Blockchain Technology: Advanced

March 22, 2023

Min Suk Kang

Assistant Professor

School of Computing/Graduate School of Information Security

NetS&P
Research Lab @ KAIST

KAIST

Two parts

- Part I: Blockchain Technology: Advanced (L1/L2, ZKP, Sharding, etc)
  - by Min Suk Kang (SoC, KAIST)

- Part II: How complicated it is to build a blockchain platform
  - by Sangmin Seo (Director, Klaytn Foundation)

Recap: Blockchain 101

Blockchain 101 lecture was very hard to follow…
as I have zero background…
Can I survive?

Don’t worry!
You can develop Web3 apps without becoming a blockchain guru.
You just need to understand some characteristics of underlying blockchain systems.

What is a blockchain?

Abstract answer: a blockchain provides
coordination between many parties,
when there is no single trusted party

if trusted party exists ⇒ no need for a blockchain

[financial systems: often no trusted party]

Blockchains: what is the new idea?

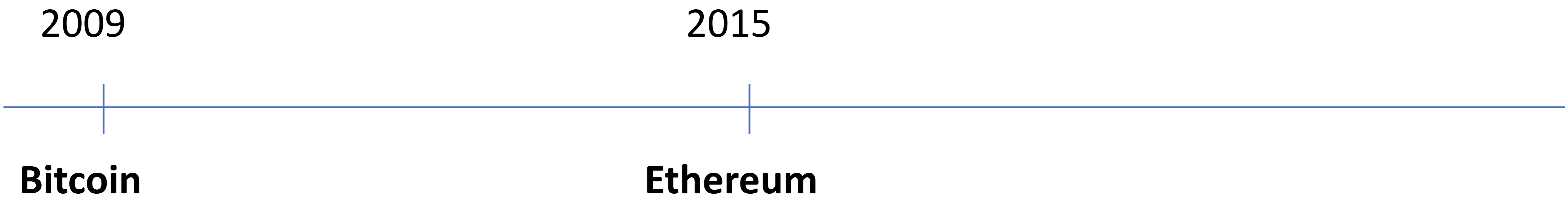
2009

Bitcoin

Several innovations:

- A practical **public append-only data structure**, secured by replication and incentives
- A fixed supply asset (BTC). Digital payments, and more.

Blockchains: what is the new idea?



2009

2015

Bitcoin

Ethereum

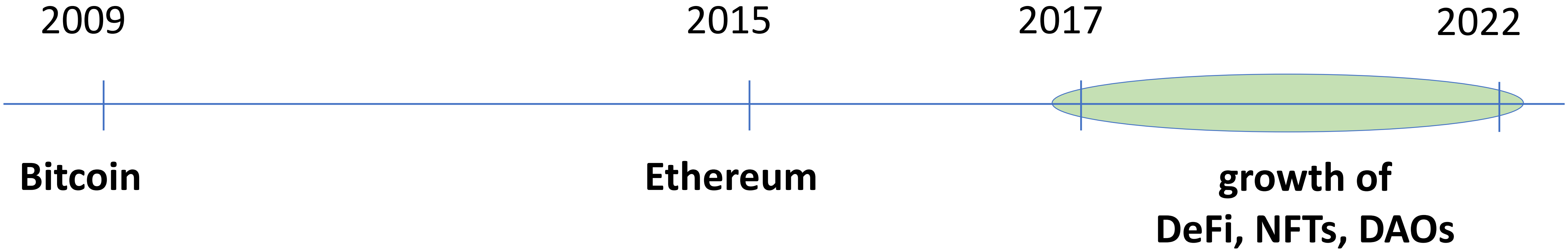
Several innovations:

- **Blockchain computer**: a fully programmable environment

  ⟹ public programs that manage digital and financial assets

- **Composability**: applications running on chain can call each other

Blockchains: what is the new idea?



2009

Bitcoin

2015

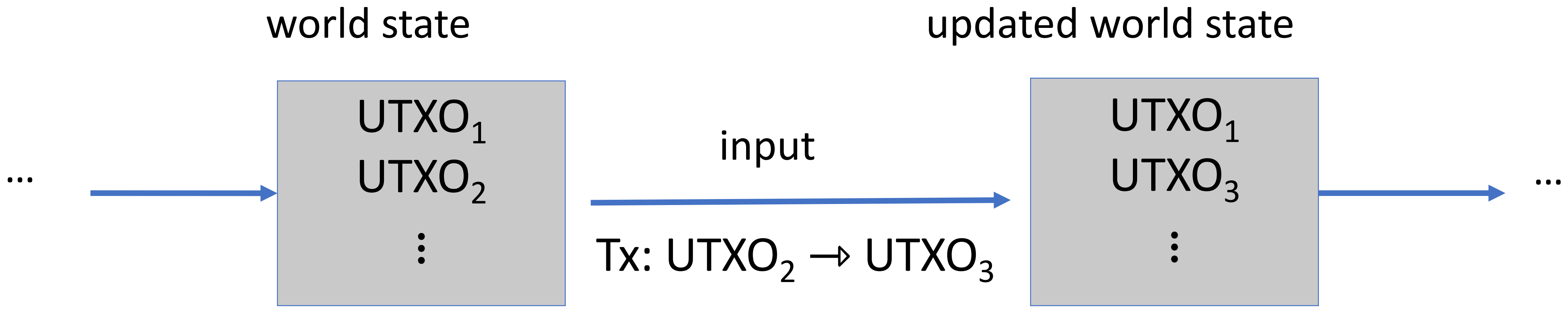
Ethereum

2017

2022

growth of
DeFi, NFTs, DAOs

Bitcoin as a state transition system



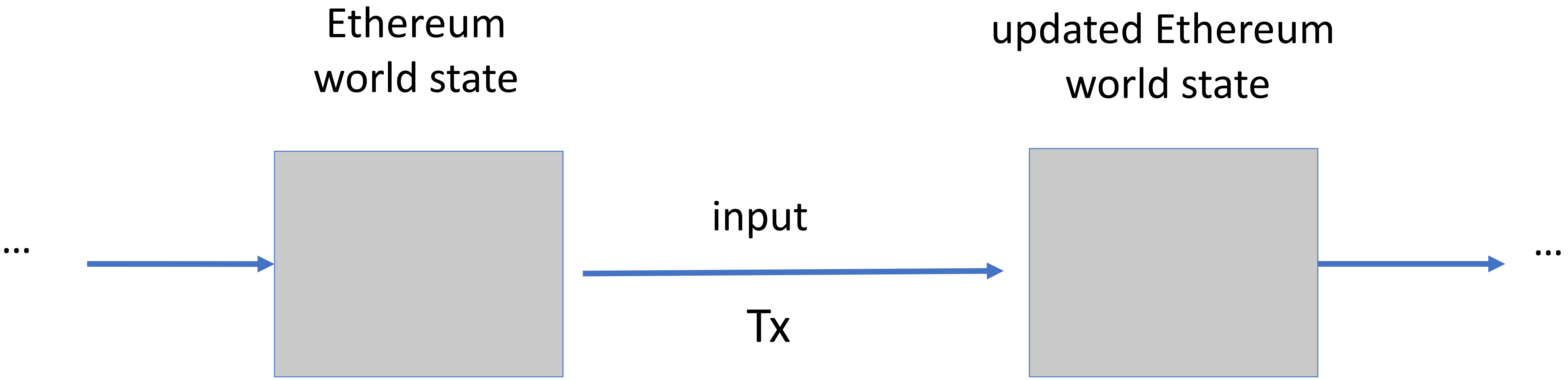
Bitcoin rules: $\boxed{F_{bitcoin} : S \times I \to S}$

S: set of all possible world states, $s_0 \in S$ genesis state
I: set of all possible inputs

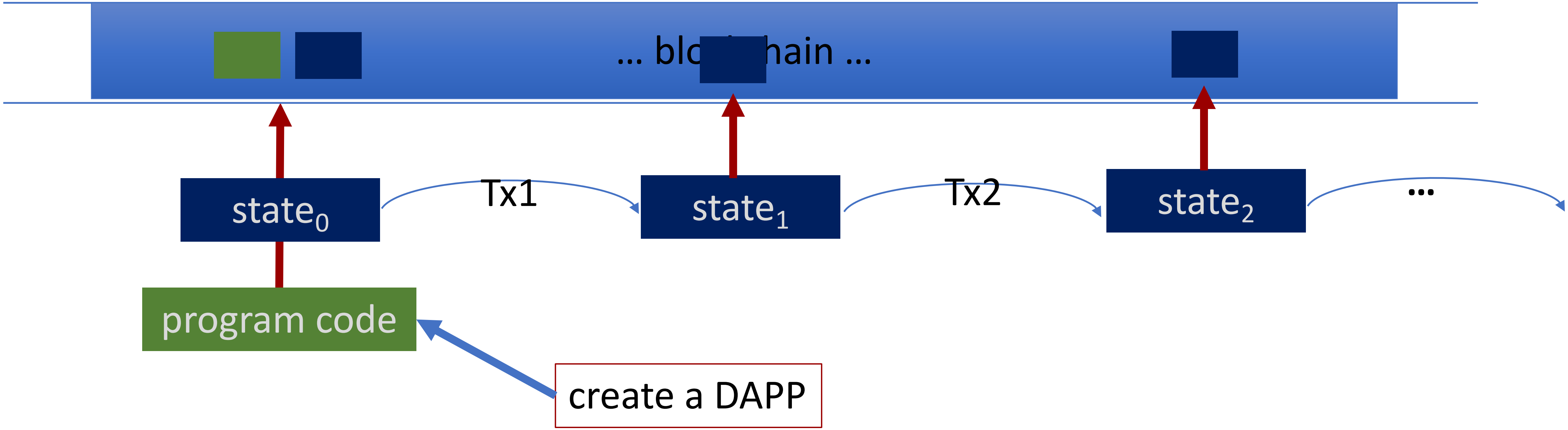
Ethereum as a state transition system

Much richer state transition functions

⇒ one transition executes an entire program



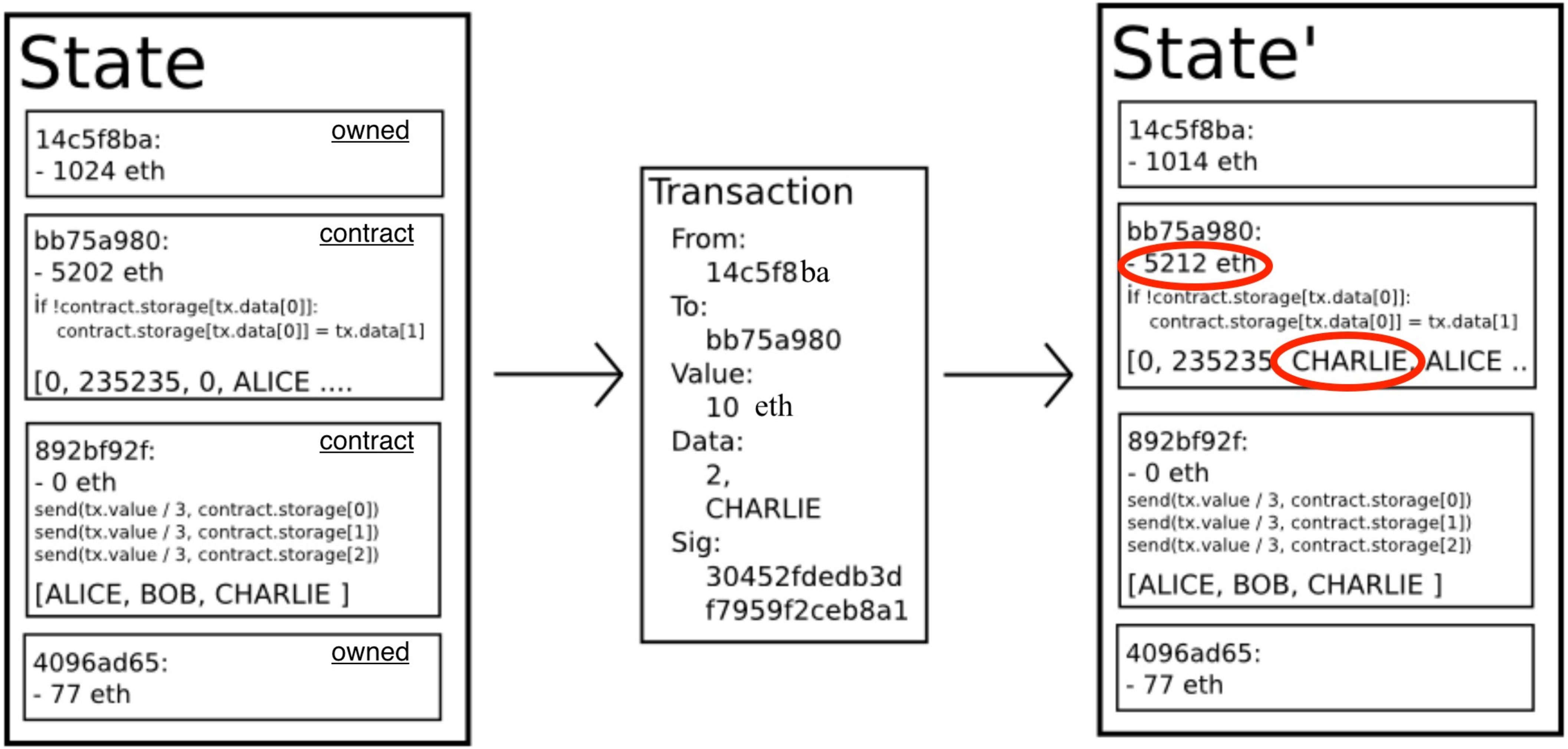
Running a program on a blockchain (DAPP)



compute layer (execution chain): The EVM

consensus layer (beacon chain)

Example Tx



State

14c5f8ba: owned
- 1024 eth

bb75a980: contract
- 5202 eth

```
if !contract.storage[tx.data[0]]:
    contract.storage[tx.data[0]] = tx.data[1]
```

[0, 235235, 0, ALICE ....

892bf92f: contract
- 0 eth

```
send(tx.value / 3, contract.storage[0])
send(tx.value / 3, contract.storage[1])
send(tx.value / 3, contract.storage[2])
```

[ALICE, BOB, CHARLIE]

4096ad65: owned
- 77 eth

Transaction

From:
14c5f8ba
To:
bb75a980
Value:
10 eth
Data:
2,
CHARLIE
Sig:
30452fdedb3d
f7959f2ceb8a1

State'

14c5f8ba:
- 1014 eth

bb75a980:
- 5212 eth

```
if !contract.storage[tx.data[0]]:
    contract.storage[tx.data[0]] = tx.data[1]
```

[0, 235235, CHARLIE, ALICE ..

892bf92f:
- 0 eth

```
send(tx.value / 3, contract.storage[0])
send(tx.value / 3, contract.storage[1])
send(tx.value / 3, contract.storage[2])
```

[ALICE, BOB, CHARLIE]

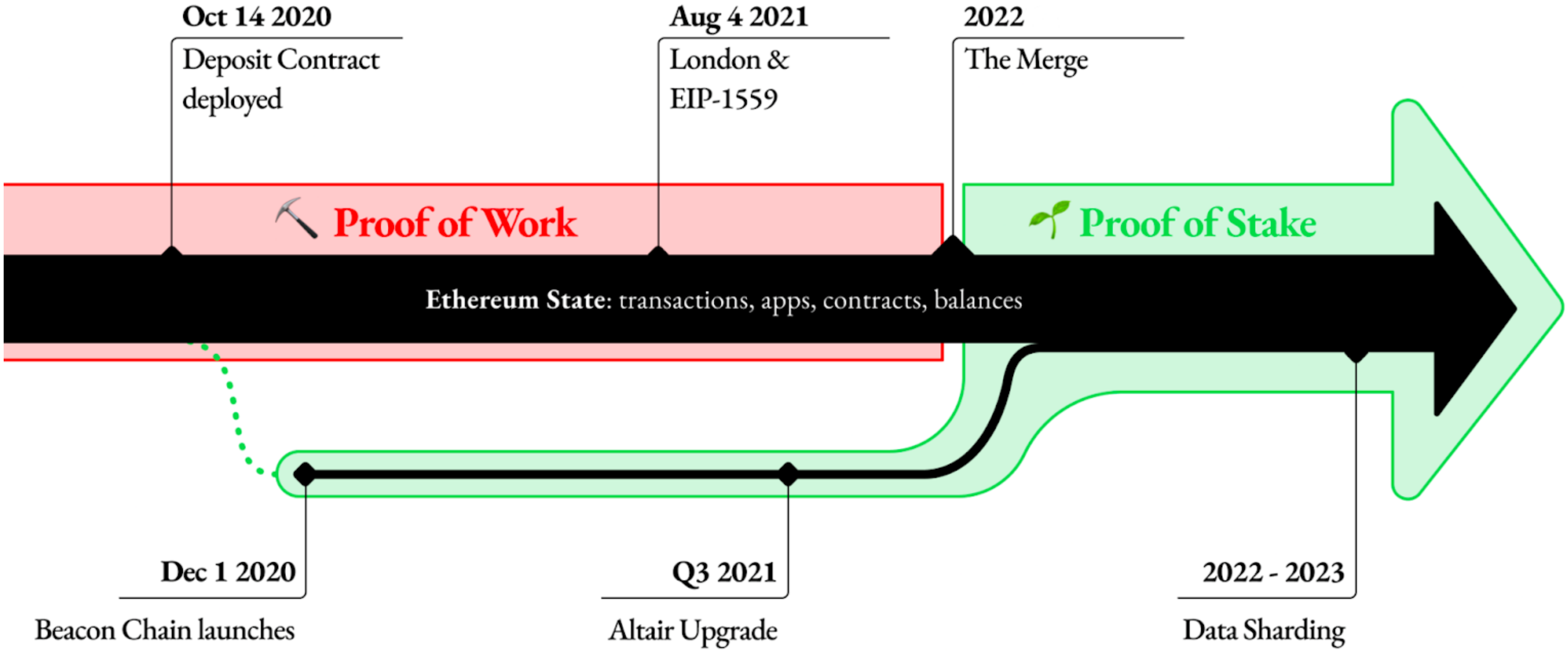
4096ad65:
- 77 eth

world state (four accounts)

updated world state

Ethereum's Upgrade Path

The Merge: when the existing PoW consensus is replaced by the Beacon Chain's PoS.
Graphic: @trent_vanepps, not "official," subject to change



Oct 14 2020
Deposit Contract deployed

Aug 4 2021
London & EIP-1559

2022
The Merge

⛏ Proof of Work

🌱 Proof of Stake

Ethereum State: transactions, apps, contracts, balances

Dec 1 2020
Beacon Chain launches

Q3 2021
Altair Upgrade

2022 - 2023
Data Sharding

Many desired properties found in blockchains

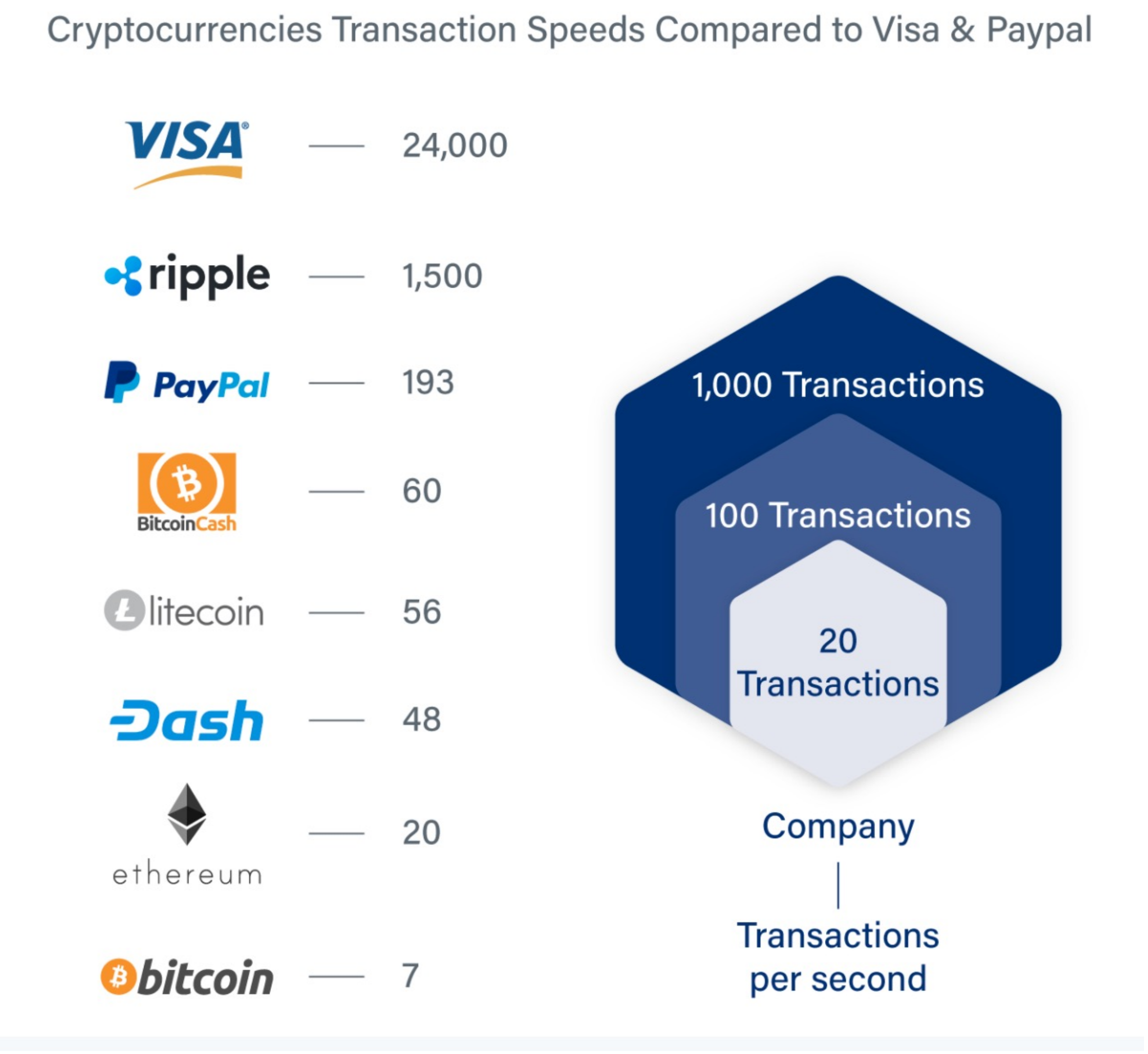
- ***Safety***: all honest participants have the same data
- ***Persistence***: once added, data can never be removed
- ***Liveness***: honest participants can add new transactions
  - dynamic availability
  - Censorship resistance

Not there yet… though

What about

- ***Throughput***: Lots of transactions per unit time, and
- ***Latency***: Short timeframe to confirm a transaction
- ***Cost***: Making transactions is too expensive

Can't we simply increase #txs per block?
(i.e., produce larger blocks?)



What is Sharding, and why it’s needed?

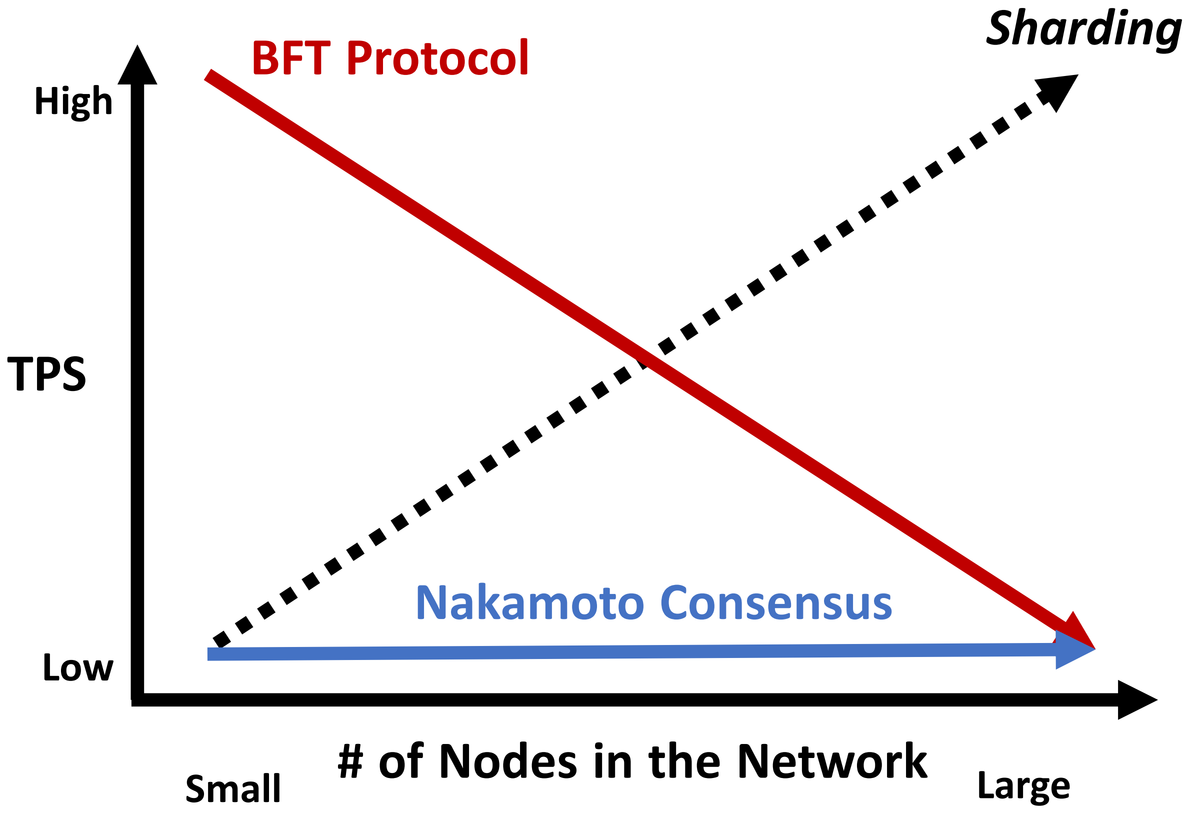
Sharding

In General:
“Method of splitting and storing a single dataset in multiple databases”

In Blockchain:
“Distributing the set of transactions to partitioned committees,
and process block in a parallel way”

A Secure Sharding Protocol For Open Blockchains

Goal: Scale *transaction rates* almost linearly with *mining power*



→ **BFT Protocol**

X **Open Environments**
- rely on Pre-established identities, PKIs

X **Scale**
- quadratic number of messages

O **Fast**
- only for small-sized networks

→ **Nakamoto Consensus**

X **Scale**
- Constant TX rate (3~7 TPS)

O **Secure**
- PoW for Miners: sybil-resistant

A Secure Sharding Protocol For Open Blockchains

Goal: Scale *transaction rates* almost linearly with *mining power*

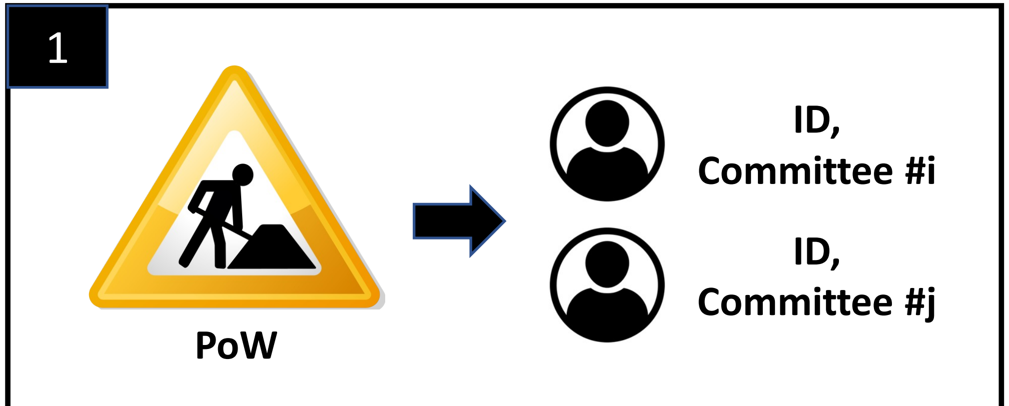
Sharding	
PoW ✚	BFT
- Generate validator's Identities - Works on Permissionless Blockchains - Assign & Form committees - Sybil-resistant	- Reaching a parallel consensus by each committee - Reaching a consensus for proposing final block

Our discussion is based on the following paper:

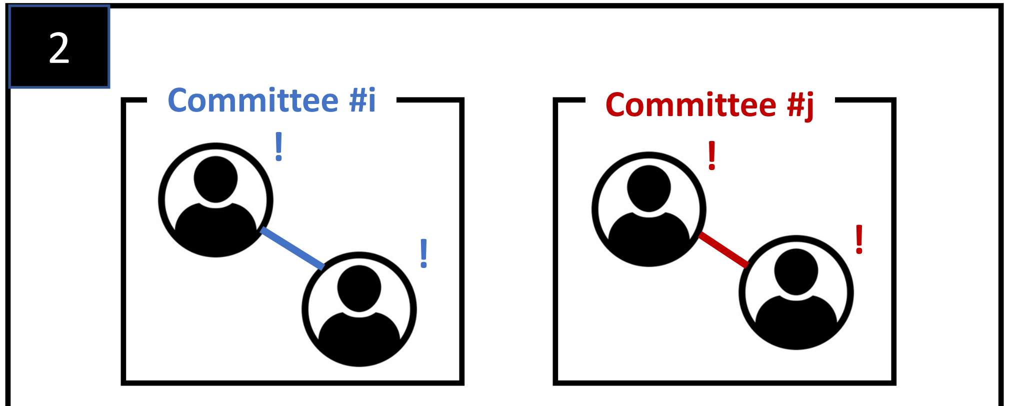
Luu, Loi, et al. "A secure sharding protocol for open blockchains." *Proceedings of ACM CCS*. 2016.

Elastico Protocol in Each Epoch:

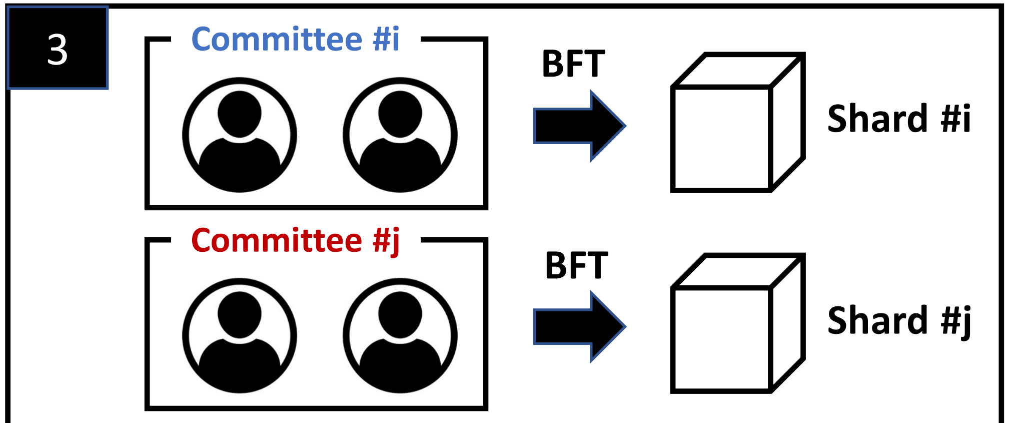
Luu, Loi, et al. "A secure sharding protocol for open blockchains." *Proceedings of ACM CCS*. 2016.



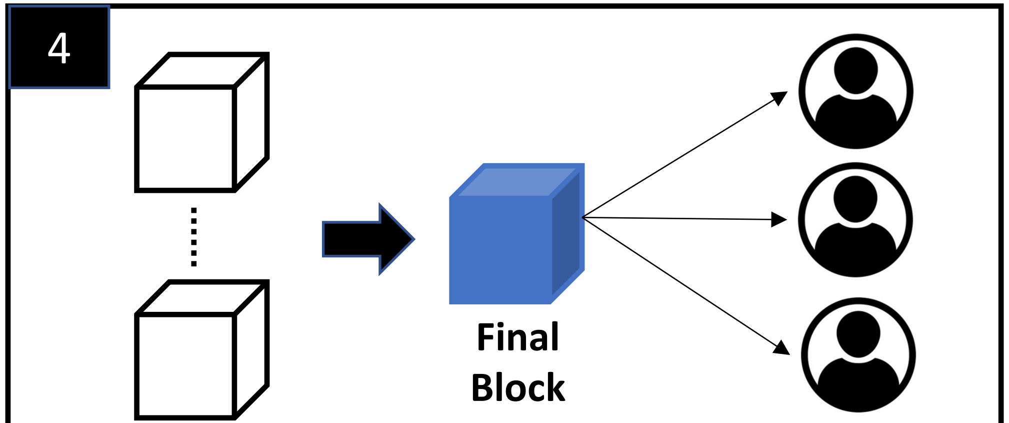
Identity Establishment & Committee Formation



Overlay Setup for Committees



Intra-committee Consensus



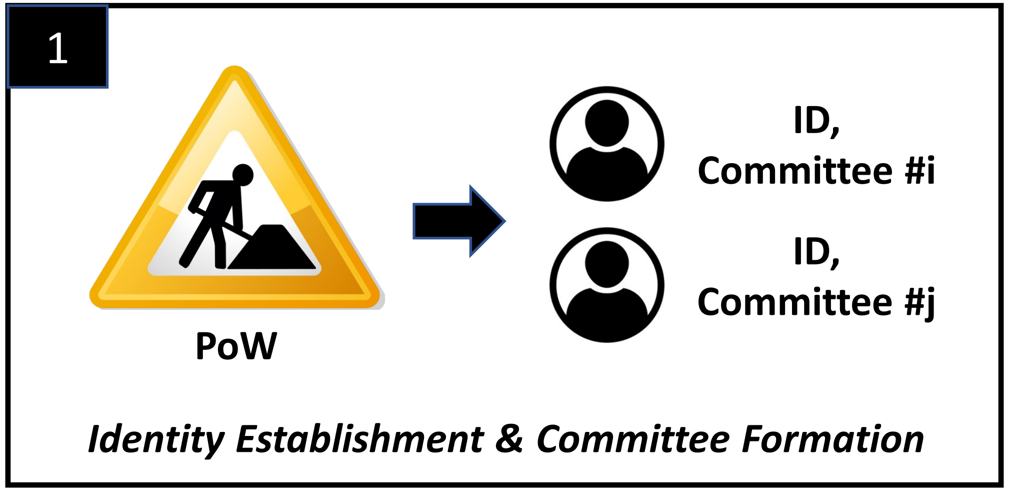
Final Consensus Broadcast

Elastico (1) : Identity Setup and Committee Formation

N Processors

K Committees

C Members per Committees



1

PoW

ID, Committee #i

ID, Committee #j

Identity Establishment & Committee Formation

$$\text{ID} = \text{H(EpochRandomness||IP||Public Key||Nonce)} \leq 2^{\gamma - D}$$

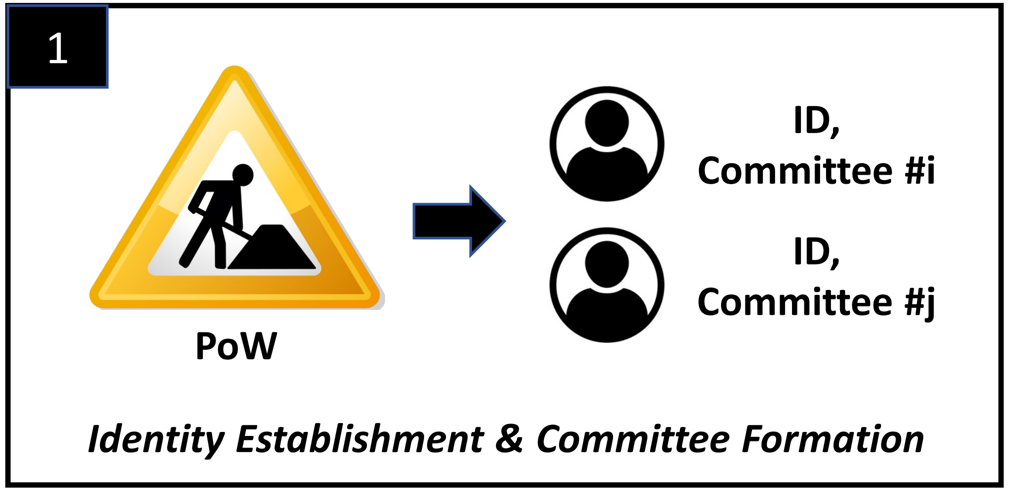
↓

00000010….101

$*\ \gamma$: bit length of Hash Output
D: Difficulty

Elastico (1) : Identity Setup and Committee Formation

N Processors
K Committees
C Members per Committees



1

PoW

ID,
Committee #i

ID,
Committee #j

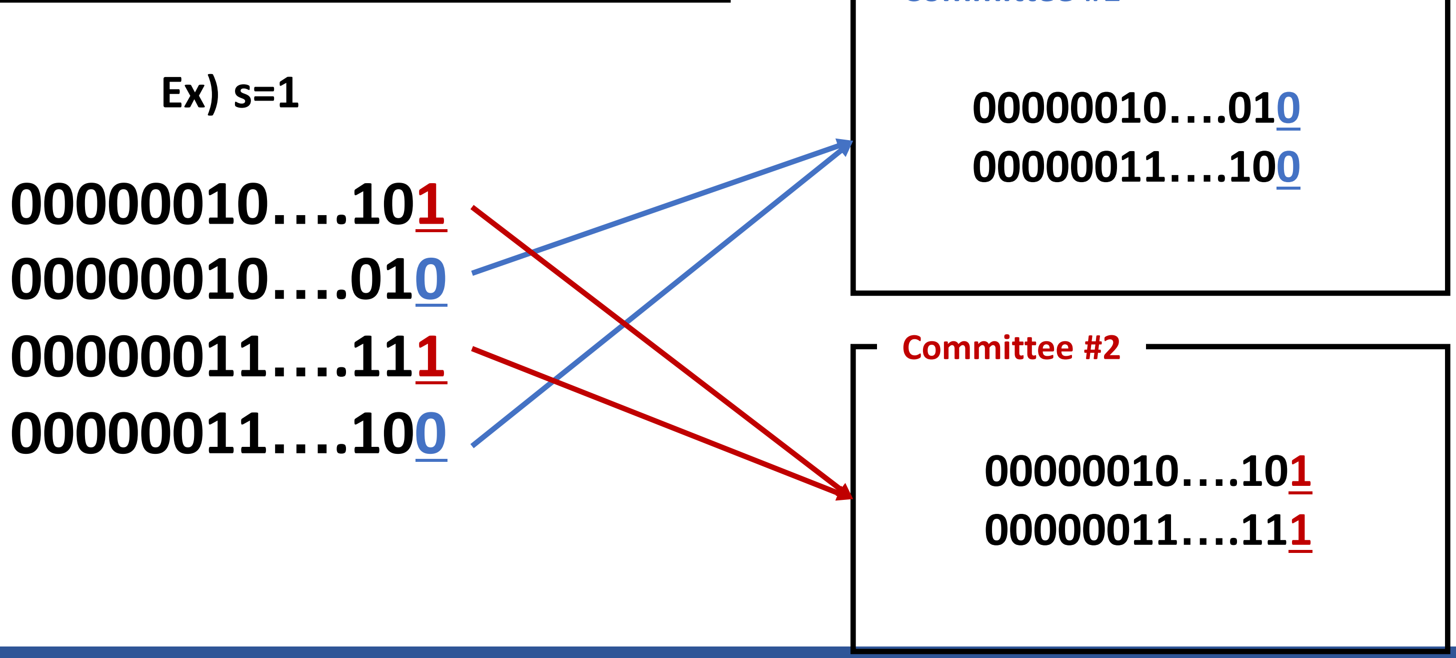
Identity Establishment & Committee Formation

Goals for assigning committees:

- ✓ Fairly distribute the nodes
- ✓ Guarantee at most 1/3 adversary nodes per committee

⟹ **Use last s bits of ID** $(2^s = K)$

Ex) s=1



00000010….101
00000010….010
00000011….111
00000011….100

Committee #1

00000010….010
00000011….100

Committee #2

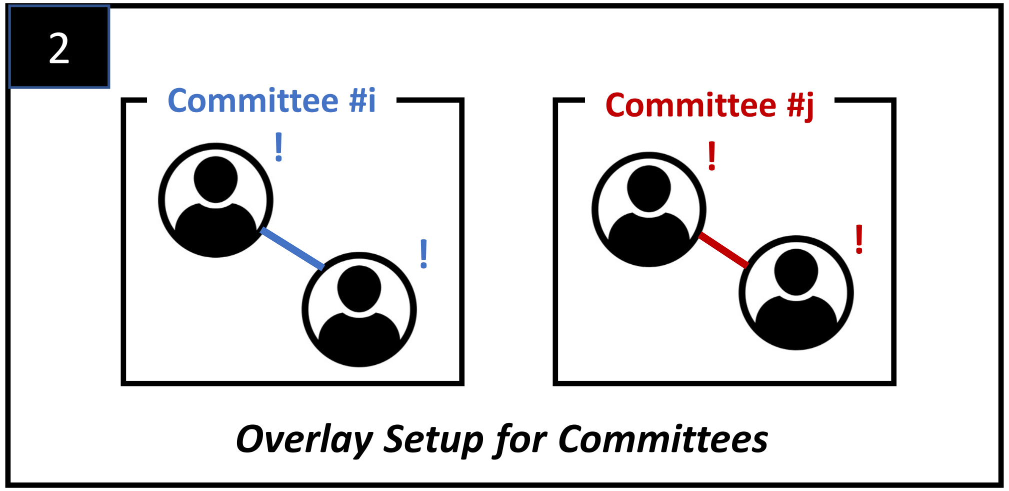
00000010….101
00000011….111

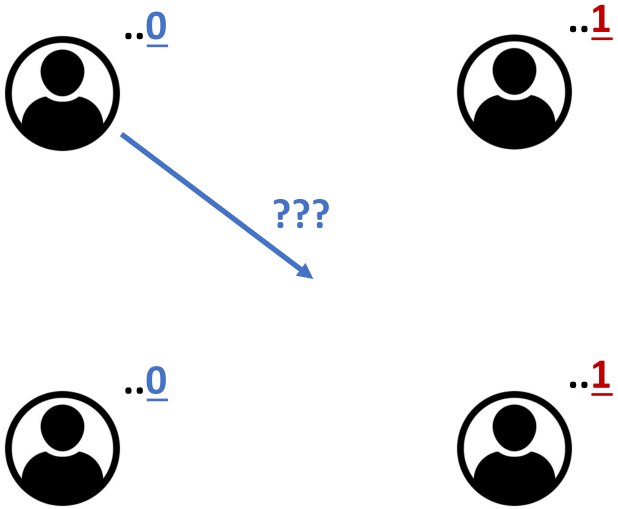
Elastico (2) : Overlay Setup for committees

N Processors

K Committees

C Members per Committees

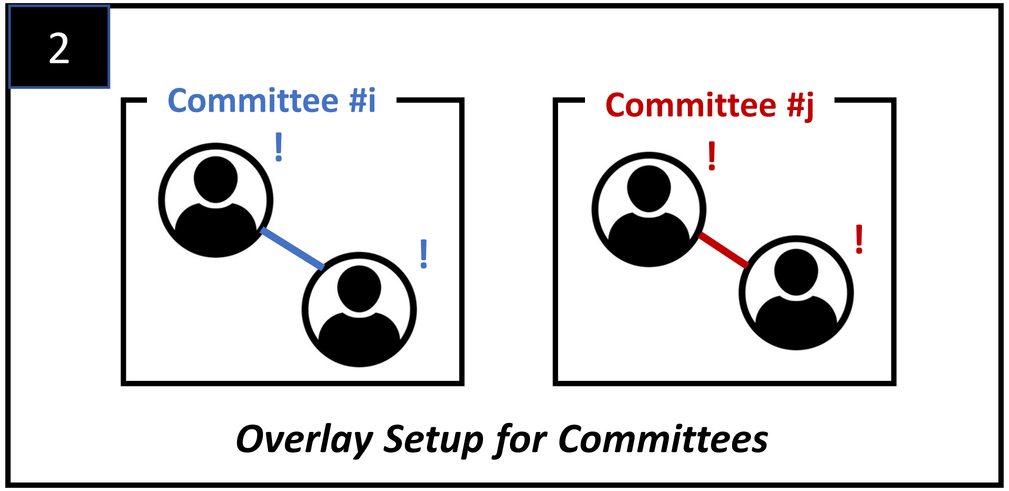




Naïve Solution?

Elastico (2) : Overlay Setup for committees

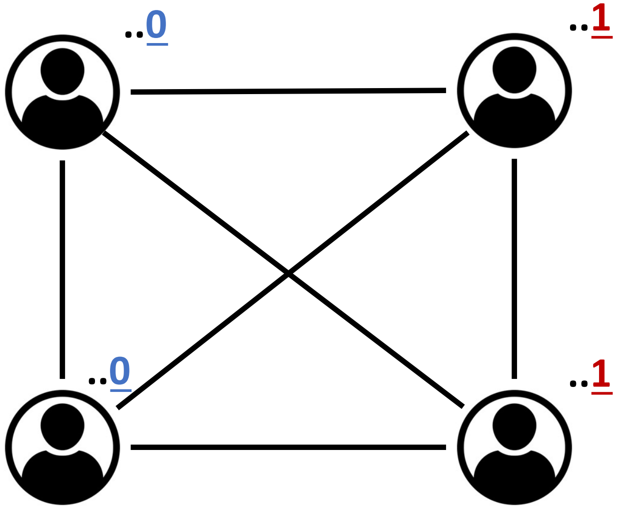
N Processors
K Committees
C Members per Committees



Naïve Solution:

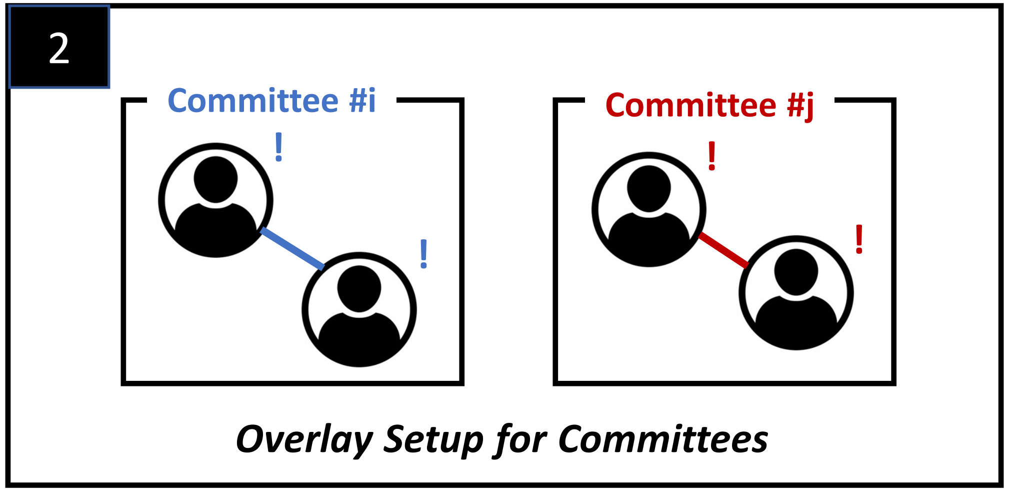
- ✓ Broadcast its identity to everyone

⟹ **quadratic messages.. O**(N^2)



Elastico (2) : Overlay Setup for committees

N Processors
K Committees
C Members per Committees



Overlay Setup for Committees

Better Solution:

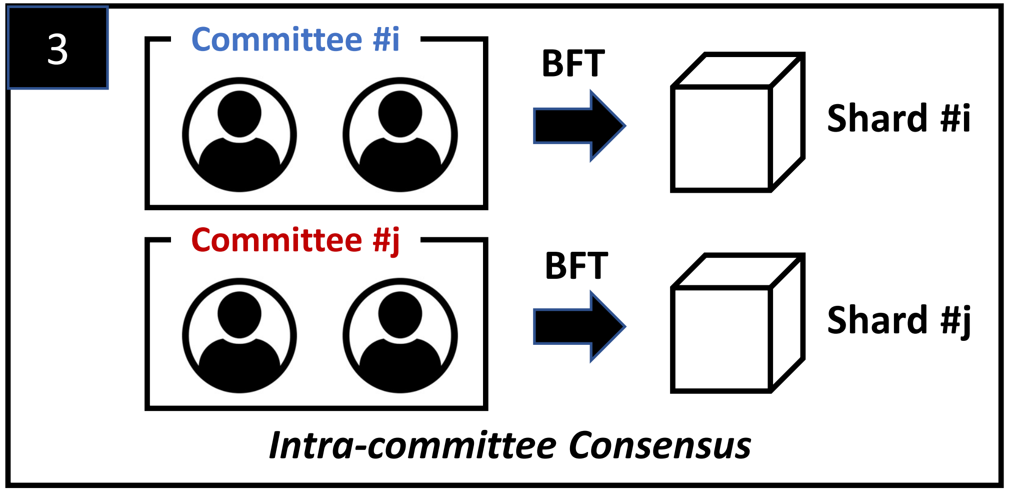
- ✓ Use Directory Committees
- ✓ First C identities become Directory Committees
- ✓ Latter nodes send IDs to Directories
- ✓ Directories send committee list once each has $\geq$ C members

$\Longrightarrow$ **O(NC)**

***Directory committees broadcast its identity to all Directory committee members.**

Elastico (3) : Intra-committee Consensus

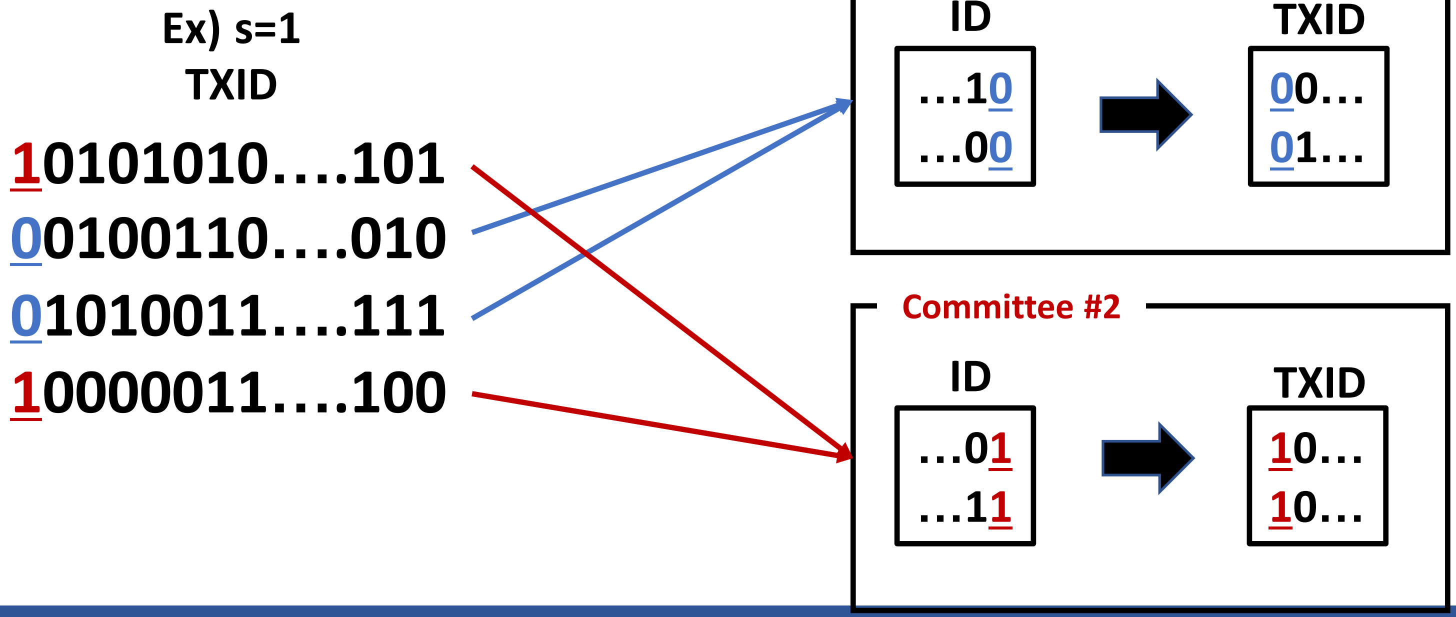
N Processors
K Committees
C Members per Committees



All committees propose disjoint shards:

✓ Each committee works on a *separate* transactions based on their ID

⟹ **Use first s bits of TXID**

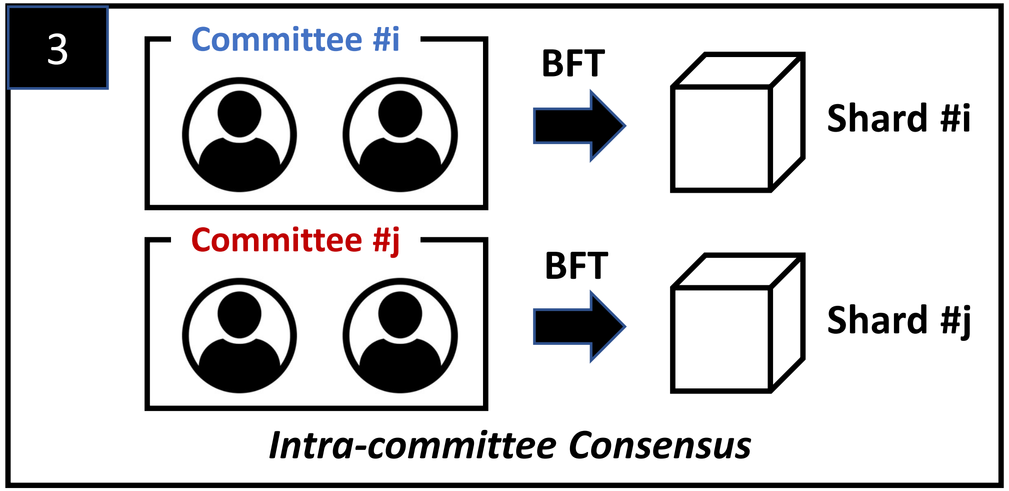


Ex) s=1
TXID

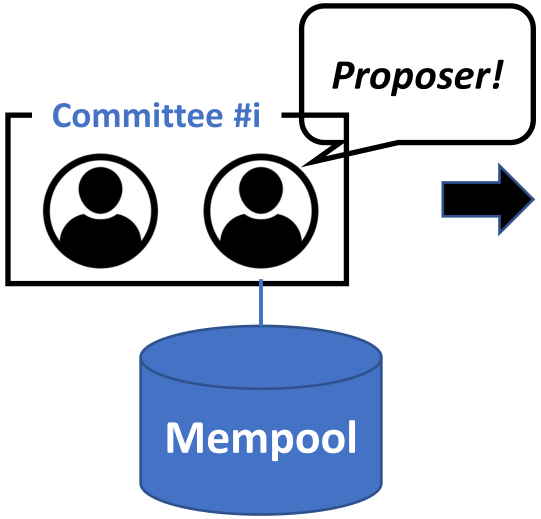
10101010….101
00100110….010
01010011….111
10000011….100

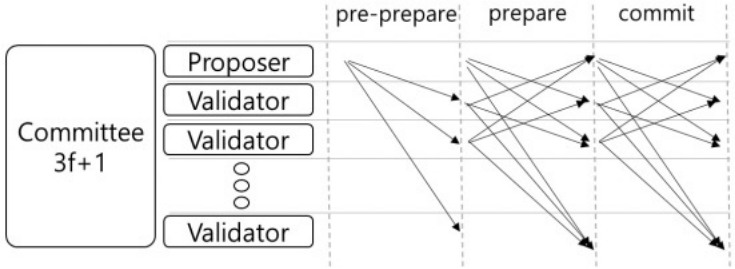
Elastico (3) : Intra-committee Consensus

N Processors
K Committees
C Members per Committees

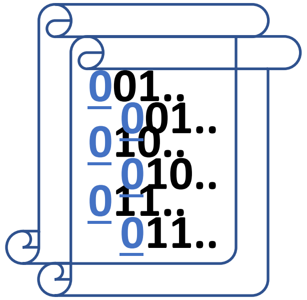


Intra-committee Consensus





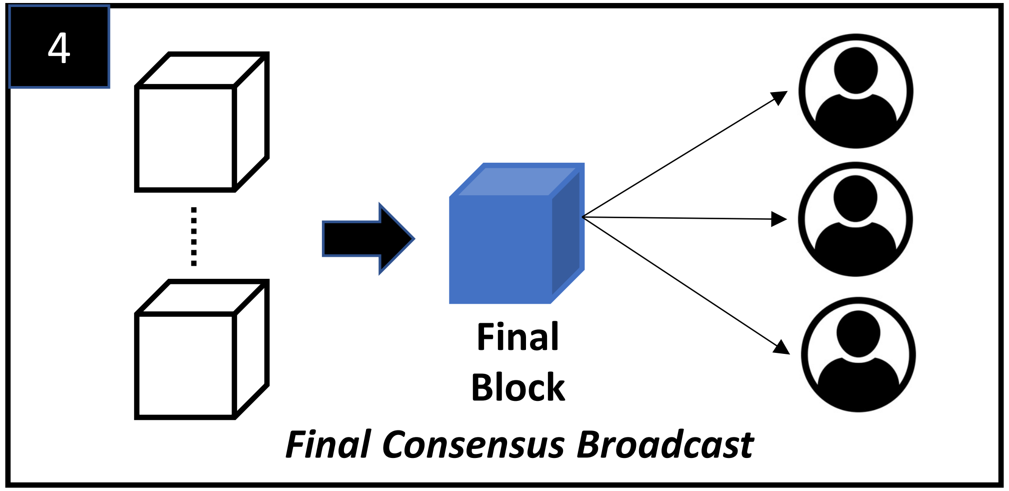
Run BFT Protocols



Agree on set of TXs (shard)

Elastico (4) : Final Consensus Broadcast

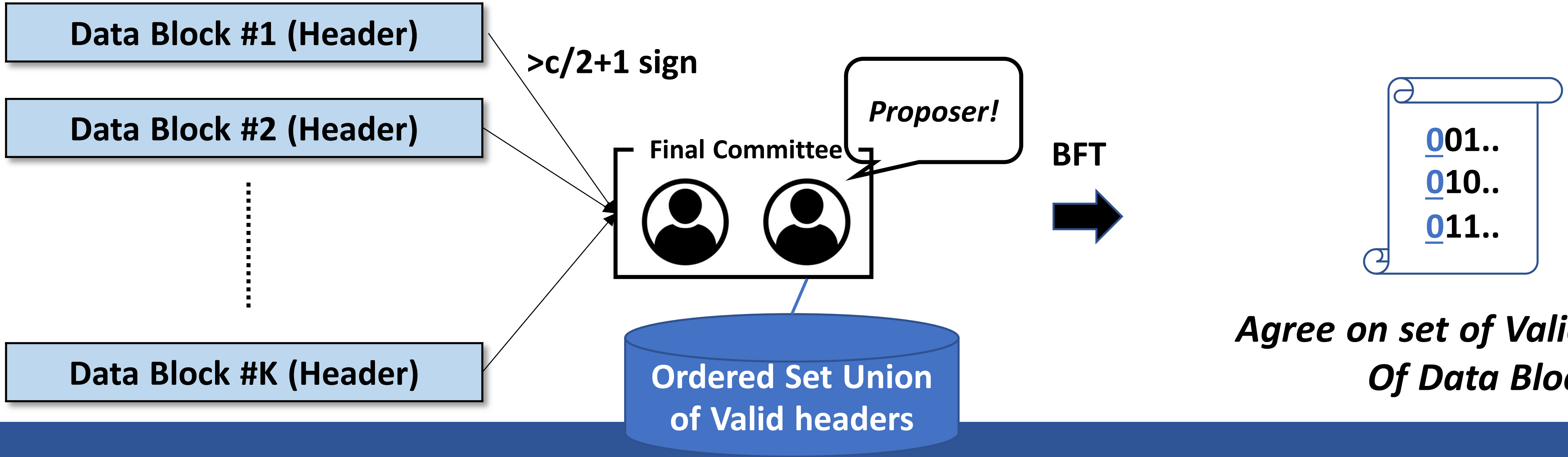
N Processors
K Committees
C Members per Committees



4

Final Block

Final Consensus Broadcast



Data Block #1 (Header)

Data Block #2 (Header)

Data Block #K (Header)

>c/2+1 sign

Final Committee

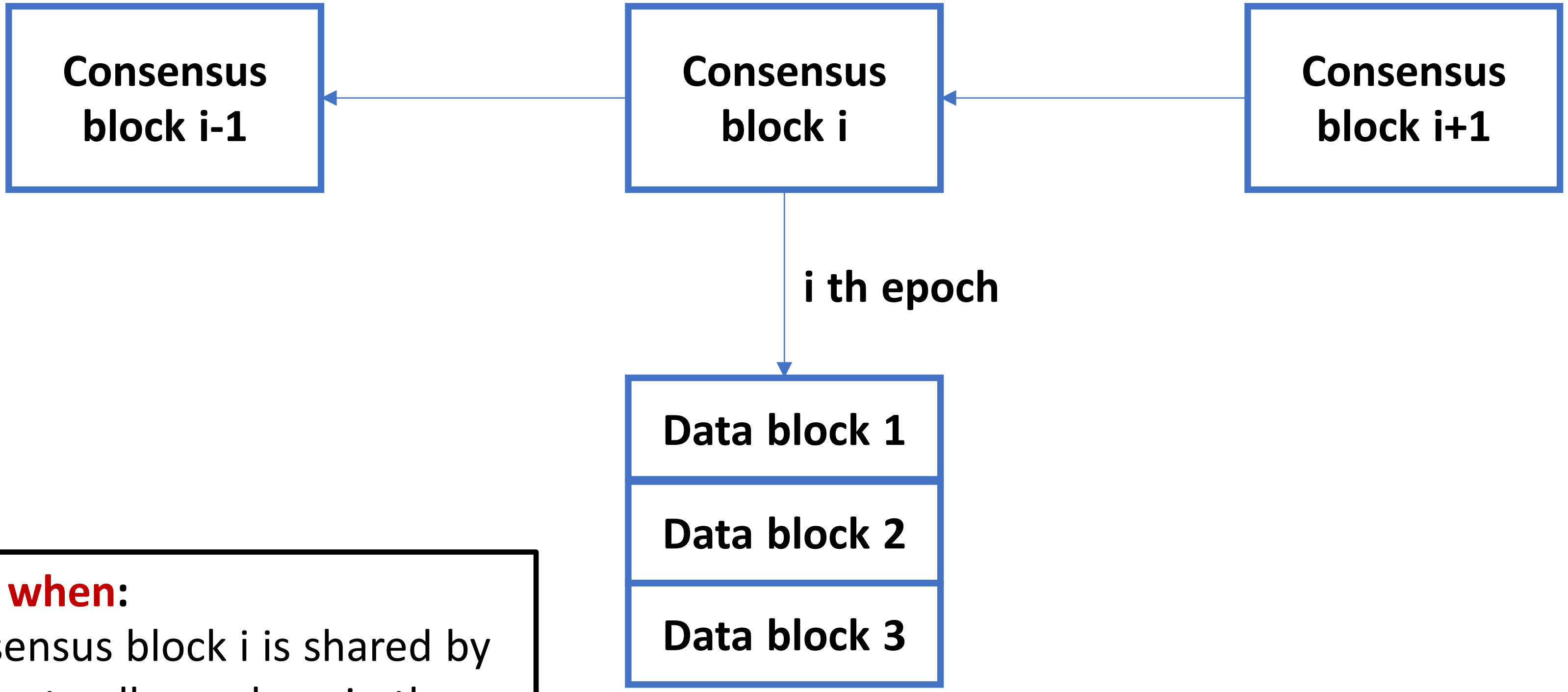
Proposer!

BFT

Ordered Set Union of Valid headers

Agree on set of Valid Headers Of Data Blocks

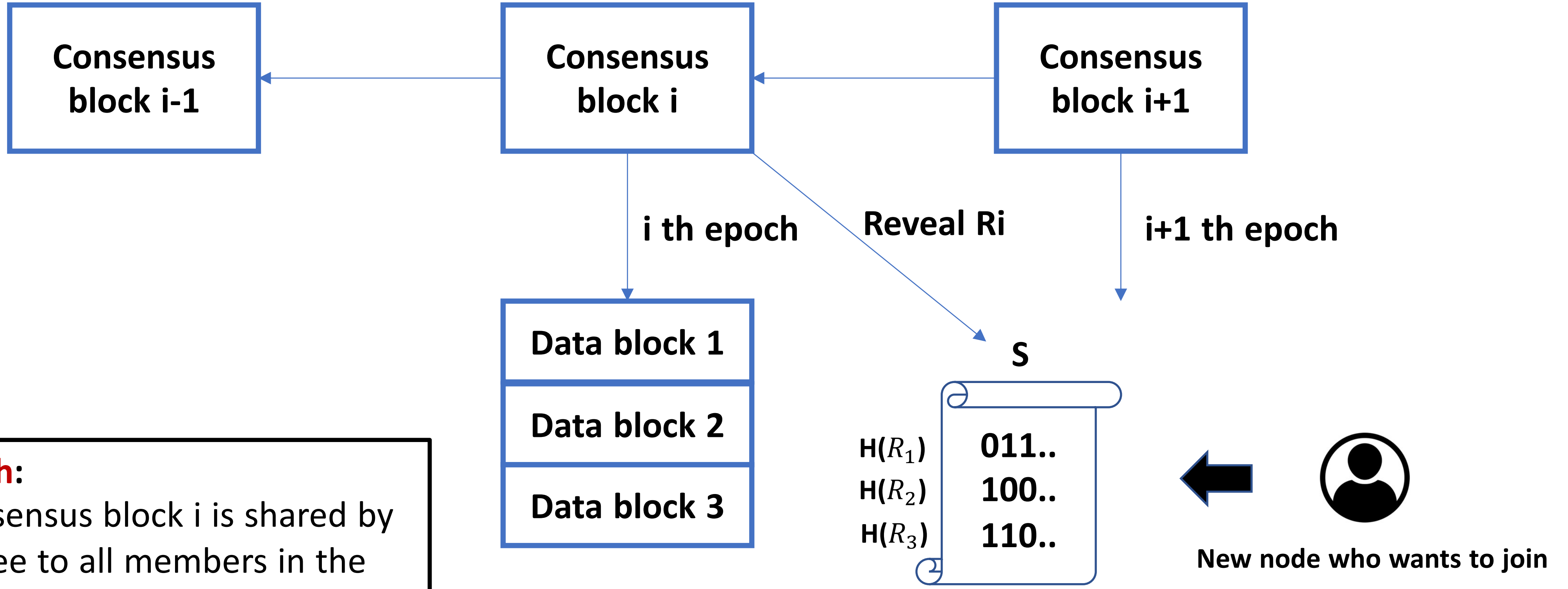
Elastico (4) : Final Consensus Broadcast



Each Epoch ends when:

- ✓ Once the consensus block i is shared by final committee to all members in the network, it is added to the blockchain.
- ✓ Each step process repeats in the next epoch i+1.
- ✓ Broadcast S along with consensus block.

Elastico (4) : Final Consensus Broadcast



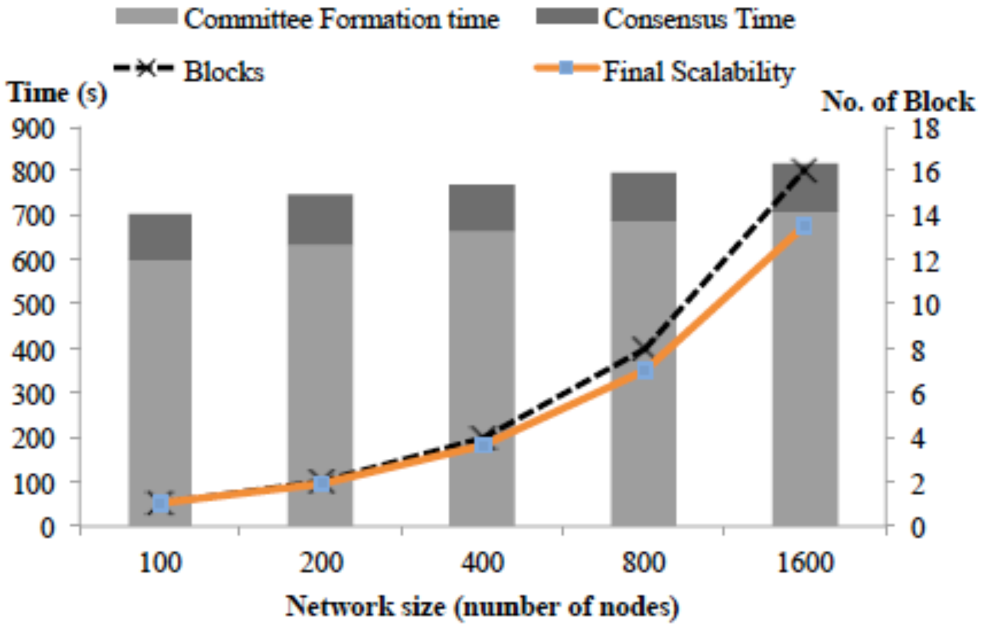
In the next epoch:

- ✓ Once the consensus block i is shared by final committee to all members in the network, it is added to the blockchain.
- ✓ Each step process repeats in the next epoch i+1.
- ✓ Broadcast S along with consensus block.

EpochRandomness = H(R_a) ⊕ H(R_b) ⊕ H(R_c) ⊕ … ⊕ H(R_j)
XOR c/2 + 1 H(R_i)s

Results

100 Members per Committees



Limitations of Sharding

- Cross-shard consensus
- Reduced composability
- New security risks

Scaling blockchains

- ***Sharding***: parallelize blockchain network
- ***Payment channel***: try not to touch blockchain (except when necessary)
- ***Rollups***: post only summary of tx/contract executions to blockchain

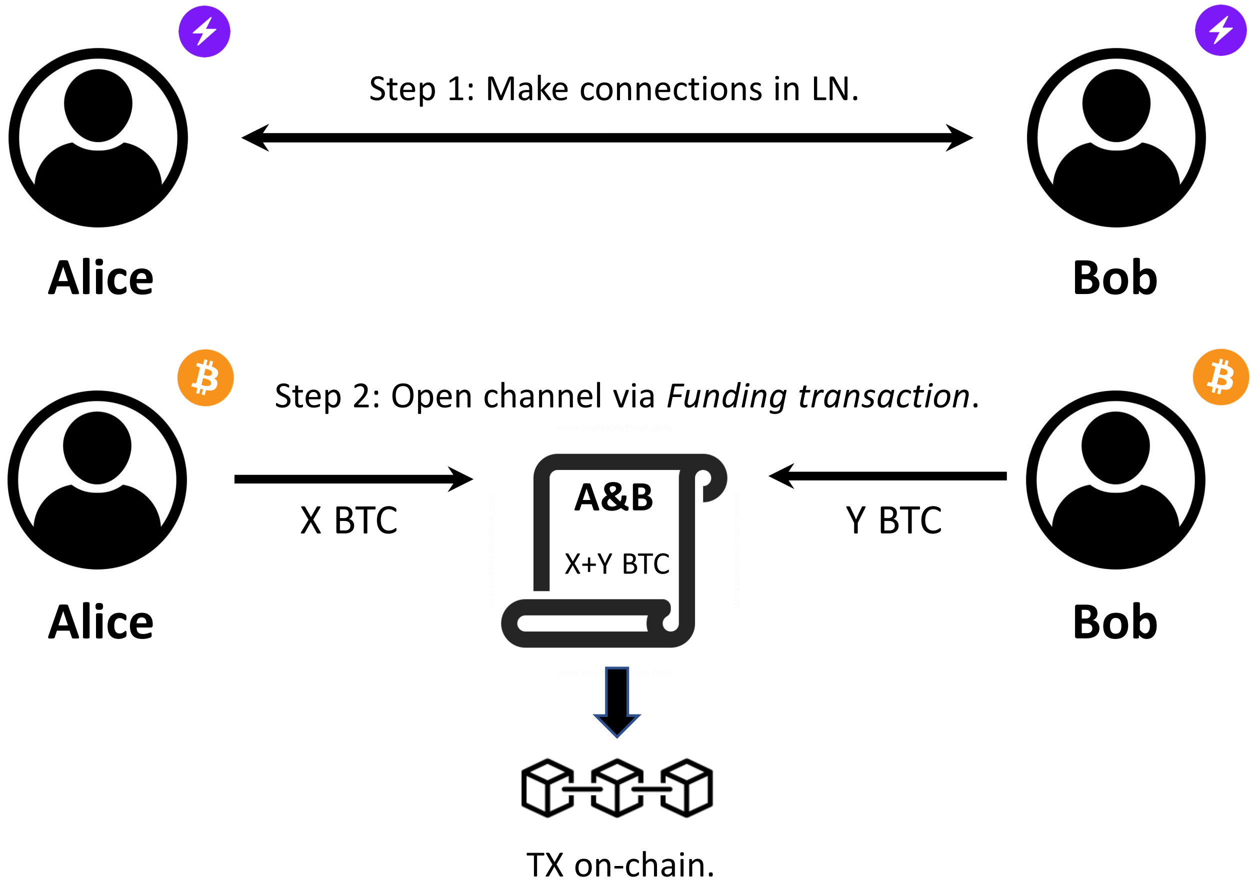
Payment Channels: Initiating

Layer 2

Lightning Network.

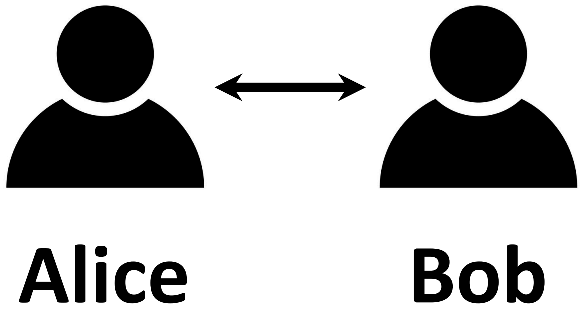
Layer 1

Main-net.



Step 1: Make connections in LN.

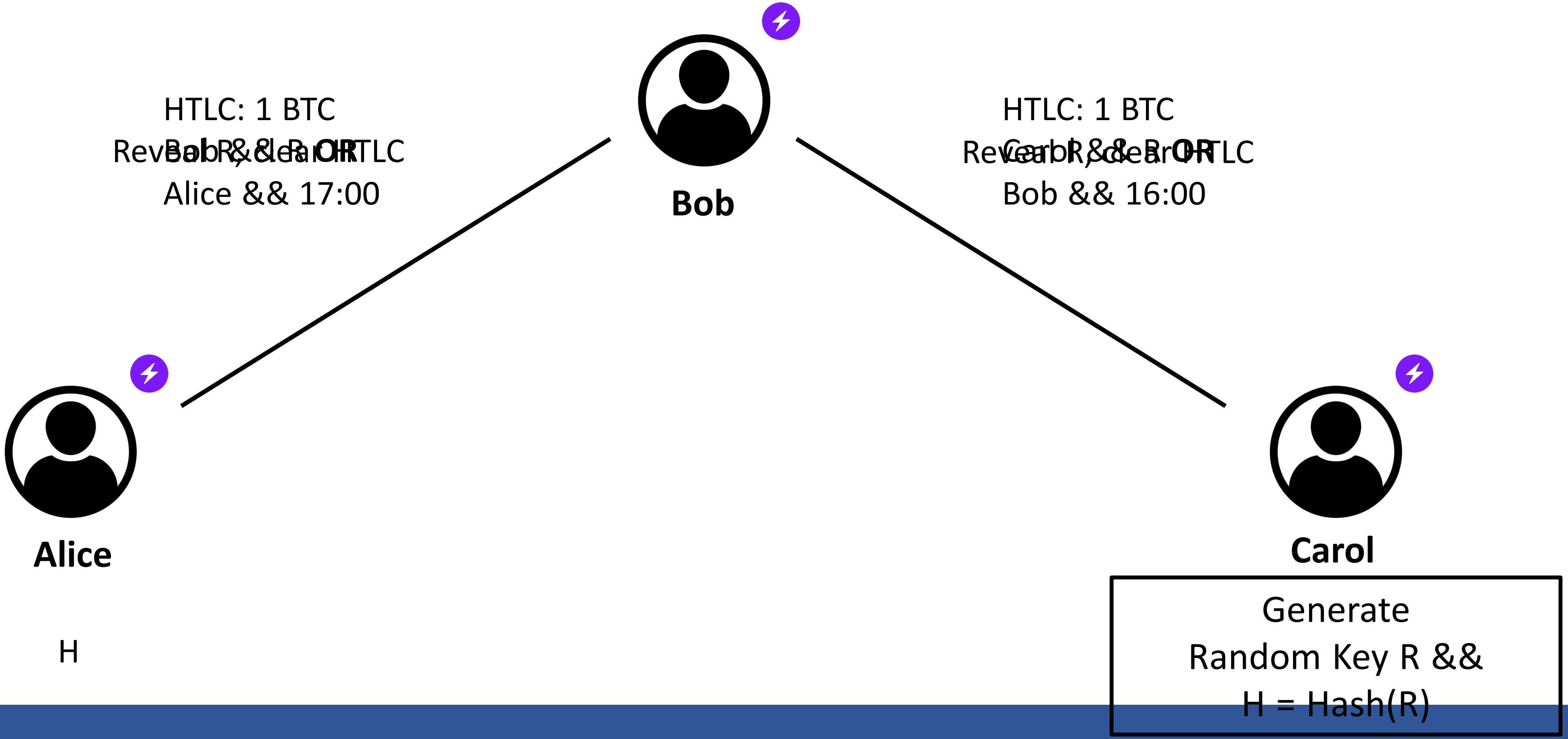
Step 2: Open channel via *Funding transaction*.



Payment Channels: Multi-hop payments (HTLC)

*Hash Time Lock Contracts

Alice wants to send Carol 1 BTC via Bob:



HTLC: 1 BTC
Bob && R OR
Reveal R, clear HTLC
Alice && 17:00

Bob

HTLC: 1 BTC
Carol && R OR
Reveal R, clear HTLC
Bob && 16:00

Alice

H

Carol

Generate
Random Key R &&
H = Hash(R)

Limitations of payment channels

- User assets should be locked up
- Mainly designed for payments but not for contracts

Scaling blockchains

- ***Sharding***: parallelize blockchain network
- ***Payment channel***: try not to touch blockchain (except when necessary)
- ***Rollups***: post only summary of tx/contract executions to blockchain

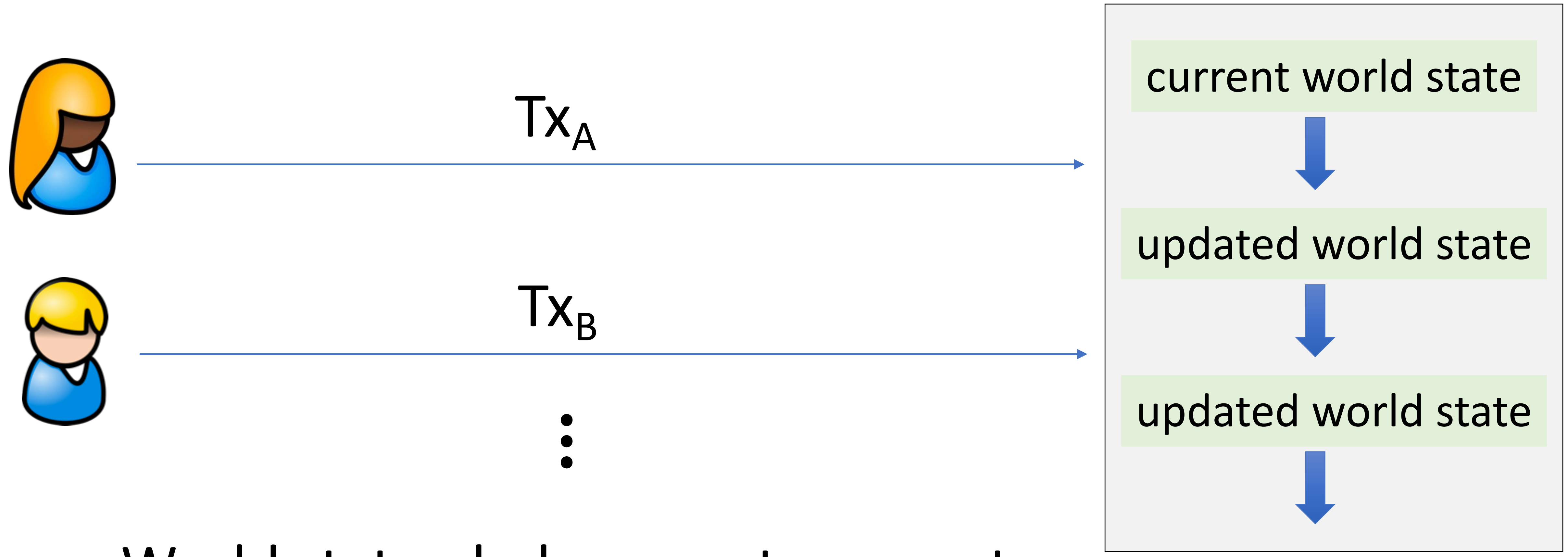
(some slides from Dan Boneh)

Kalodner, Harry, et al. "Arbitrum: Scalable, private smart contracts." *in Proceedings of USENIX Security,* 2018.

Basic layer-1 blockchain

Can handle 15 Tx/sec …

A layer-1 blockchain
(e.g., Ethereum)



Tx_A

Tx_B

⋮

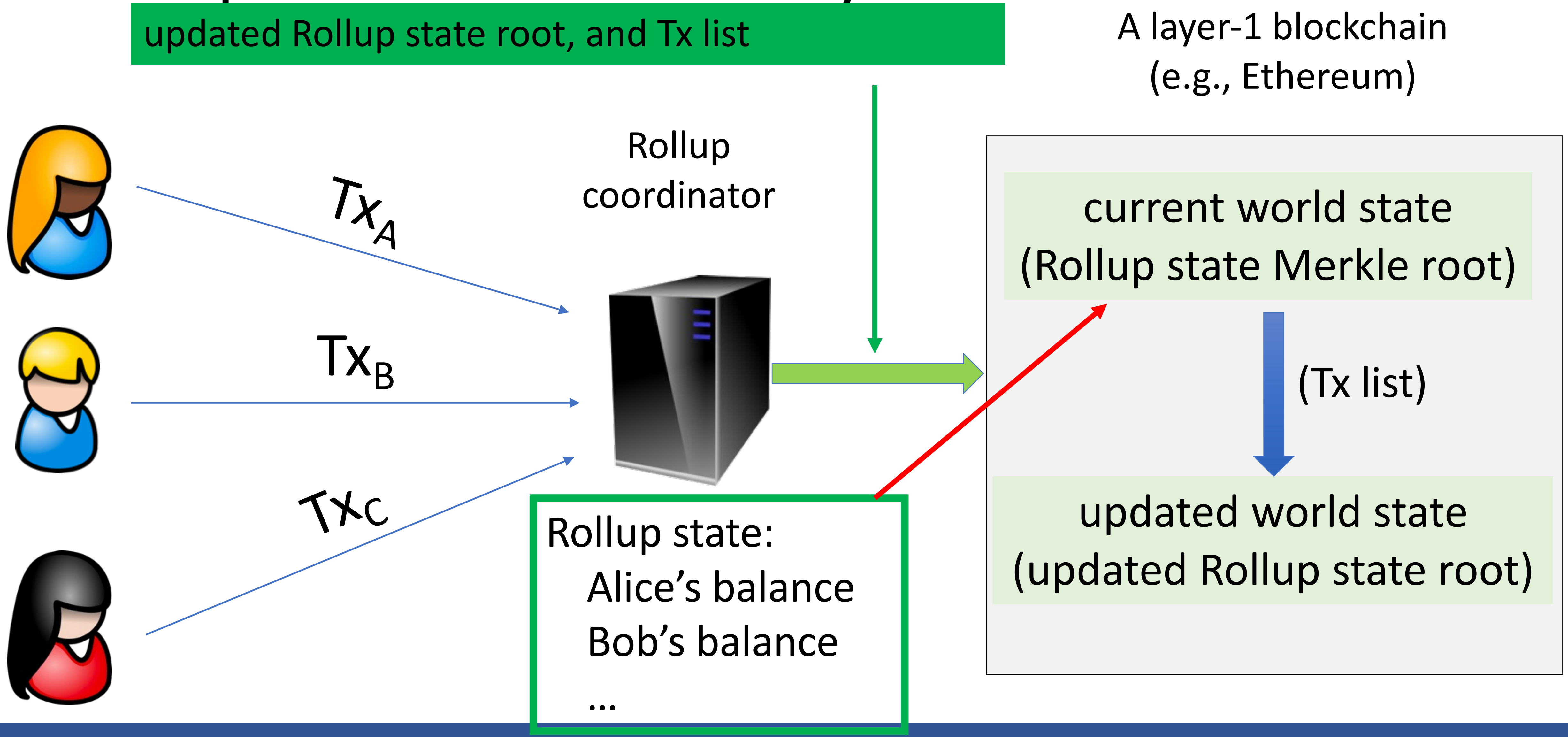
current world state

updated world state

updated world state

World state: balances, storage, etc.

Rollup idea 1: batch many Tx into one



updated Rollup state root, and Tx list

A layer-1 blockchain
(e.g., Ethereum)

Rollup
coordinator

current world state
(Rollup state Merkle root)

(Tx list)

updated world state
(updated Rollup state root)

Rollup state:
Alice’s balance
Bob’s balance
…

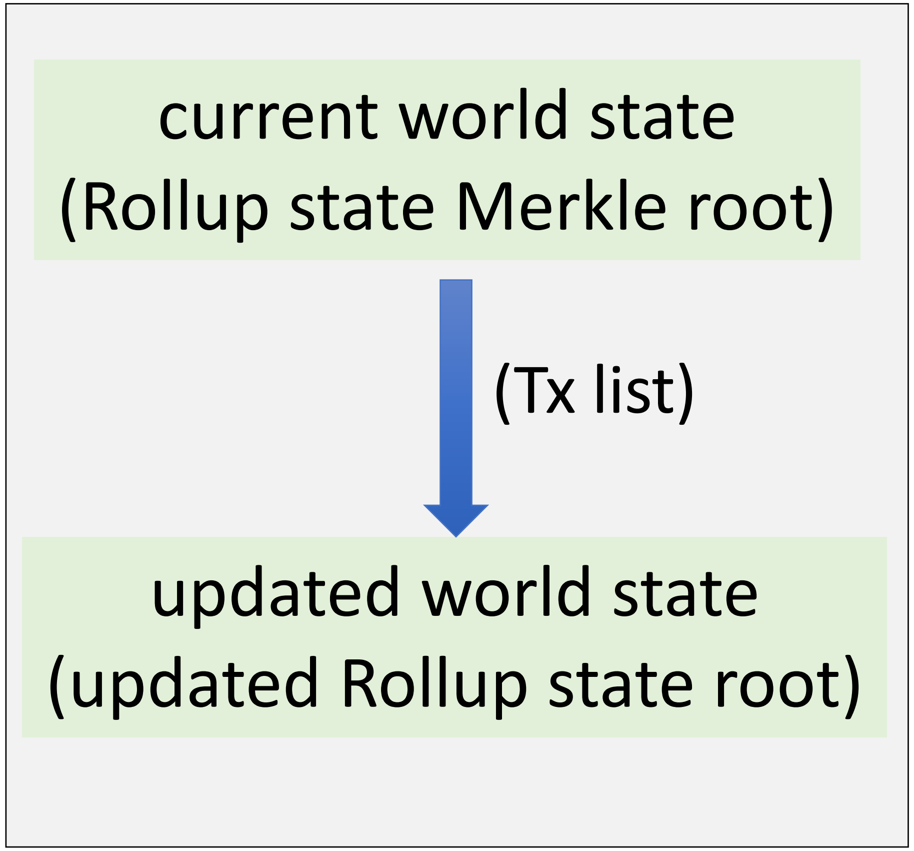
Rollup idea 1: batch many Tx into one

Key point:

- *Hundreds* of transactions on Rollup state are batched into a *single* transaction on layer-1

  ⇒ 100x speed up in Tx/sec

A layer-1 blockchain
(e.g., Ethereum)



Rollup state:
Alice’s balance
Bob’s balance
…

Two potential problems of rollup

Problem 1: what if coordinator is dishonest?

- It could steal funds from the Rollup contract
- It could issue fake Tx on behalf of users

Problem 2: what if coordinator stops providing service?

- If Rollup state is lost, how can we initialize a new coordinator?

Handling dishonest coordinators

a.k.a. optimistic rollup

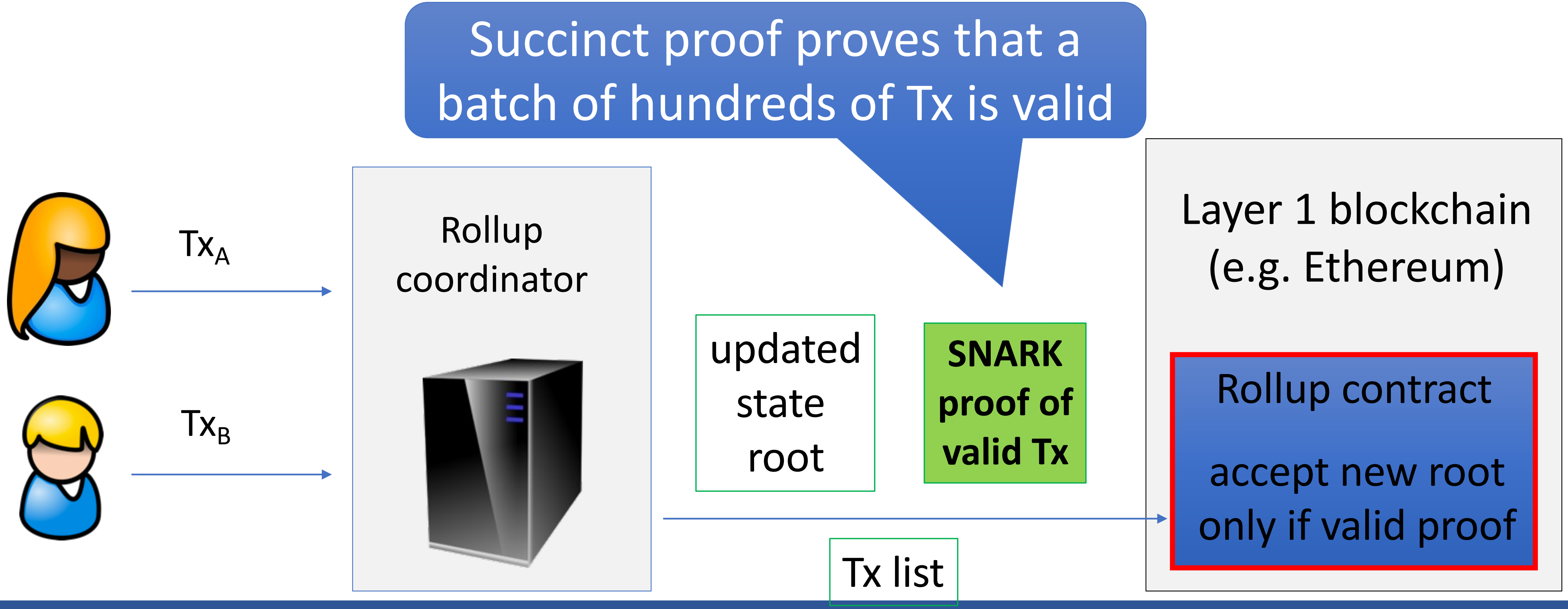
- Idea 1: Let multiple coordinators disagree and present a proof of fraud
  - If all the coordinators output the same contract execution => unanimous agreement => L1 chain processes immediately
  - If no unanimous agreement => at least one coordinator challenges
    - Through interactions between coordinators, a concise fraud proof is sent to L1 chain => L1 checks one computation step
    - Lier’s stake will be slashed
  - Dispute resolution period: typically 7 days

Handling dishonest coordinators

a.k.a. zk-rollup

- Idea 2: Let coordinators provide proof of validity
  - Coordinator processes all tx and outputs succinct proof that proves that a batch of hundreds of tx is valid
  - L1 efficiently verifies the validity proof and accepts it

Verifying Rollup state updates



Succinct proof proves that a batch of hundreds of Tx is valid

Tx_A

Tx_B

Rollup coordinator

updated state root

SNARK proof of valid Tx

Tx list

Layer 1 blockchain (e.g. Ethereum)

Rollup contract

accept new root only if valid proof

What the SNARK proof proves

SNARK proof is **short** and **fast** to verify:

⇒ Cheap to verify proof on the slow L1 chain (with EVM support)

Public statement: (old state root, new state root, Tx list)

Witness: (state of each touched account pre- and post- batch,
Merkle proofs for touched accounts, user sigs)

SNARK proof proves that:

(1) all user sigs on Tx are valid, (2) all Merkle proofs are valid,
(3) post-state is the result of applying Tx list to pre-state

The end result

Rollup contract on L1 ensures coordinator cannot cheat:

- all submitted Tx must have been properly signed by users
- all state updates are valid

⇒ Rollup contract on L1 will accept any update with a valid proof

⇒ Producing validity proof (zkSNARK proof) is expensive though

Two potential problems of rollup

Problem 1: what if coordinator is dishonest?

- It could steal funds from the Rollup contract
- It could issue fake Tx on behalf of users

Optimistic rollup or zk-rollup

Problem 2: what if coordinator stops providing service?

- If Rollup state is lost, how can we initialize a new coordinator?

Data availability committee

What’s next?

- Remaining issues
  - Mature rollup technologies?
  - Censorship in rollups?
  - L3?
  - …

Two parts

- Part I: Blockchain Technology: Advanced (L1/L2, ZKP, Sharding, etc)
  - by Min Suk Kang (SoC, KAIST)

After the break…

- Part II: How complicated it is to build a blockchain platform
  - by Sangmin Seo (Director, Klaytn Foundation)