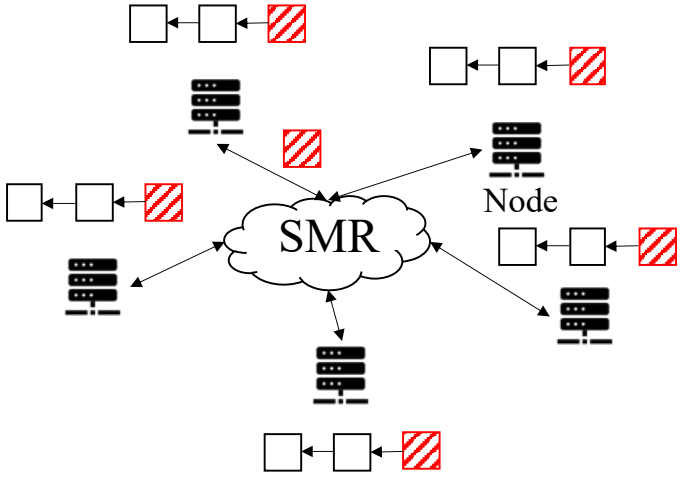
Poor scalability because each node:

- processes all transactions
- stores the whole ledger



Blockchain sharding

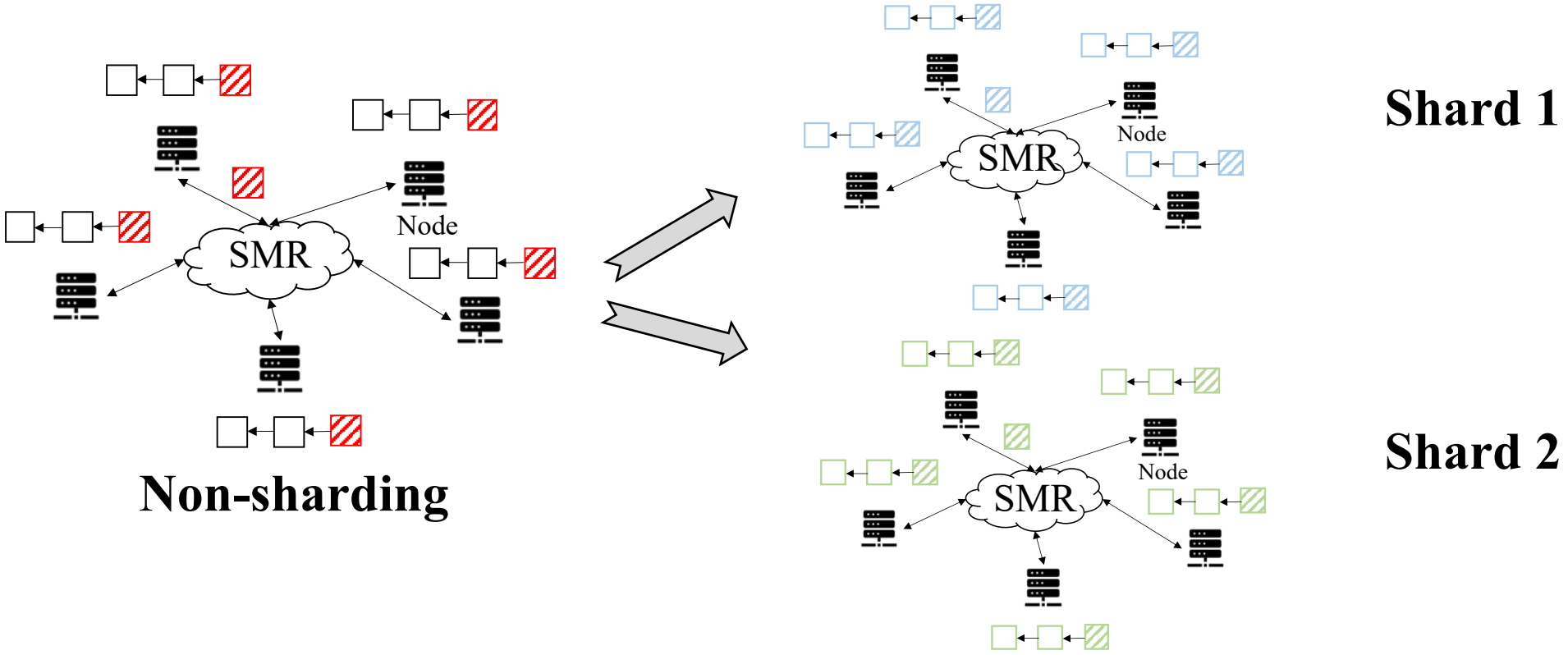
Blockchain sharding: sharding for parallel execution



Non-sharding

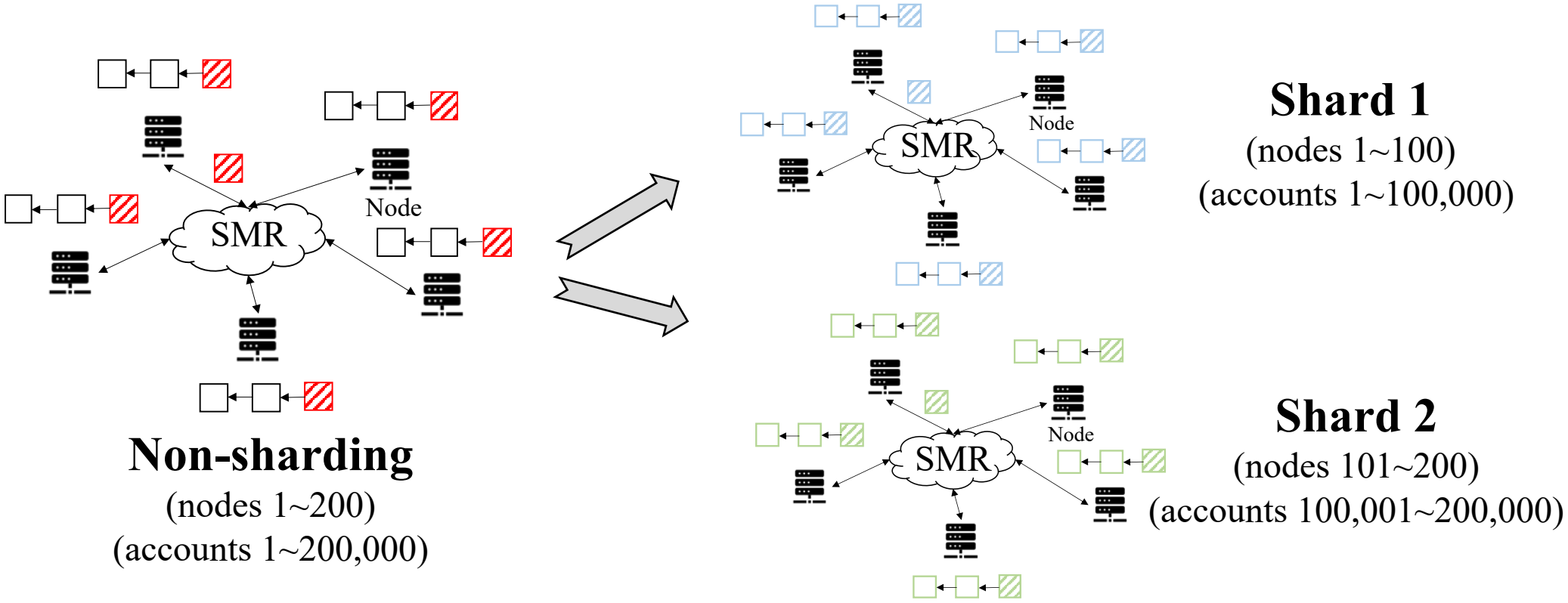
Blockchain sharding

Blockchain sharding: sharding for parallel execution



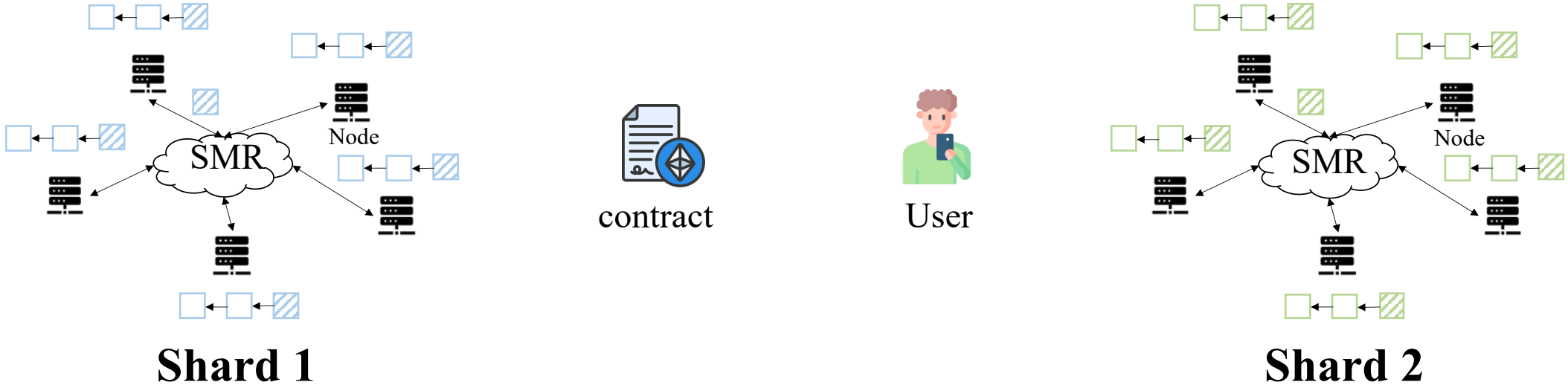
Blockchain sharding

Blockchain sharding: sharding for parallel execution



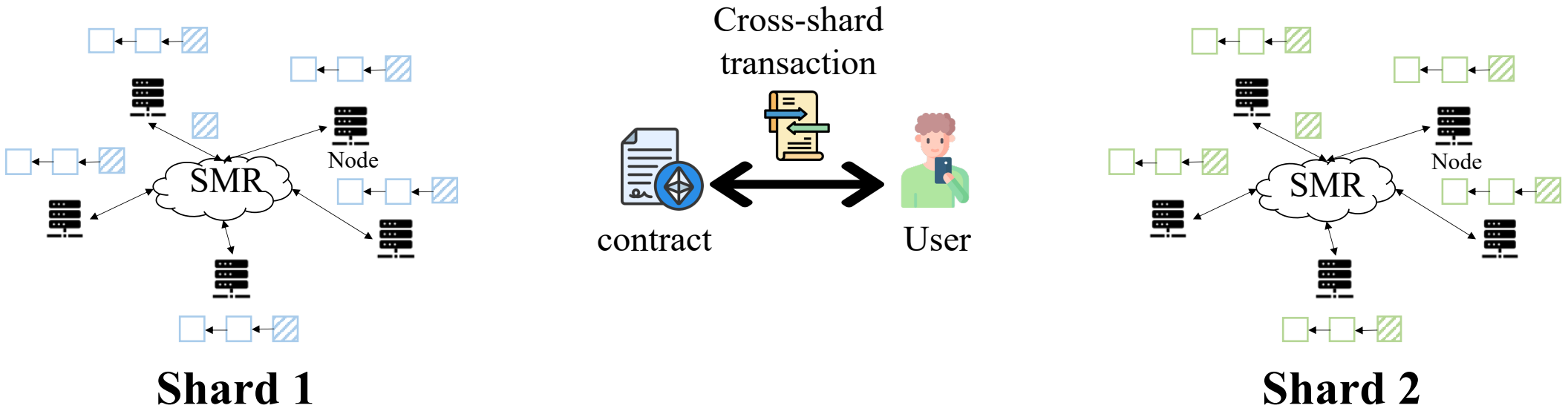
Cross-shard transactions

Transactions involve data from multiple shards



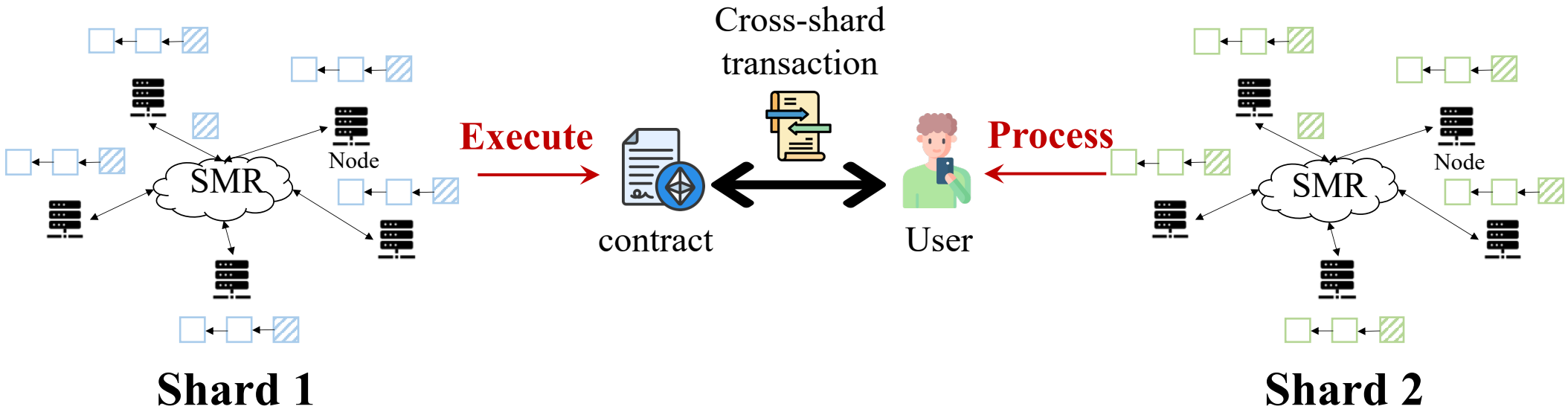
Cross-shard transactions

Transactions involve data from multiple shards



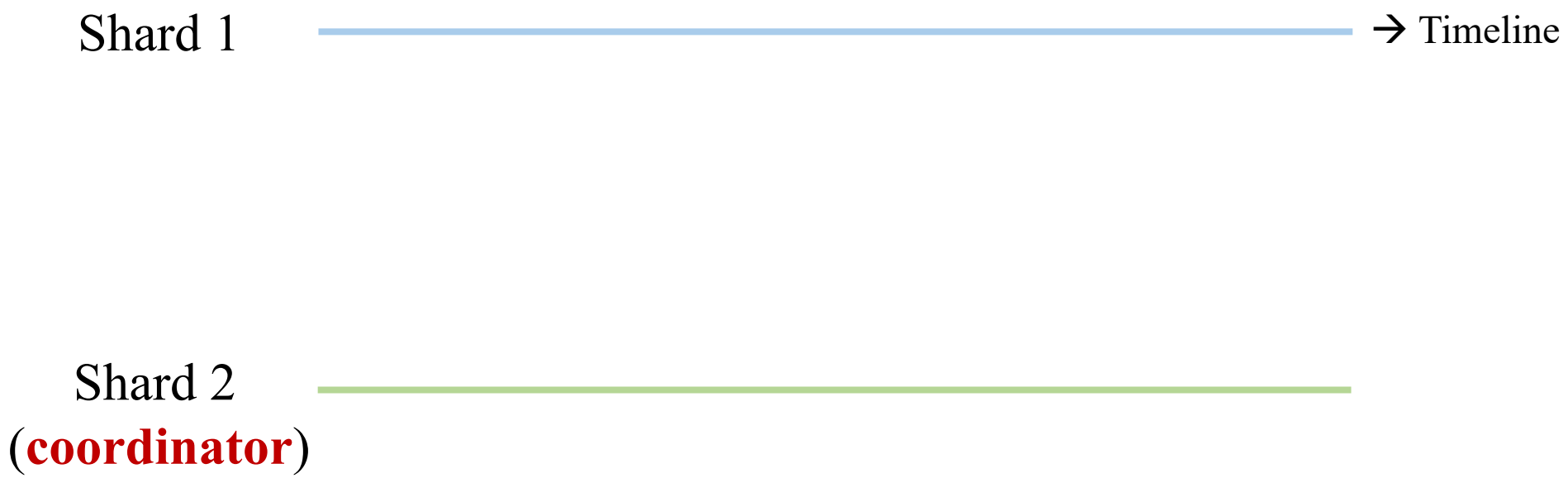
Cross-shard transactions

- **Process**: modify the account, e.g., withdraw coins to pay for the transaction
- **Execute**: change the state of the invoked smart contract



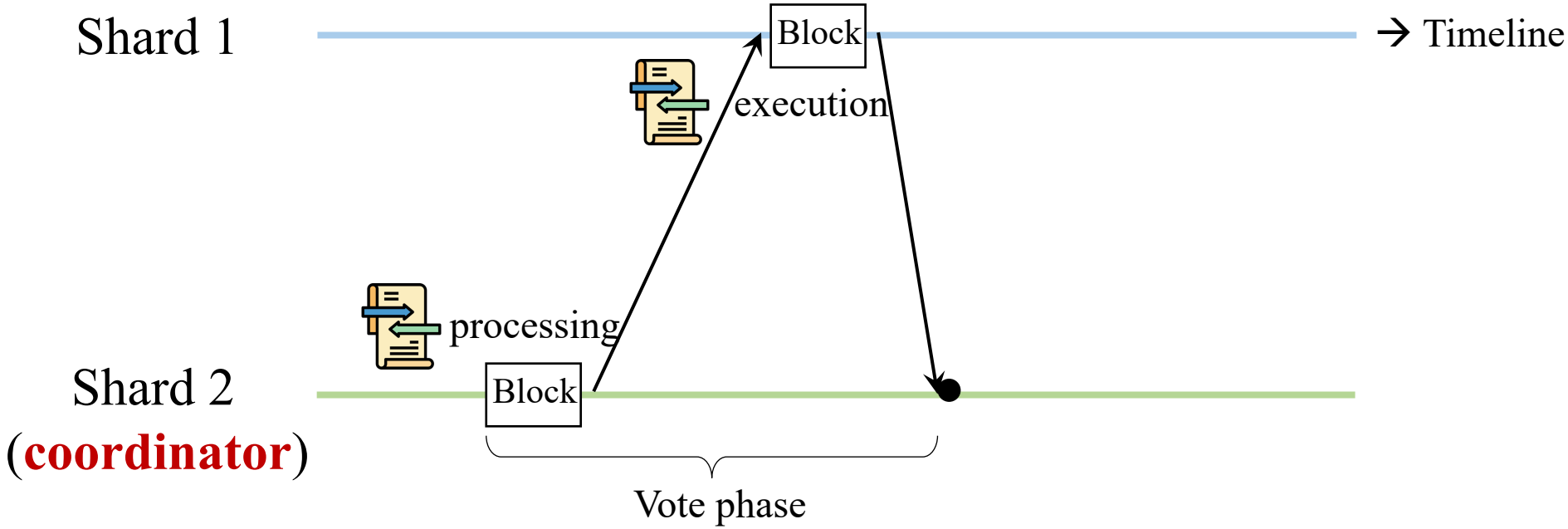
Cross-shard consensus

Two-phase commit: driven by a coordinator



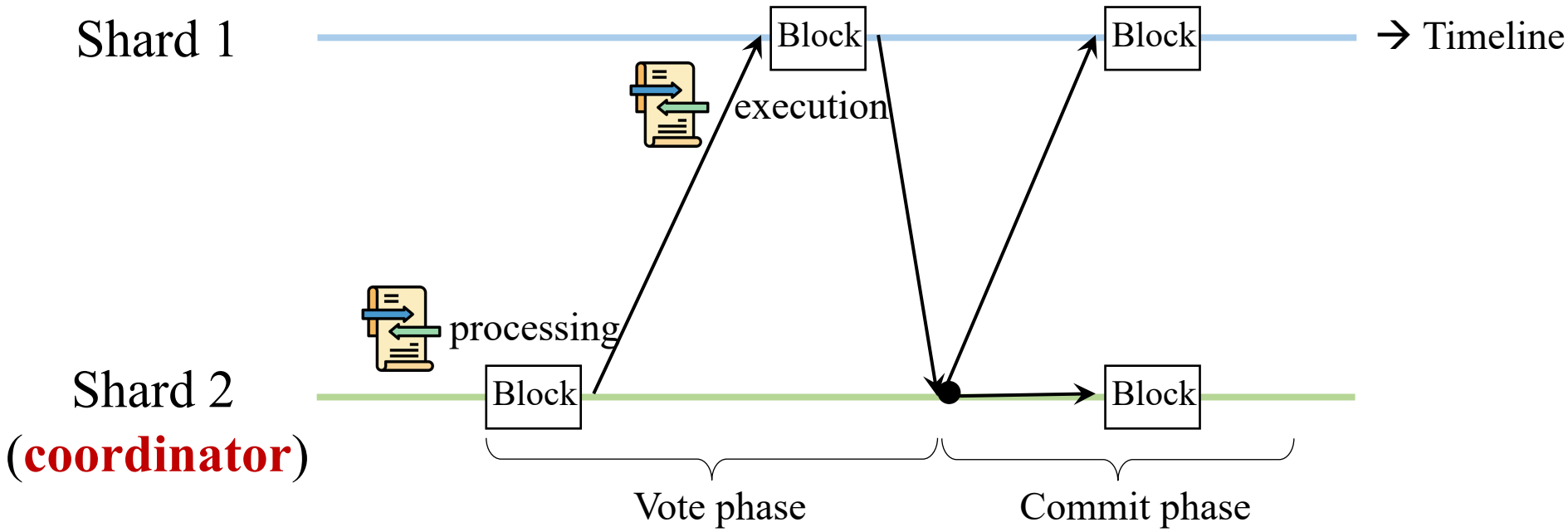
Two-phase commit

1. Vote phase: shards process and execute the transaction individually



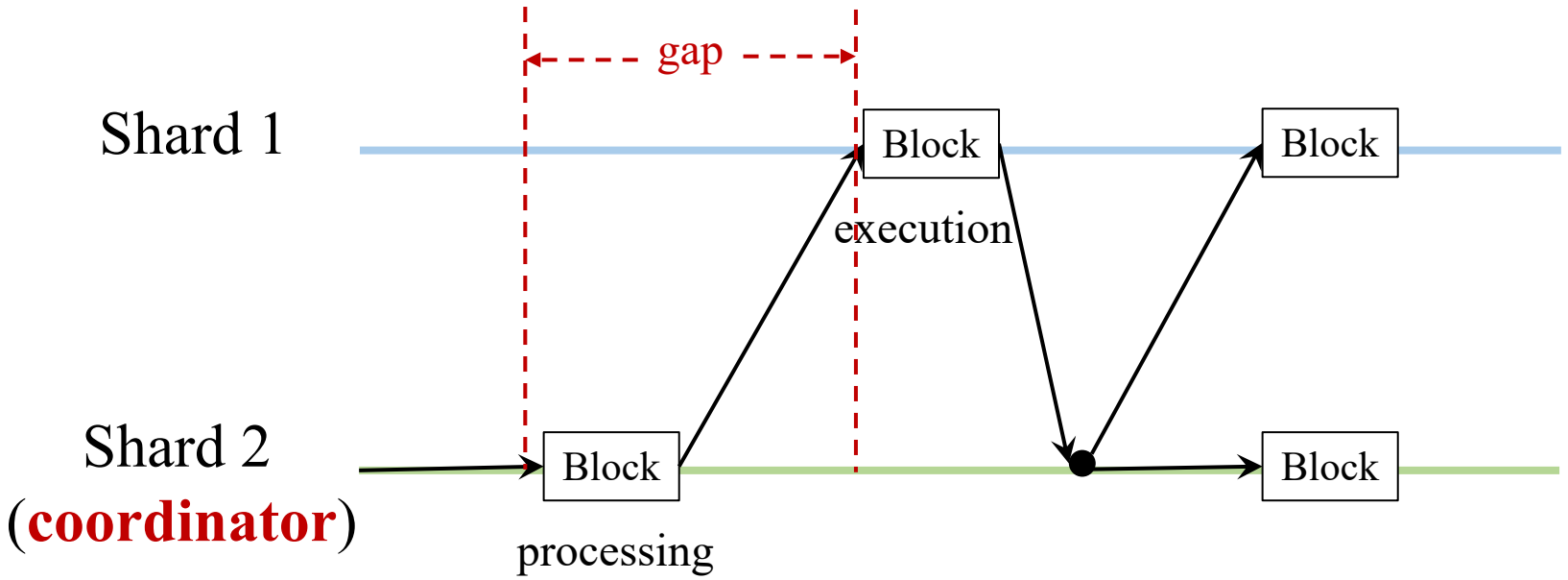
Two-phase commit

2. Commit phase: shards commit the transaction consistently



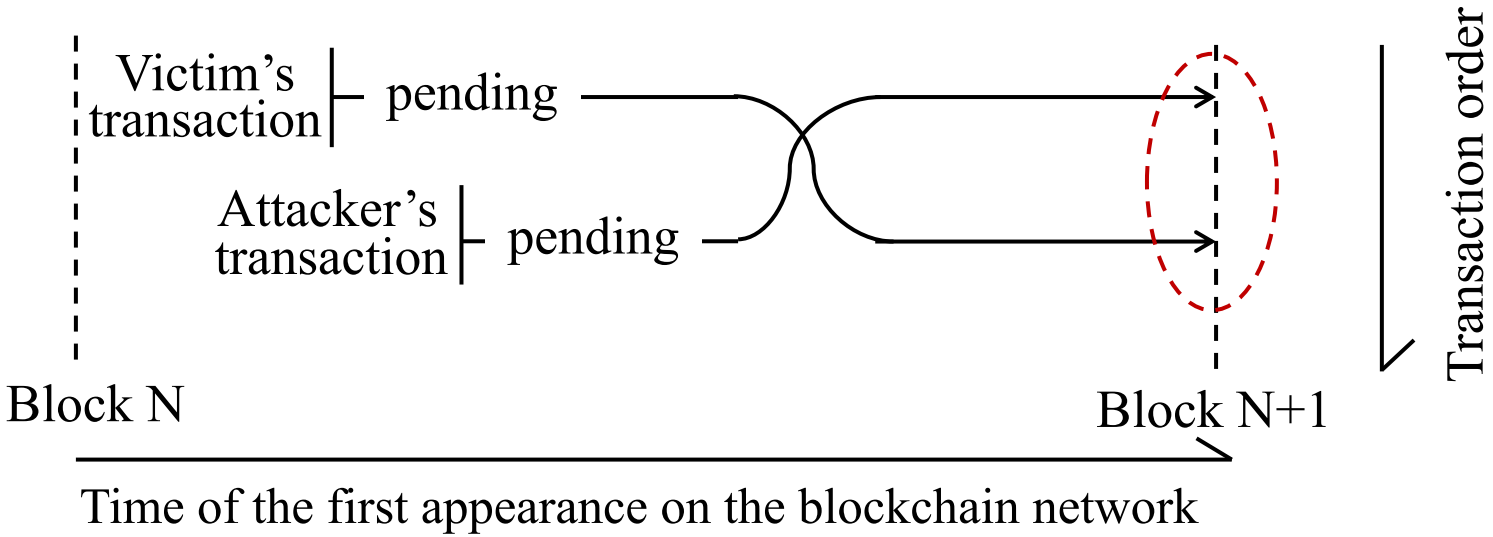
Observation – Process-execute gap

The two-phase commit **decouples** transaction **processing** and **execution**

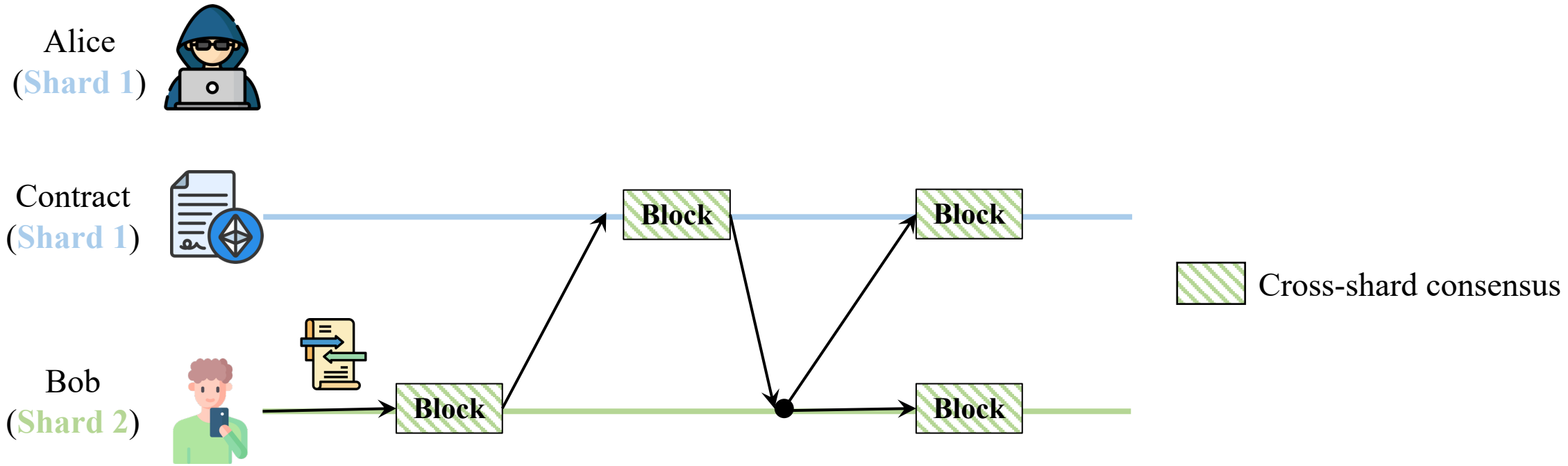


Definition – Front-running attacks

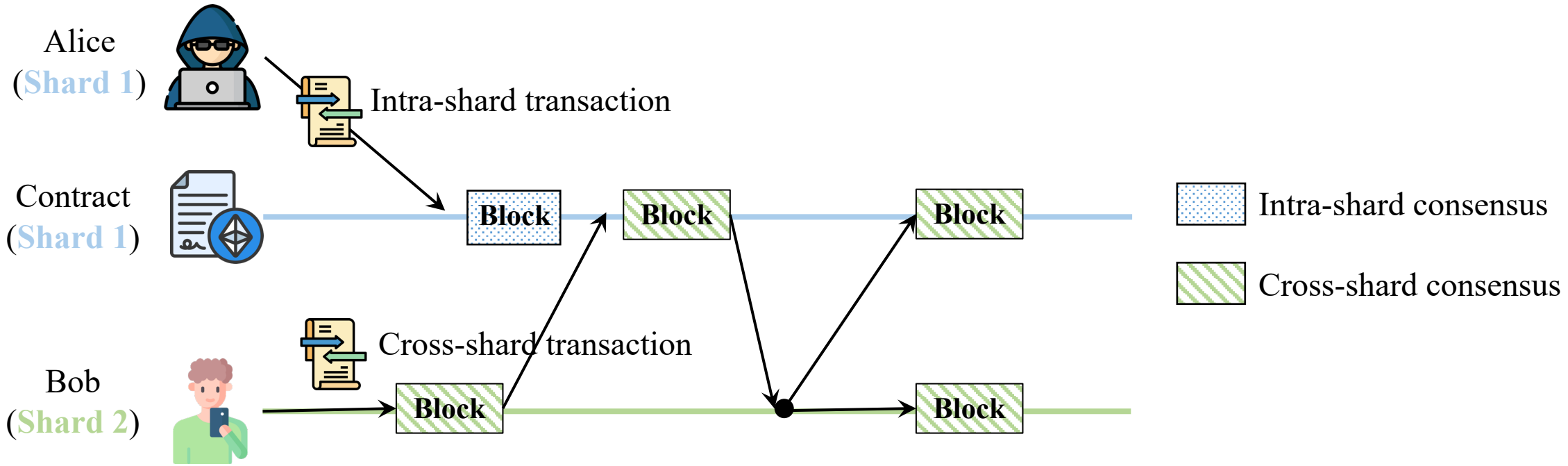
Attackers manipulate the transaction order



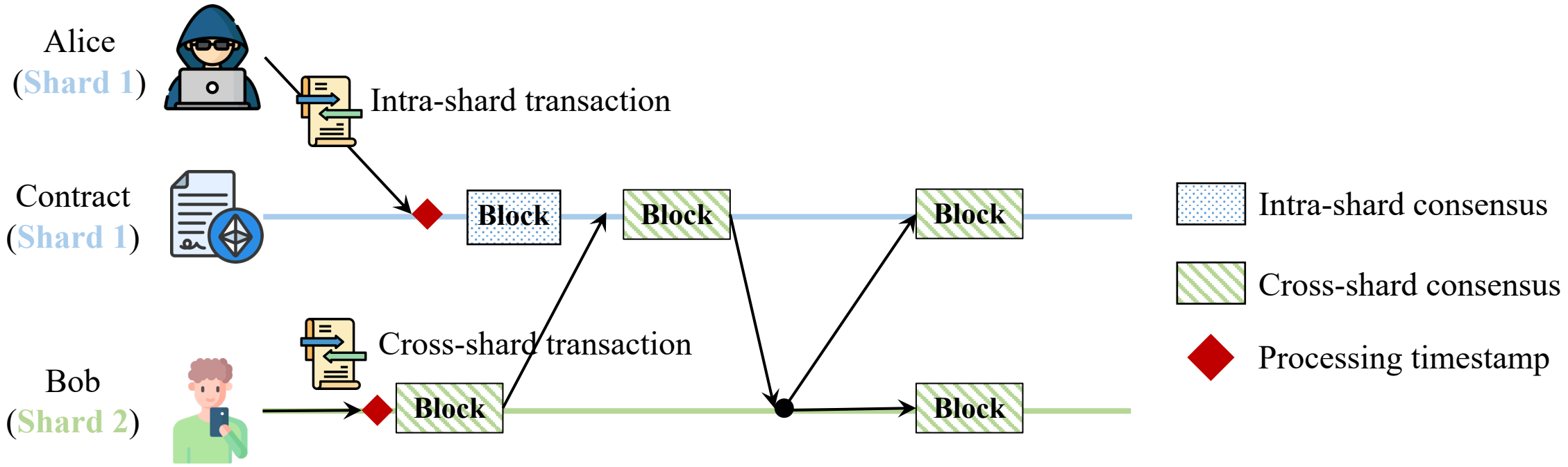
Motivation – Front-running attacks across shards



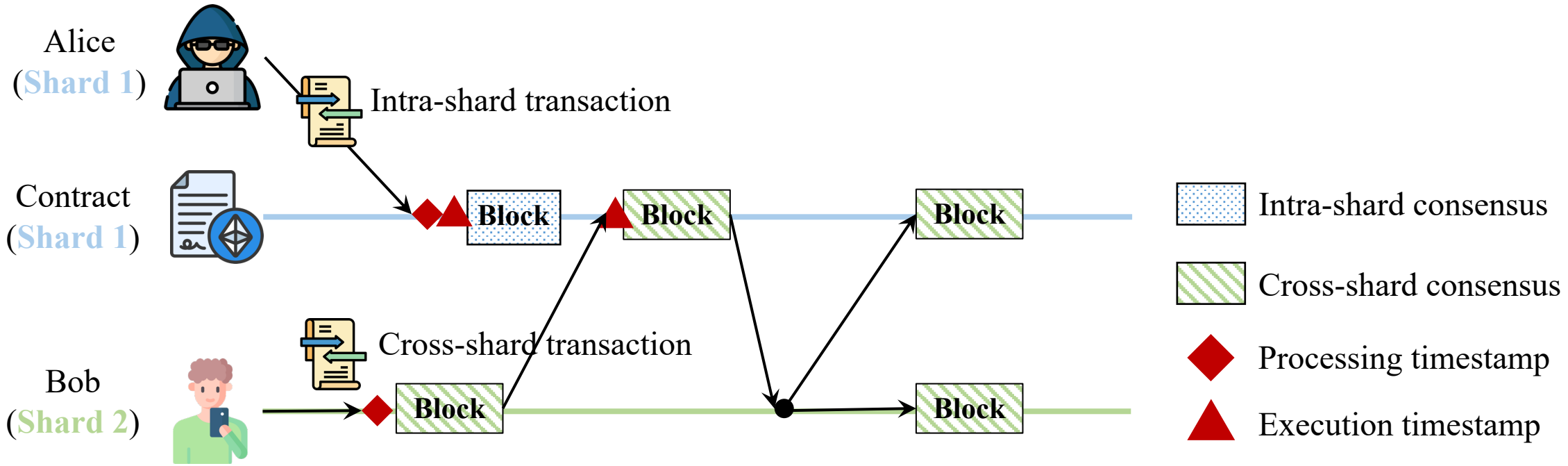
Motivation – Front-running attacks across shards



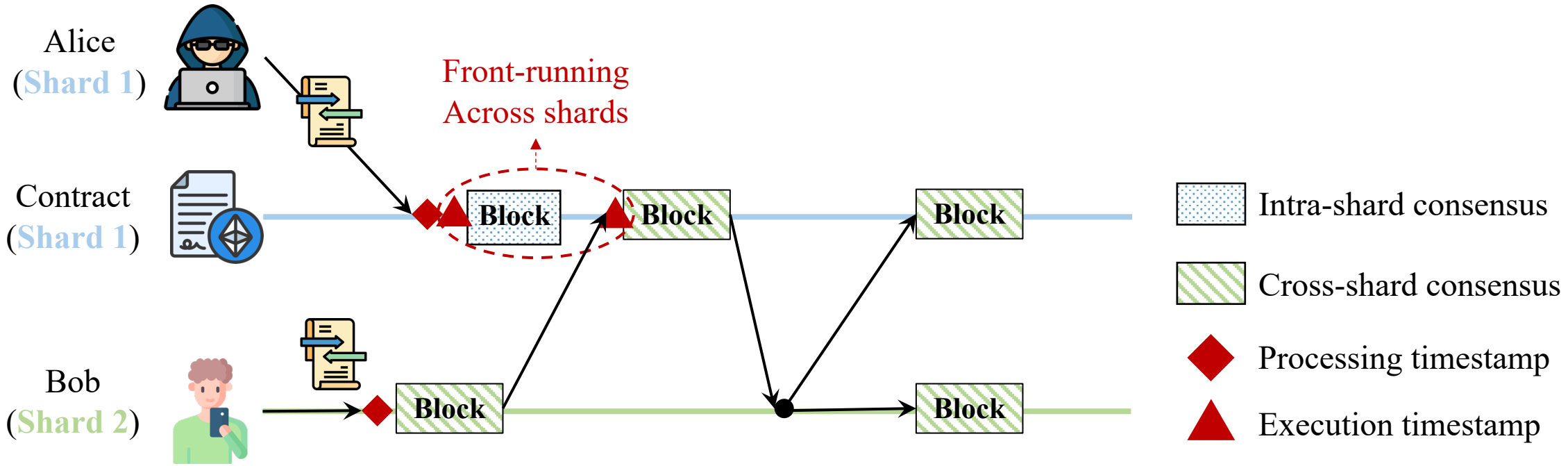
Motivation – Front-running attacks across shards



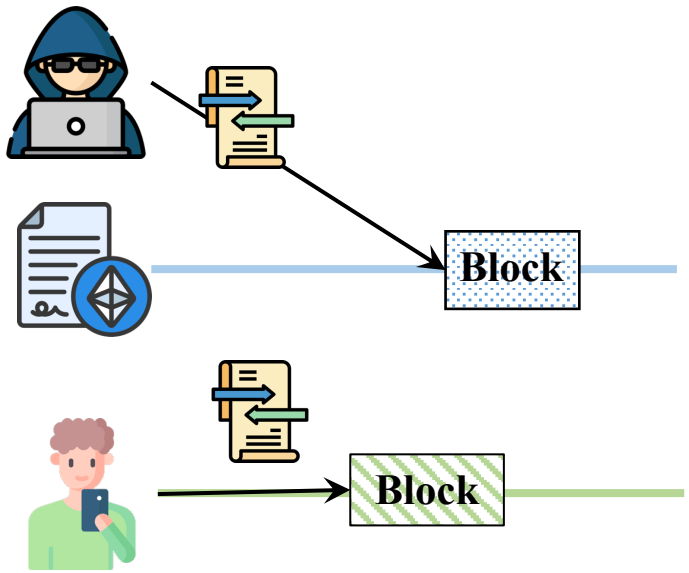
Motivation – Front-running attacks across shards



Motivation – Front-running attacks across shards

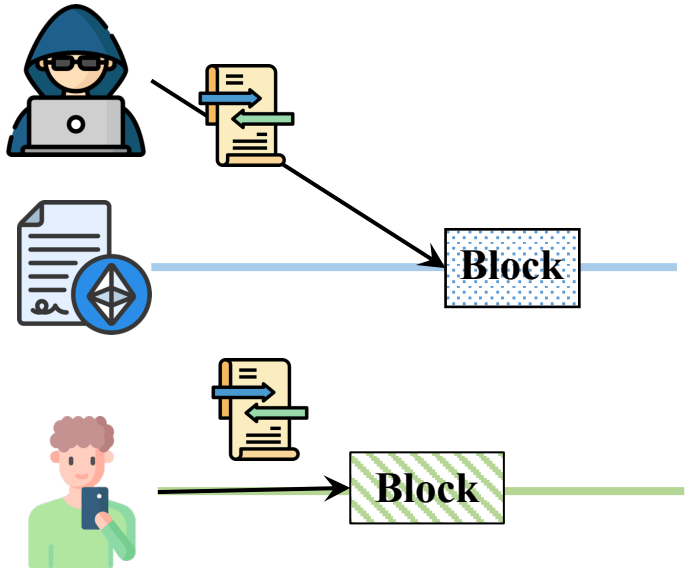


Problem statement – Why could happen



Problem statement – Why could happen

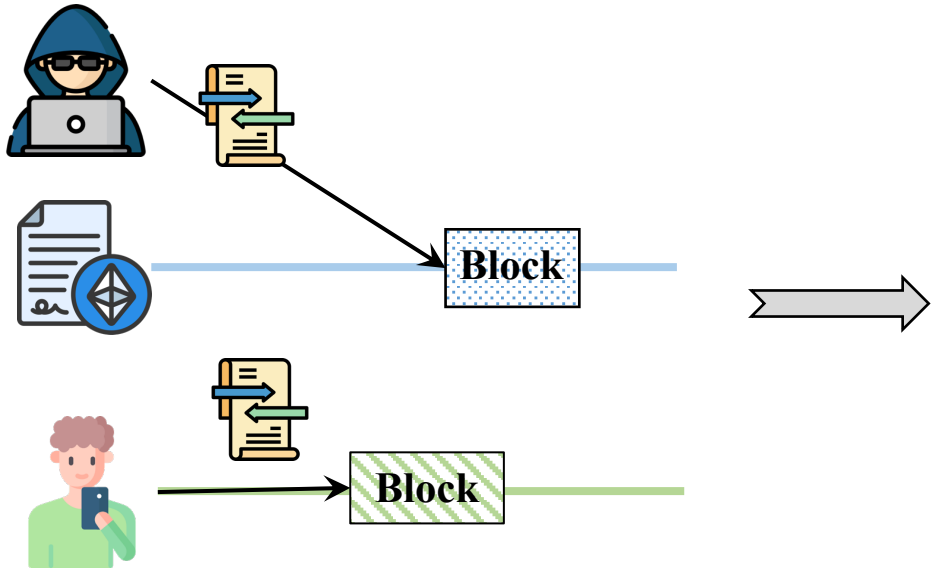
No order is defined between cross-shard and intra-shard transactions



Which one should be executed first?

Problem statement – Why could happen

No order is defined between cross-shard and intra-shard transactions

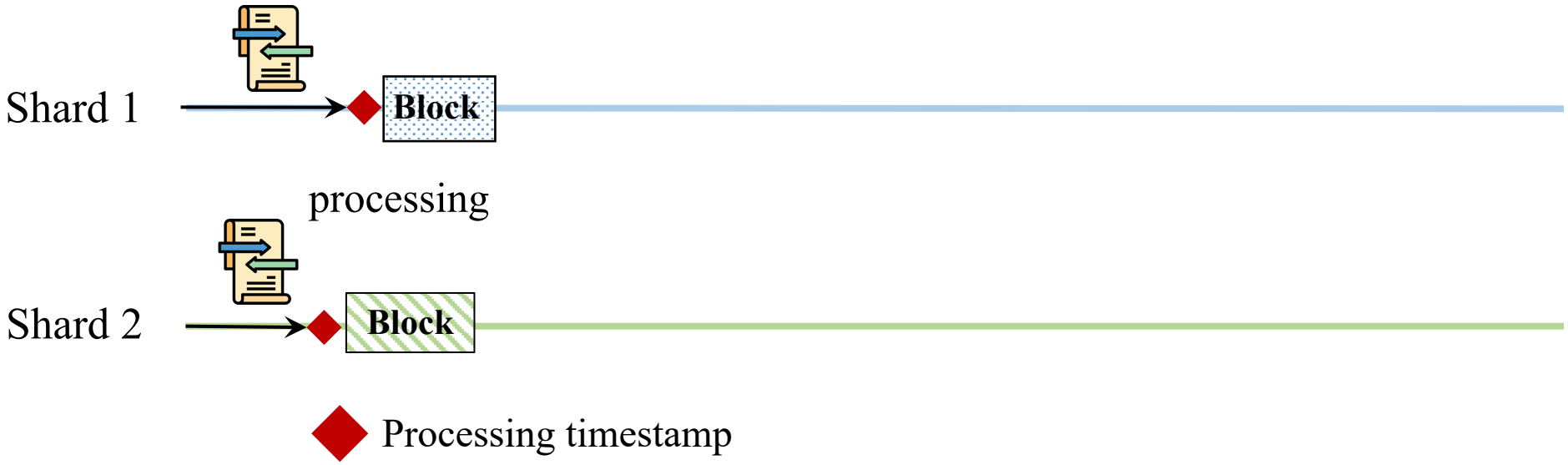


Which one should be executed first?

Finalization fairness: the **execution** order should be consistent with the **processing** order

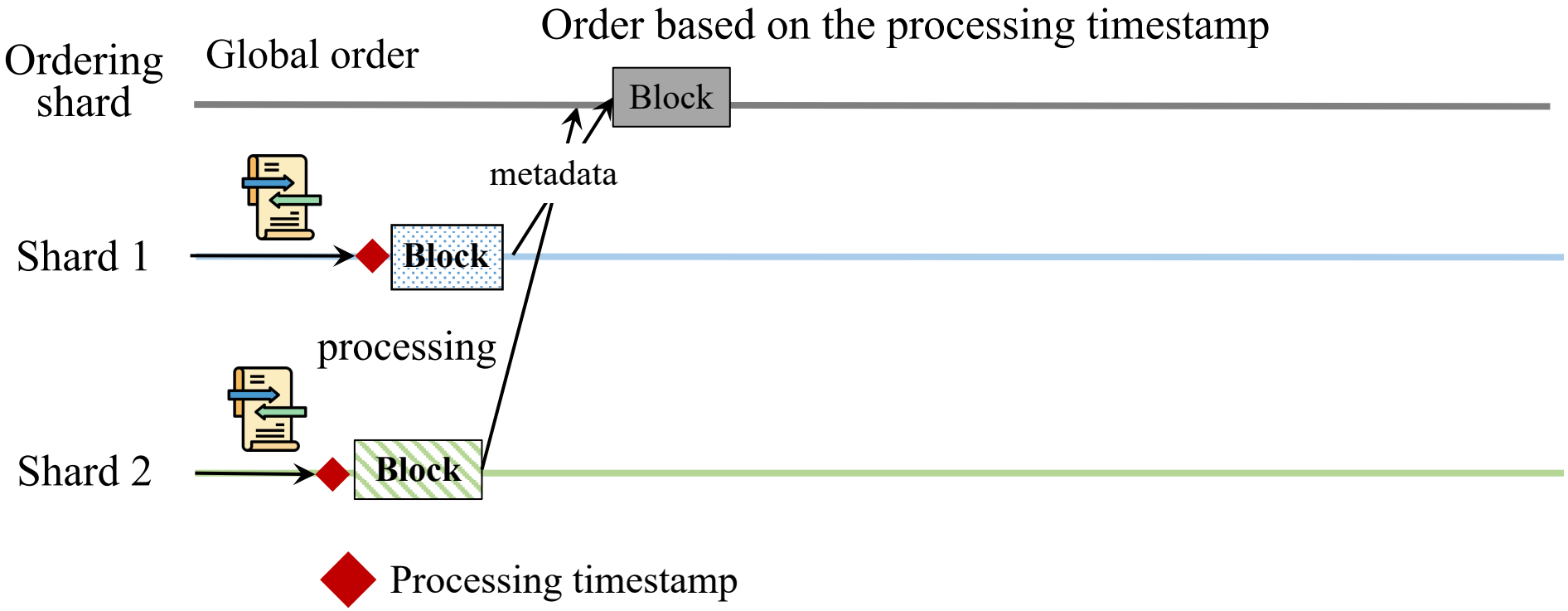
Main idea – An ordering phase

Goal: execution order = processing order



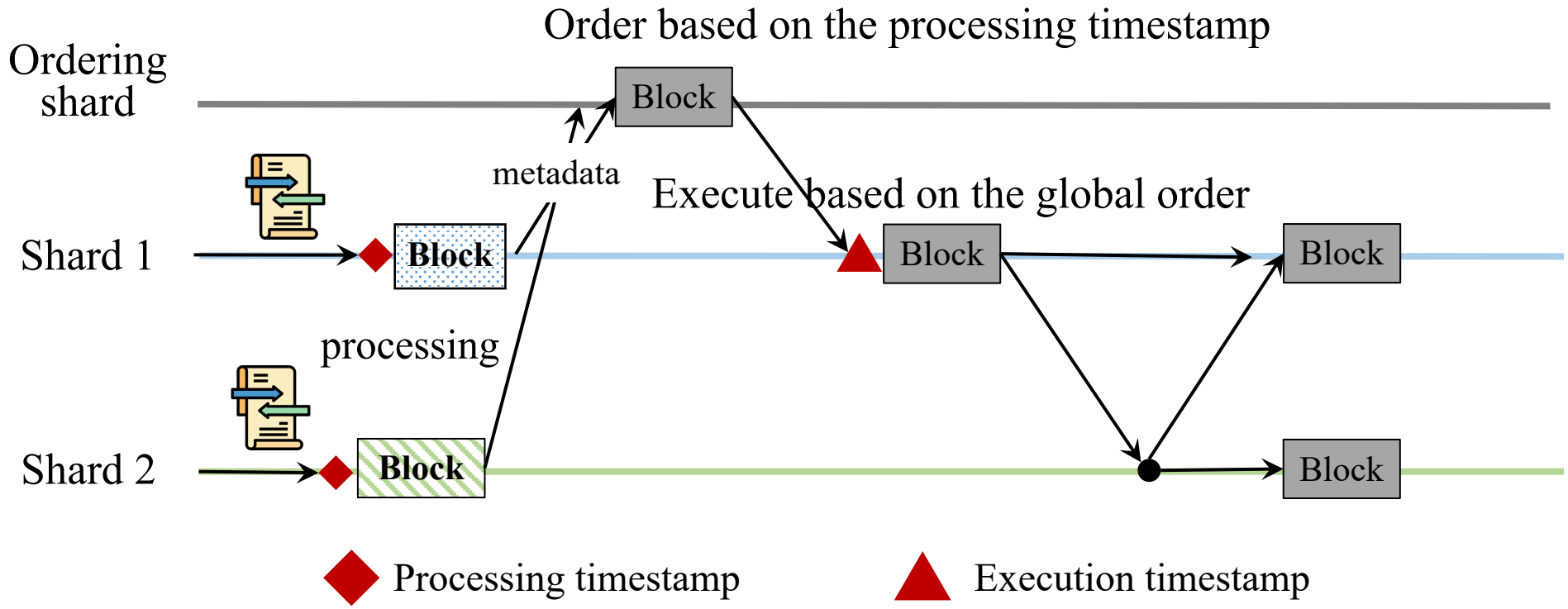
Main idea – An extra ordering phase

Ordering transactions globally before their executions



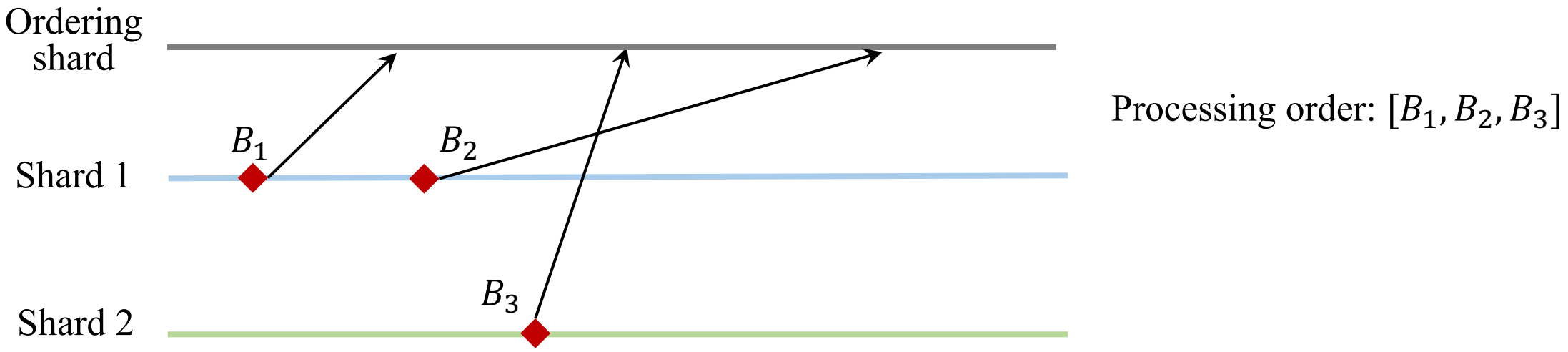
Main idea – An extra ordering phase

Ordering transactions globally before their executions



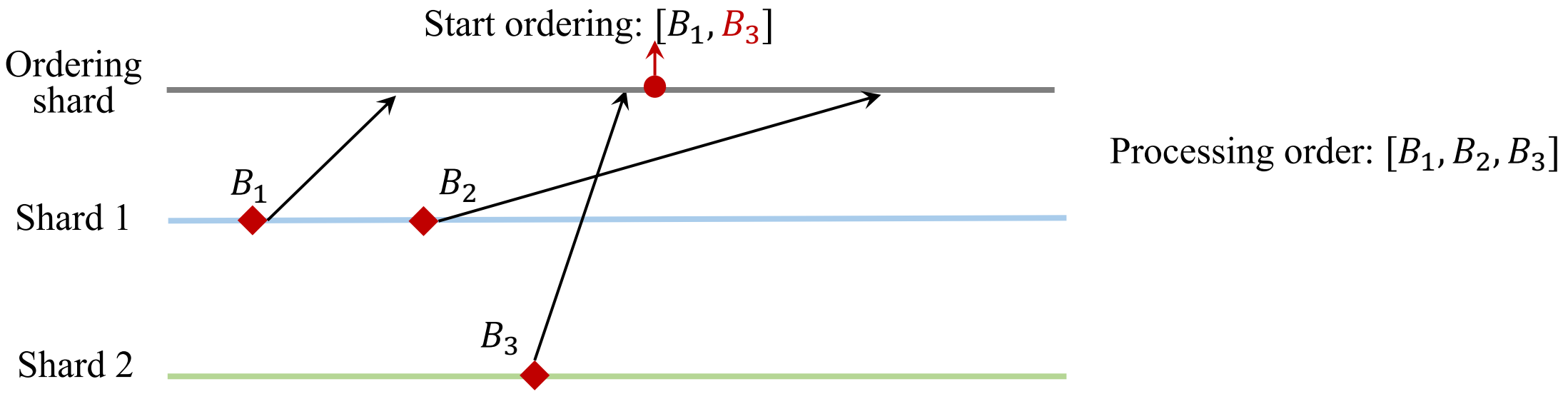
Challenge – When to order

Chaotic received order due to the asynchronous network



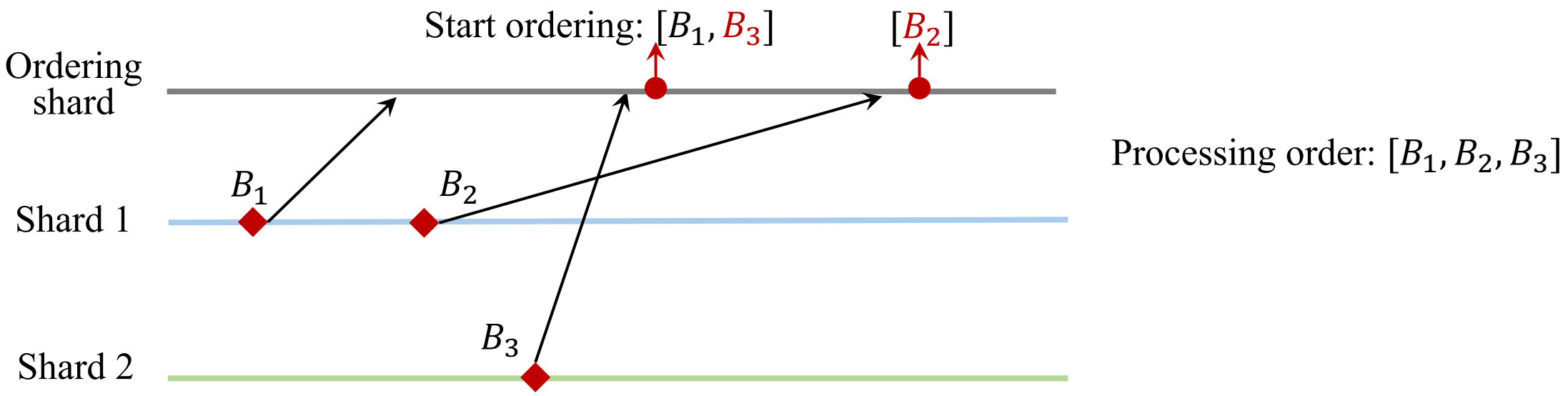
Challenge – When to order

Chaotic received order due to the asynchronous network



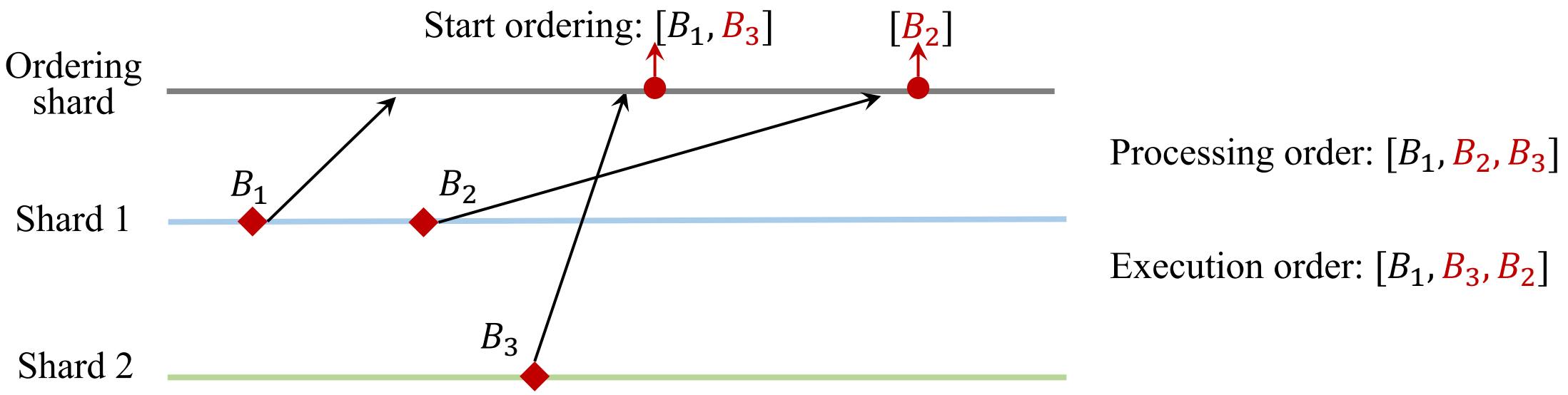
Challenge – When to order

Chaotic received order due to the asynchronous network



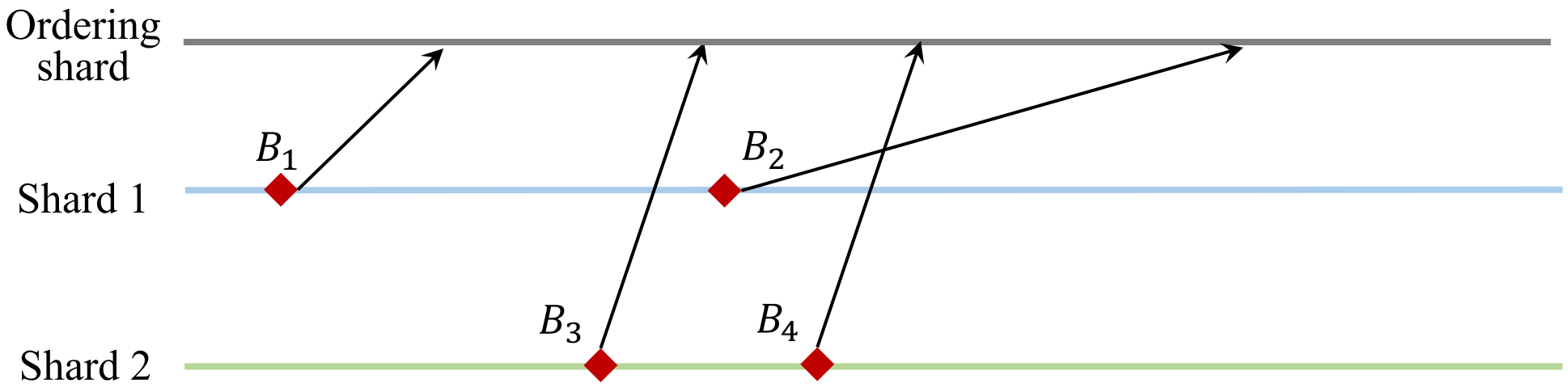
Challenge – When to order

Chaotic received order due to the asynchronous network



Solution – Haechi

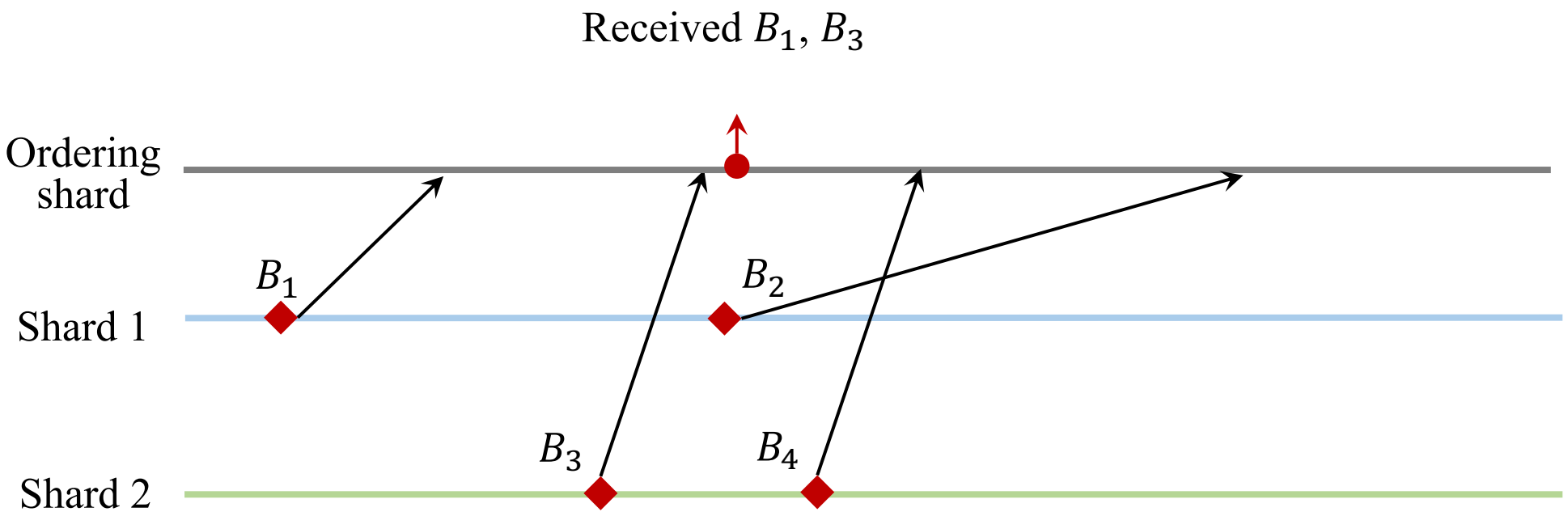
Asynchronous ordering solution



Solution – Haechi

At-least-one policy:

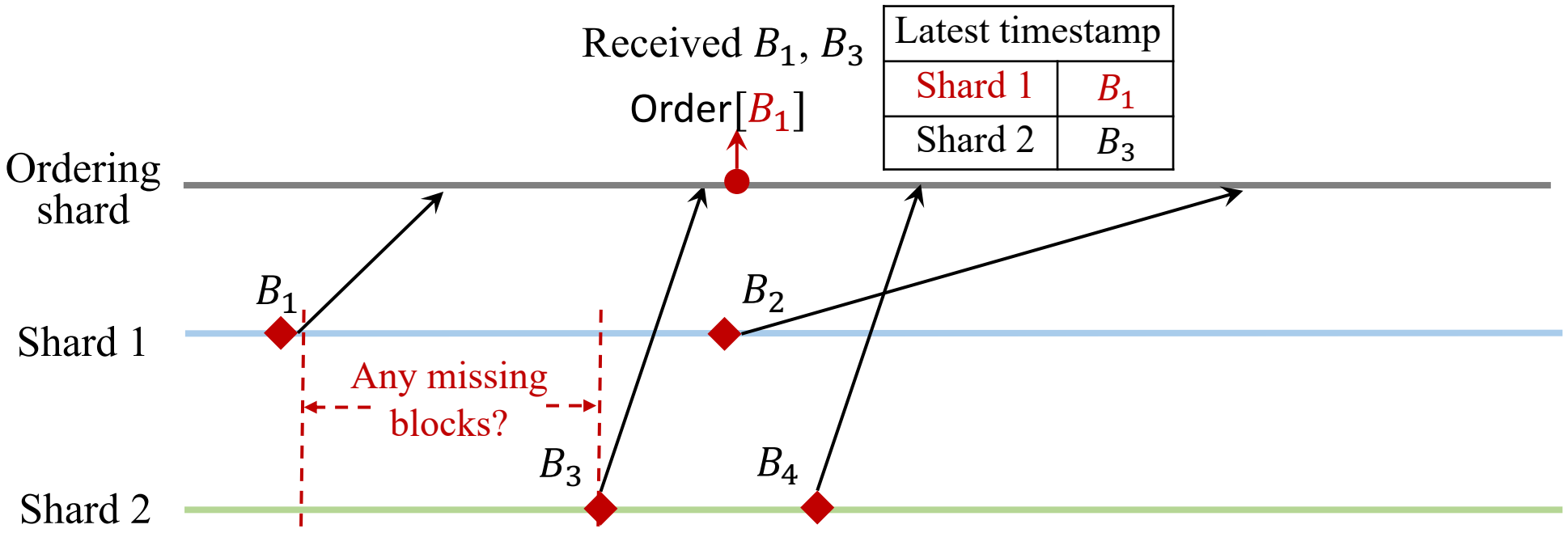
Start ordering only after receiving at **least one block** from all shards



Solution – Haechi

Ordering policy

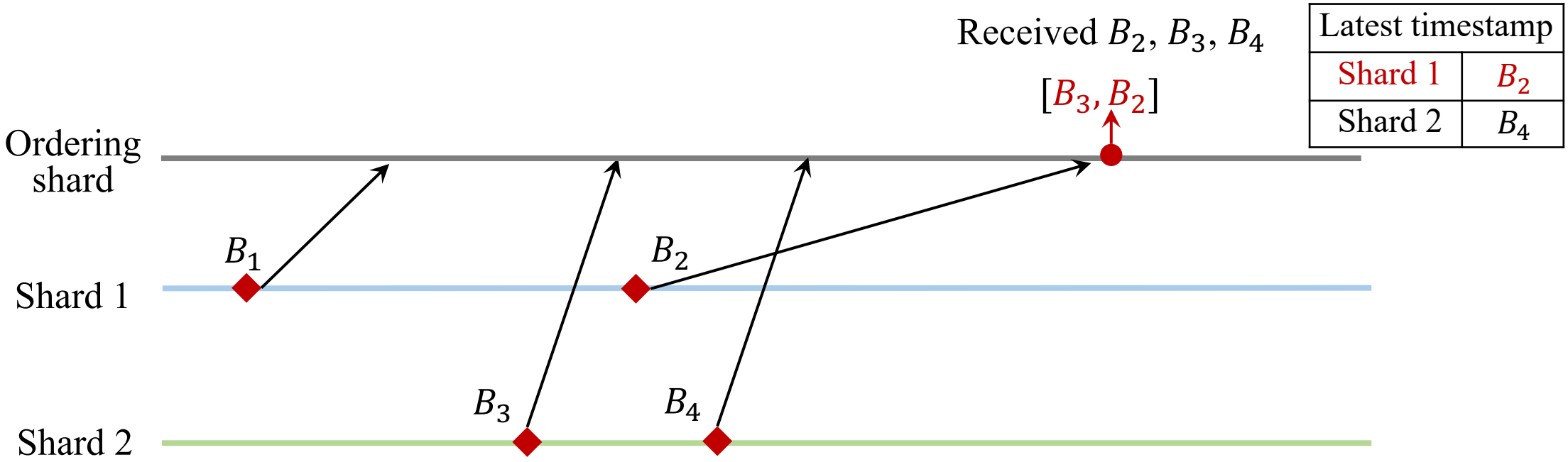
Order blocks before the **minimum** timestamp of all latest received timestamps from all shards



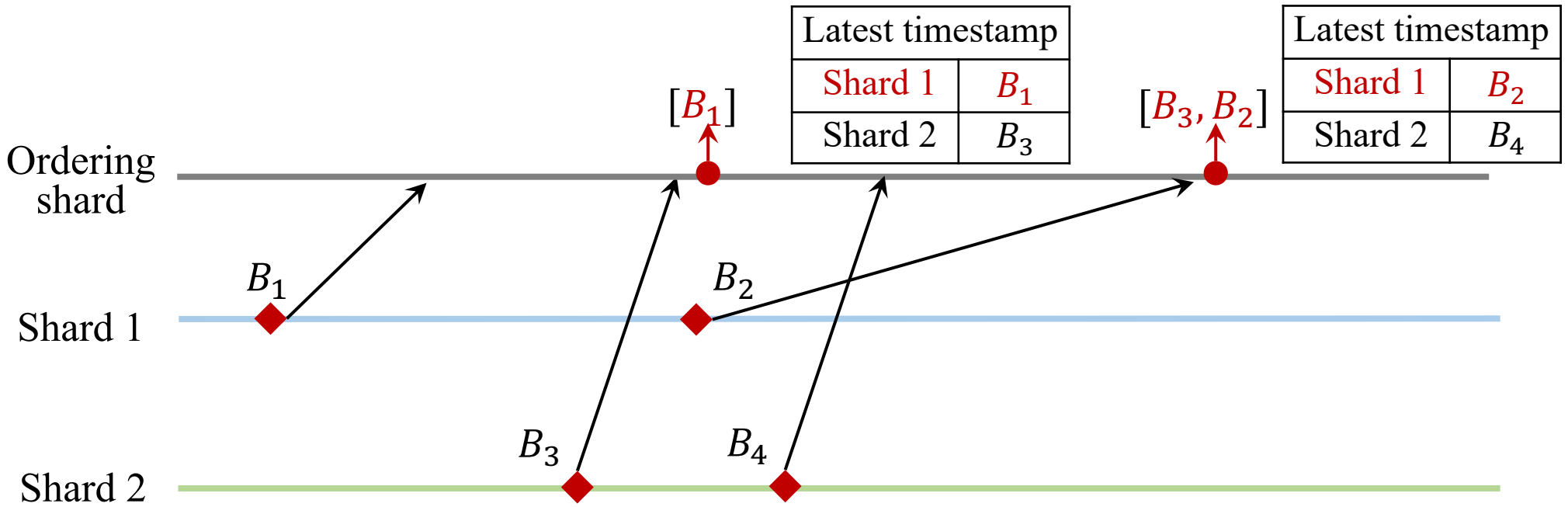
Solution – Haechi

Ordering policy

Order blocks before the **minimum** timestamp of all latest received timestamps from all shards



Solution – Haechi

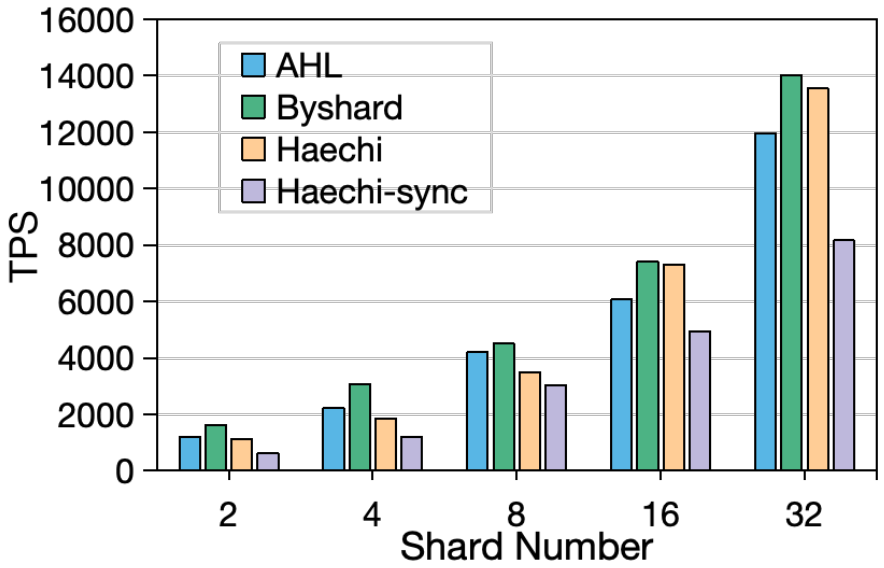


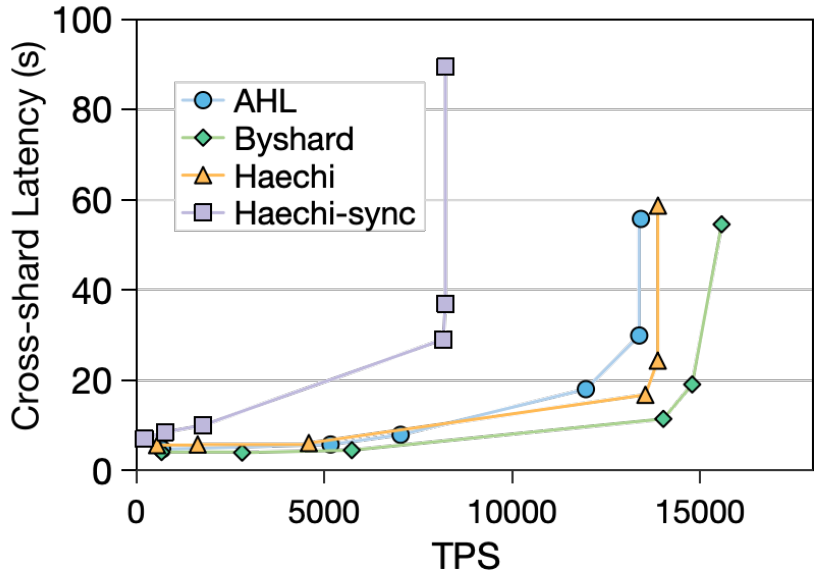
Evaluations

Implementation: based on Tendermint

Setting: 990 nodes running in AWS EC2 instances

Comparison: AHL[1] (single-coordinator), Byshard[2] (multi-coordinator)

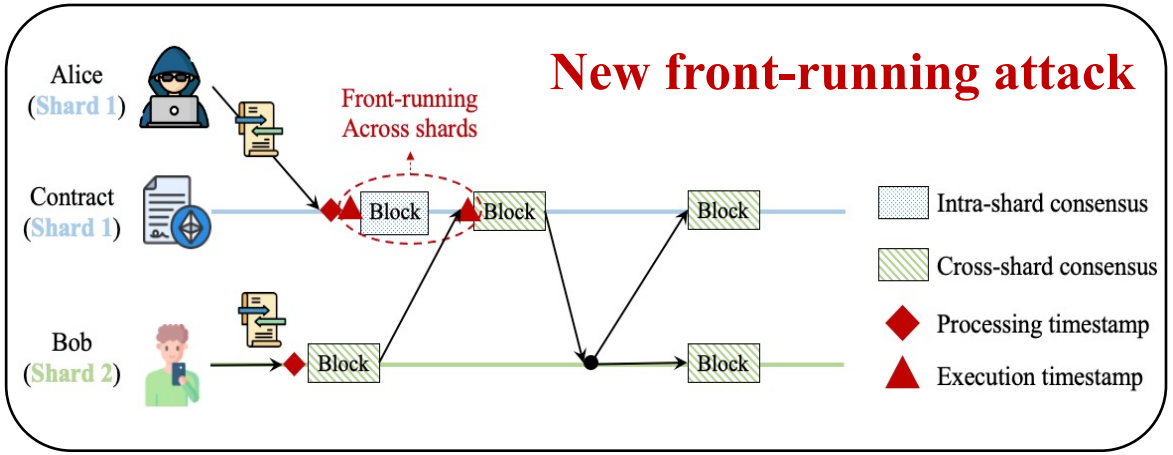




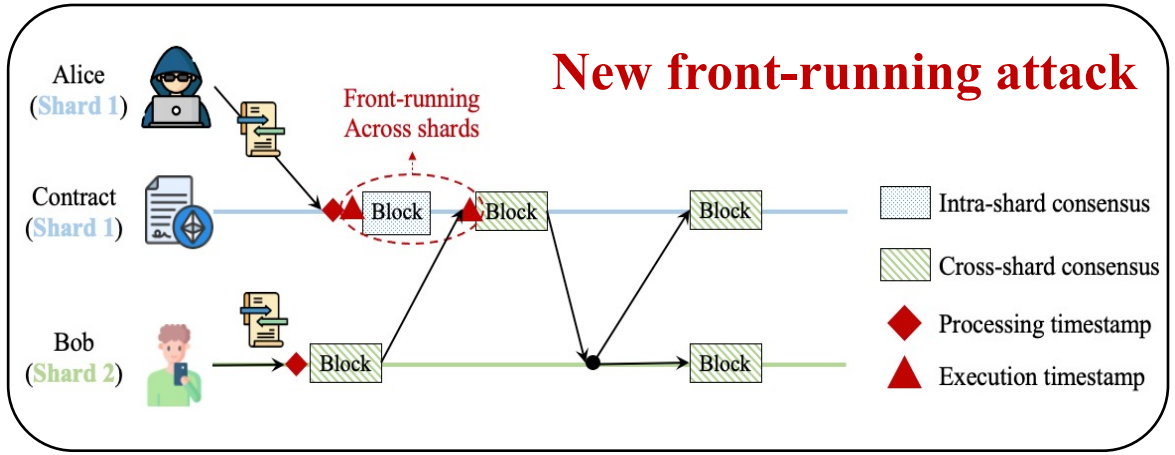
[1] Dang, Hung, et al. *Towards scaling blockchain systems via sharding*. SIGMOD. 2019.

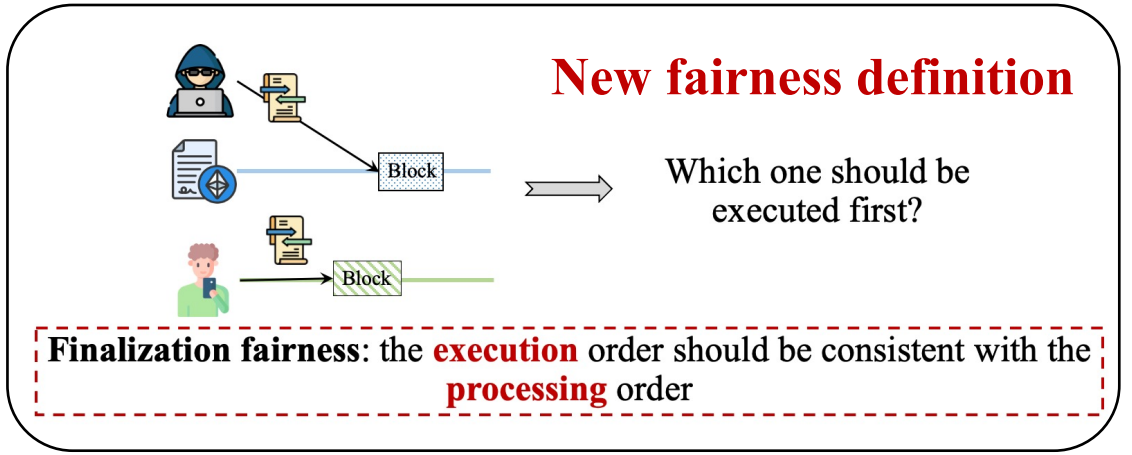
[2] Hellings, Jelle, et al. *Byshard: Sharding in a byzantine environment*. VLDB. 2021.

Takeaway

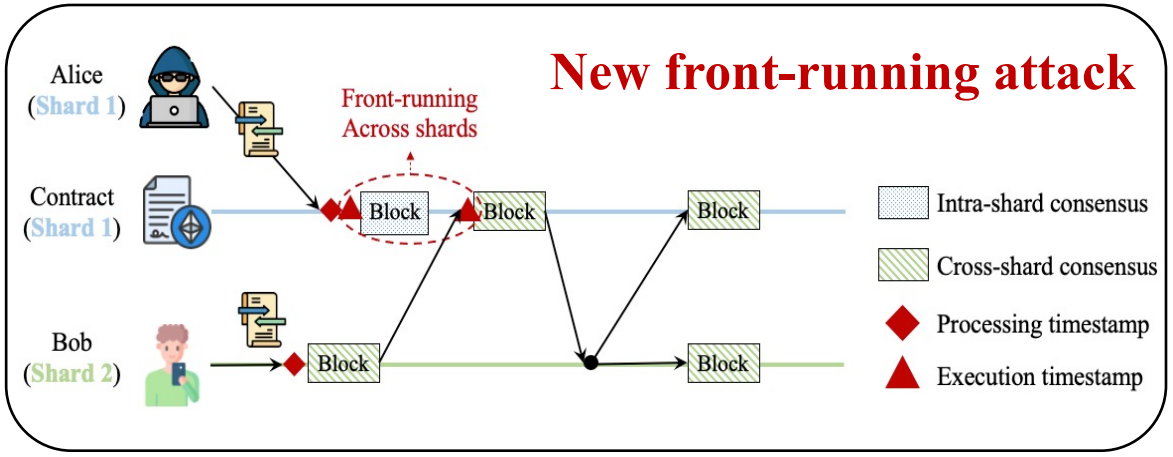

New front-running attack
Alice
(Shard 1)
Front-running
Across shards
Contract
(Shard 1)
Block
Block
Block
Bob
(Shard 2)
Block
Block
Intra-shard consensus
Cross-shard consensus
Processing timestamp
Execution timestamp

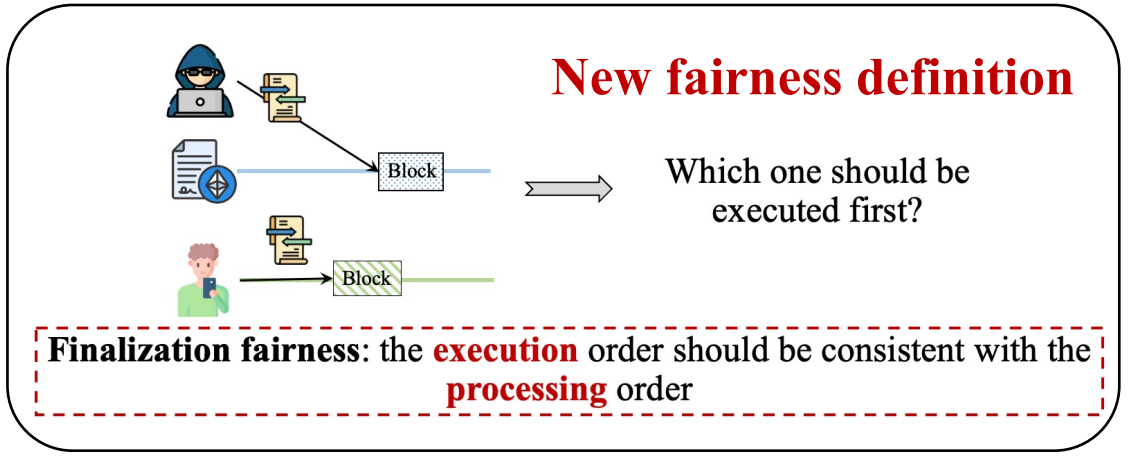
Takeaway

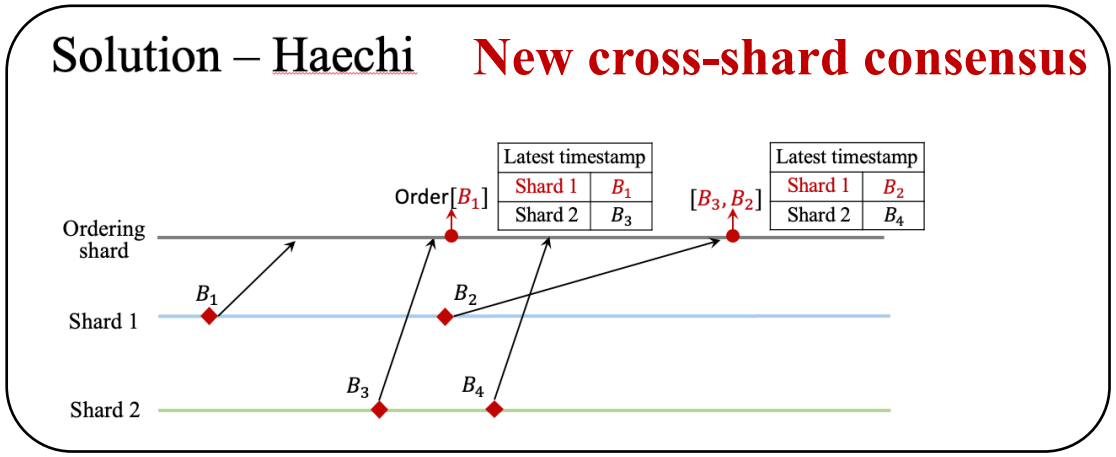

New front-running attack
Alice
(Shard 1)
Contract
(Shard 1)
Bob
(Shard 2)
Front-running
Across shards
Block
Block
Block
Block
Block
Intra-shard consensus
Cross-shard consensus
Processing timestamp
Execution timestamp


New fairness definition
Block
Block
Which one should be
executed first?
Finalization fairness: the execution order should be consistent with the processing order

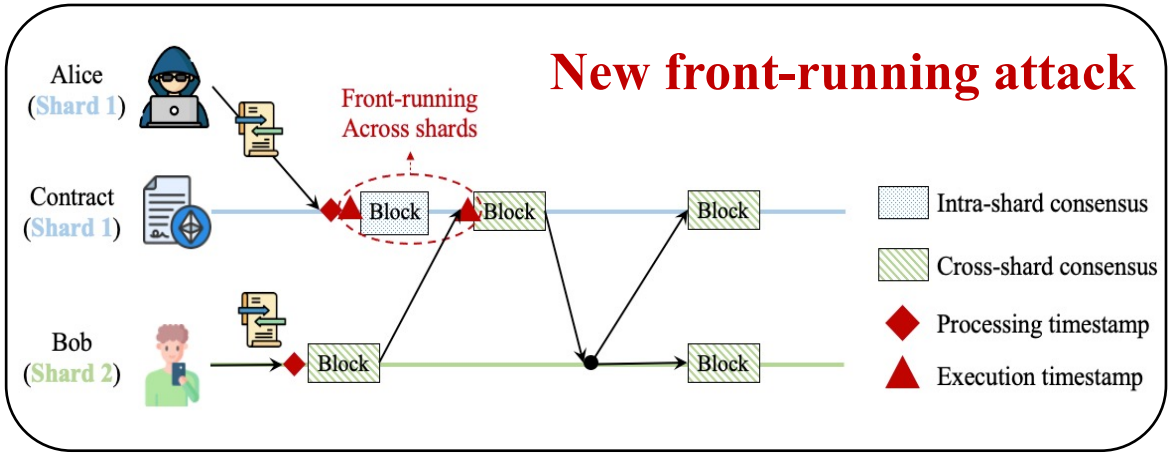
Takeaway

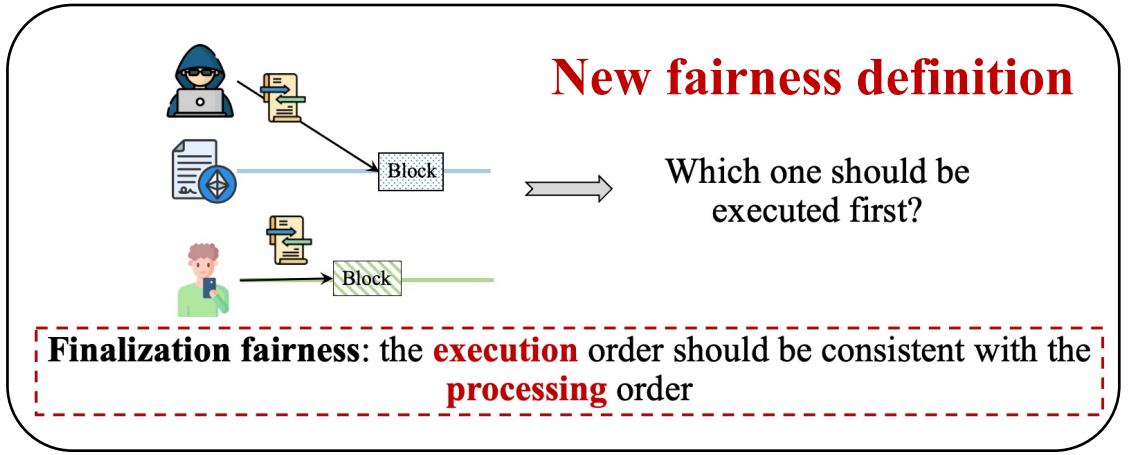

New front-running attack
Alice
(Shard 1)
Front-running
Across shards
Contract
(Shard 1)
Bob
(Shard 2)
Block
Block
Block
Block
Block
Intra-shard consensus
Cross-shard consensus
Processing timestamp
Execution timestamp

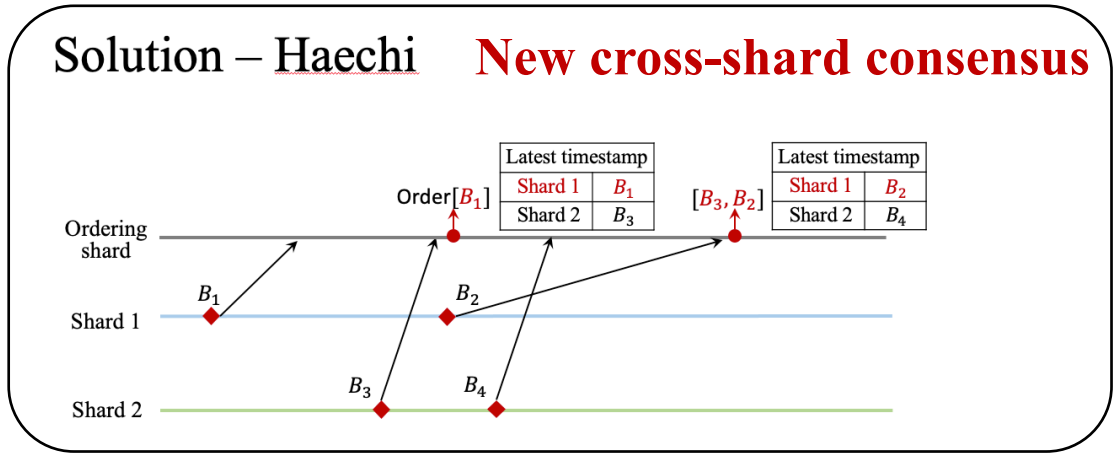

New fairness definition
Block
Block
Which one should be
executed first?
Finalization fairness: the execution order should be consistent with the processing order

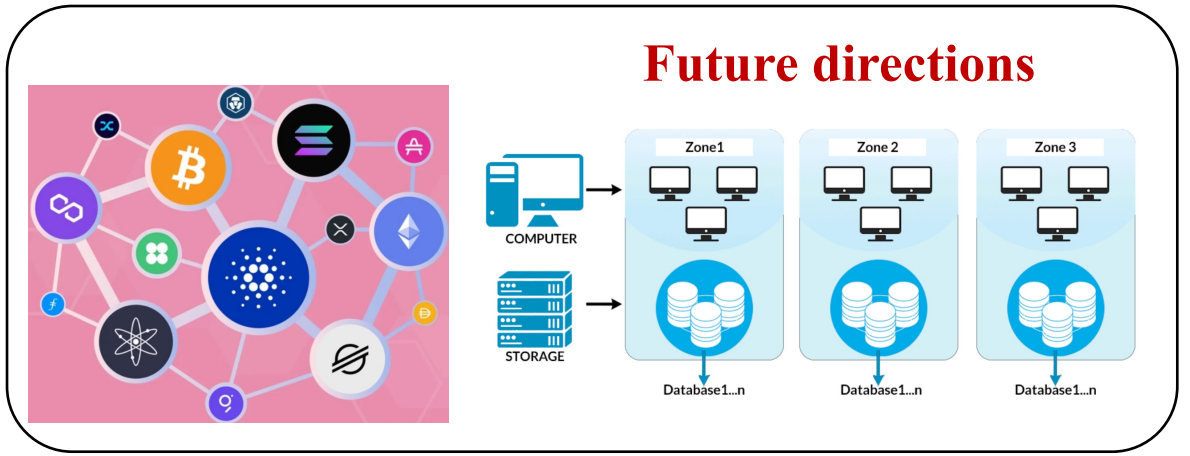

Solution – Haechi
New cross-shard consensus
Latest timestamp
Shard 1 B_1
Shard 2 B_3
Order$[B_1]$
$[B_3, B_2]$
Latest timestamp
Shard 1 B_2
Shard 2 B_4
Ordering shard
Shard 1
Shard 2
B_1
B_2
B_3
B_4

Takeaway


New front-running attack
Alice
(Shard 1)
Front-running
Across shards
Contract
(Shard 1)
Bob
(Shard 2)
Block
Block
Block
Block
Block
Intra-shard consensus
Cross-shard consensus
Processing timestamp
Execution timestamp


New fairness definition
Block
Block
Which one should be
executed first?
Finalization fairness: the execution order should be consistent with the processing order


Solution – Haechi
New cross-shard consensus
Latest timestamp
Shard 1 B_1
Shard 2 B_3
Order$[B_1]$
$[B_3, B_2]$
Latest timestamp
Shard 1 B_2
Shard 2 B_4
Ordering shard
Shard 1
Shard 2
B_1
B_2
B_3
B_4


Future directions
COMPUTER
STORAGE
Zone1
Zone 2
Zone 3
Database1...n
Database1...n
Database1...n