OpenST Mosaic

Running meta-blockchains on Ethereum to scale DApps to billions of users

ETHBERLIN
HACKATHON / WORKSHOPS / TALKS

7 September, 2018 ben@ost.com @benjaminbollen # ostmosaic

Introducing OpenST Mosaic

- Today is the first time we dive into details of **OpenST Mosaic.**

- **OpenST Protocol** [Oct 2017 v0.9.0 - present v0.9.4] aims to **onboard a billion new users onto Ethereum** by **tokenizing mainstream applications.**

- **Requirements:**
  - Scalable programmable money (EVM) for on-chain token rules
  - Easy for millions of novice users to adopt today.

  * This week first partners on mainnet with v0.9.2

- **Plasma:** scale each app off-chain; data-availability problem
- **Payment channels** don't have smart contracts
- **Sharding** + **Casper** Ethereum = Ethereum v2.0 (future work + high stakes)

- **Mosaic is token-sharding at layer2 on ETH v1.0 (and later ETH v2.0)**

In Next 20 Minutes…

- OpenST Mosaic is a **consensus protocol** to run **meta-blockchains** on top of Ethereum.
- Each **meta-blockchain** runs in **parallel** and injects an **additional state space** into Ethereum.
- Each meta-blockchain is **secured by Ethereum** with an **open, staked validator** set.
- **All transactions** on meta-blockchains are **asynchronously finalised** and **committed onto Ethereum.**
- Mosaic has a **message-passing** protocol between Ethereum and meta-blockchains (later directly between meta-blockchains).

Problem We Are Trying to Solve

Existing Mainstream Apps with Millions of Users	→	Blockchains don’t scale
DApps	→	Same problem

Problem We Are Trying to Solve

Existing Mainstream Apps with Millions of Users → Blockchains don’t scale

DApps → Same problem

- but **tx/s finalised** can scale **linearly** with **# nodes**
  1. if we can **shard the total state space** across nodes
  2. and **offset** the **overhead** incurred by **more nodes**.

- **Solution:** Asynchronously derive the security from ETH (PoW) to finalise **auxiliary chains**.

Simple Calculation (only took me 4 years)

Ethereum @ 10 tx/s

simplest toy model:

100x (AUX @ 100 tx/s) = 10.000 tx/s

100 nodes / AUX * 100 AUX * 1 tx / node => 10.000 tx to commit on Ethereum

@10 tx/s takes minimally 1000s on Ethereum at full capacity

but 10.000.000 tx processed on aux; or 99.90% efficient

*but we can do **even better**:*

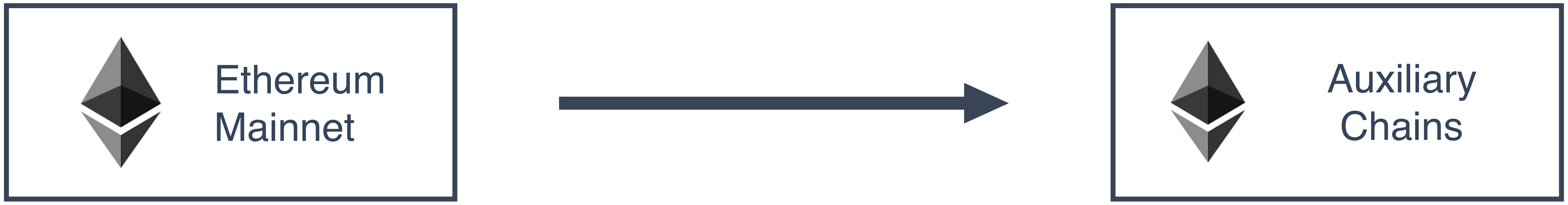
for N nodes, and constant #nodes/AUX

if there is **no time-constraint** to commit to Ethereum

cost: $\frac{\text{tx}}{s} \sim \mathcal{O}\left(\frac{N}{\Delta t}\right)$ benefit: $\frac{\text{tx}}{s} \sim \mathcal{O}(\alpha N)$

e.g. commit once daily, to finalise 864 millions transactions, for 1.2% of Ethereum's daily capacity!

Move Bulk of Transactions Off Ethereum & Onto Auxiliary Chains!

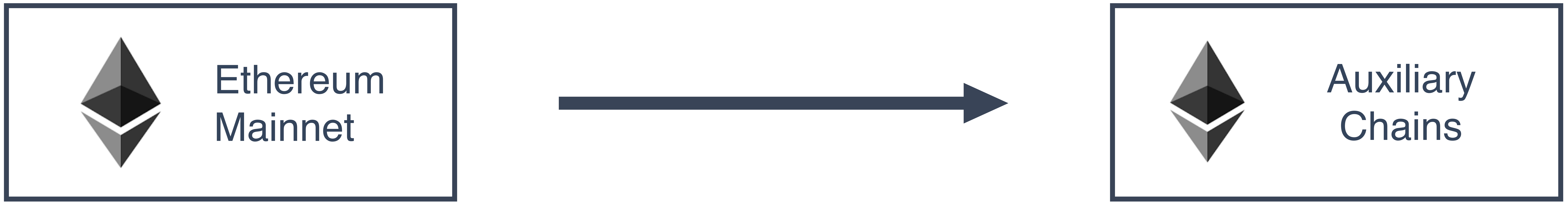


Move Bulk of Transactions Off Ethereum & Onto Auxiliary Chains!

Ethereum Mainnet
Auxiliary Chains

Hold on, sidechains have been around for years! What's new here?

Move Bulk of Transactions Off Ethereum & Onto Auxiliary Chains!



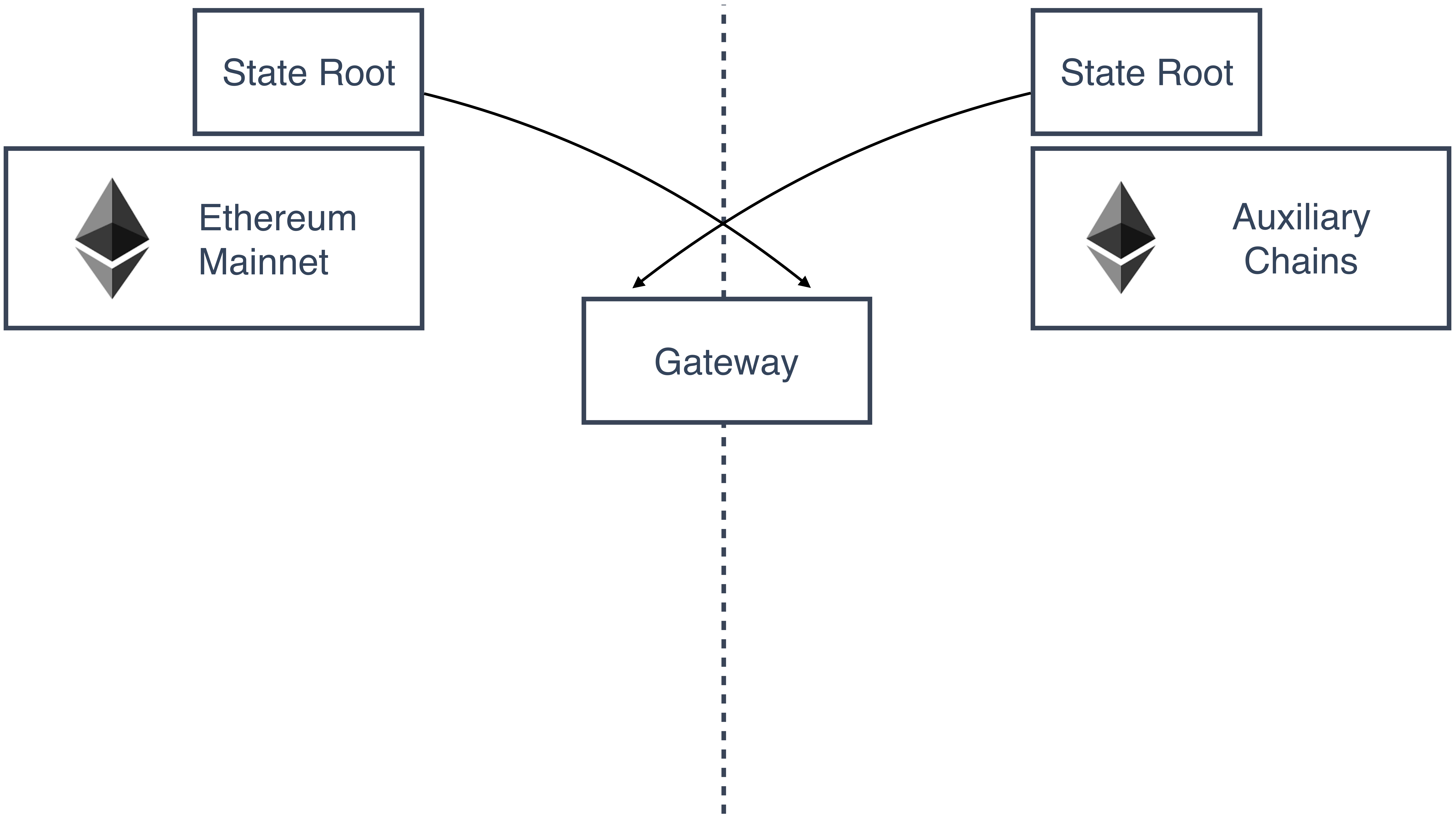
A **meta-blockchain** is **fully defined** with a consensus engine **in a core contract** on Ethereum.

Each meta-blockchain has an **open, staked validator set**.

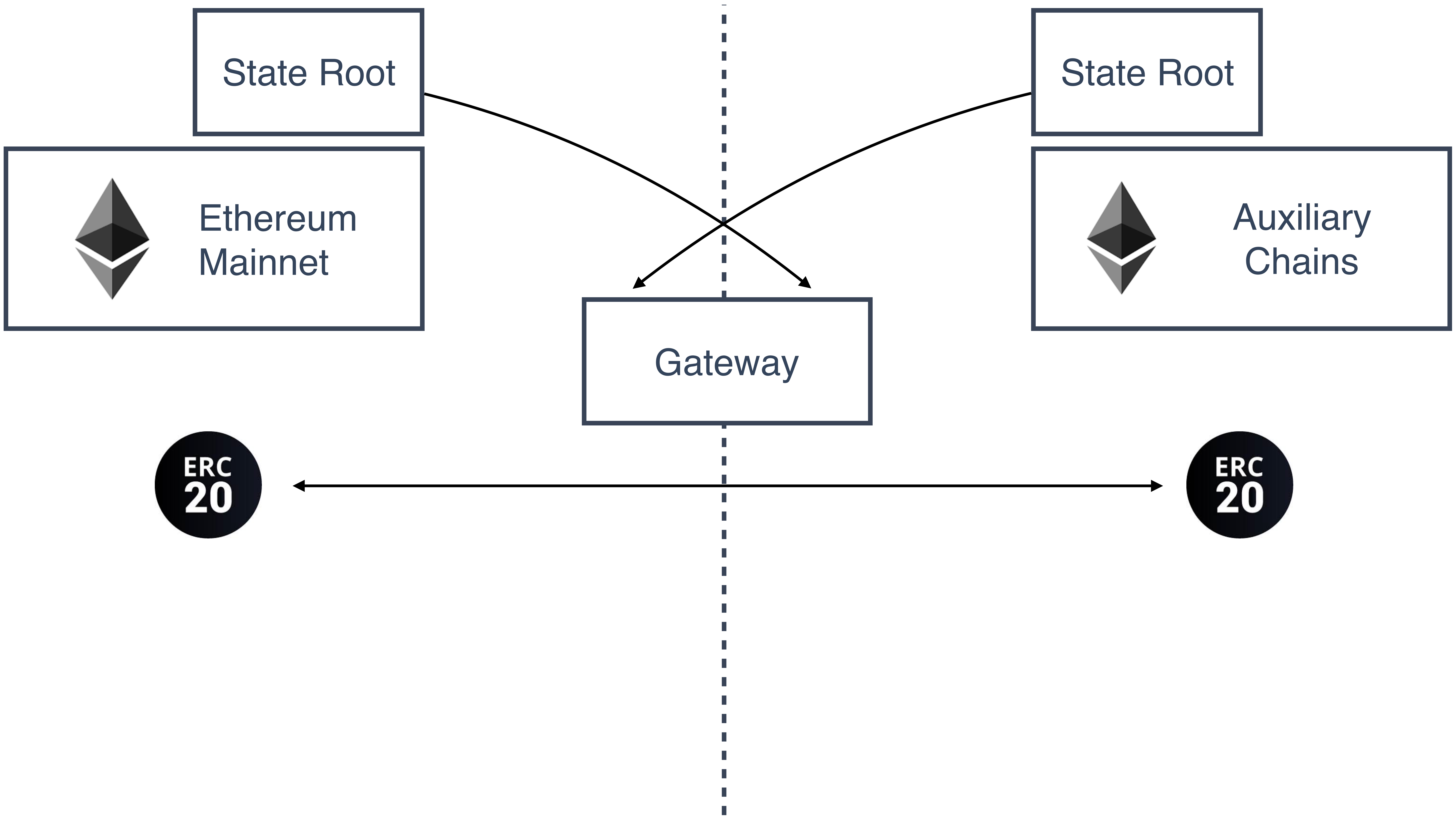
Mosaic's **BFT consensus rules** allow the meta-blockchain to offload **block formation** onto an auxiliary chain.

Gateways allow **ERC20** tokens to move in/out of meta-blockchains, **all value kept on Ethereum**.

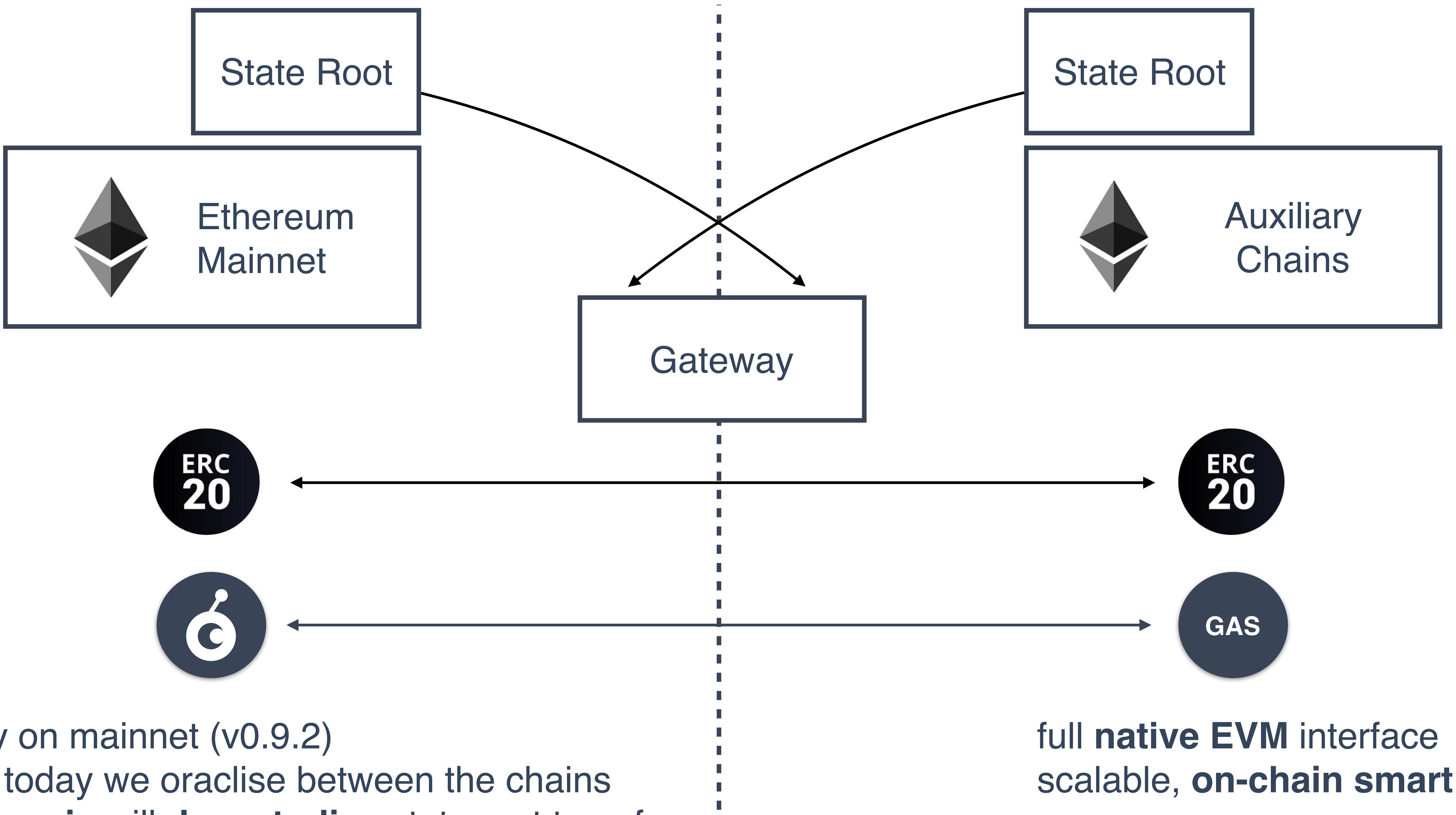
Token-Sharding



Token-Sharding



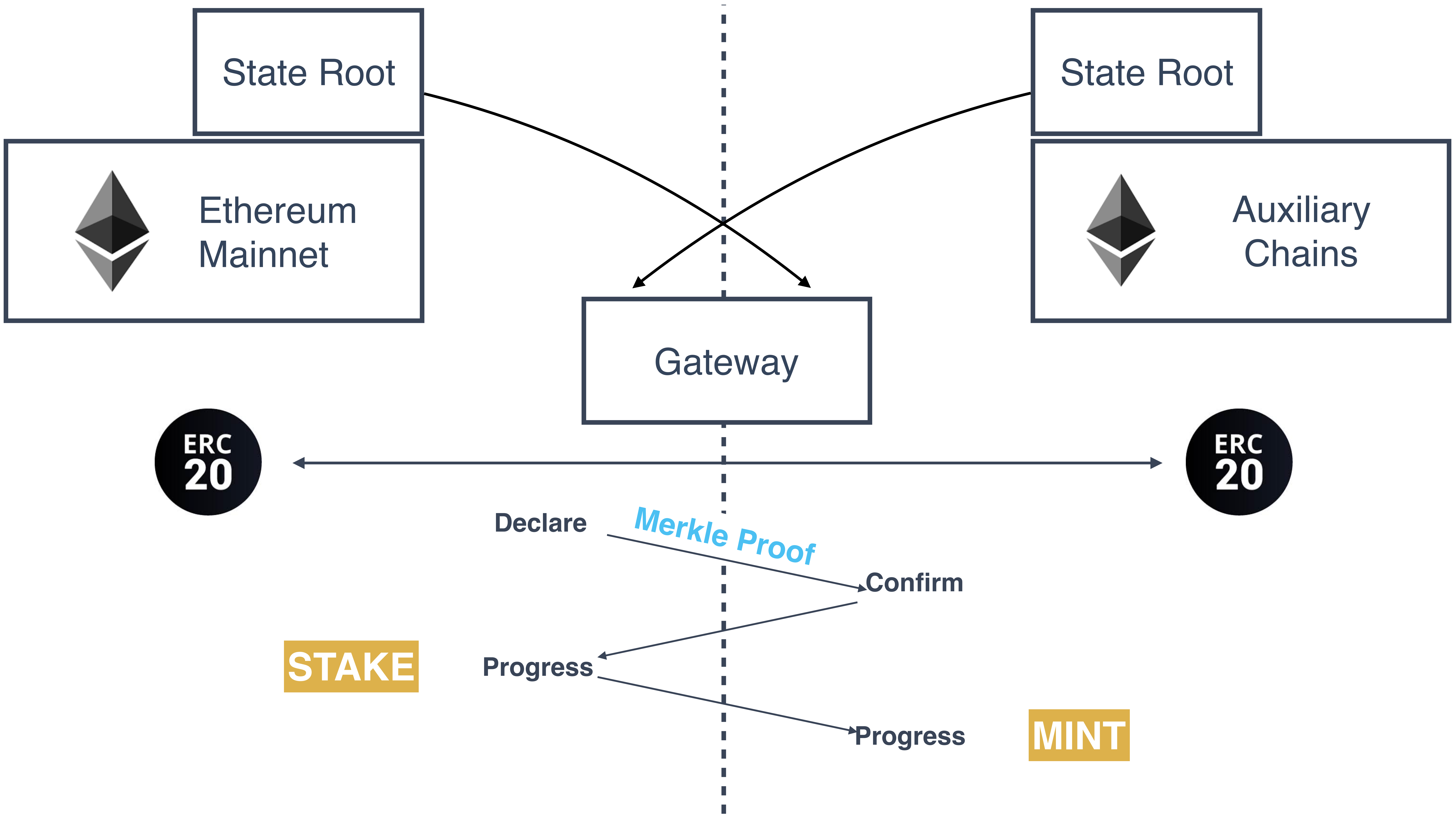
Token-Sharding



today on mainnet (v0.9.2)
BUT today we oraclise between the chains
=> **Mosaic** will **decentralise** state root transfer

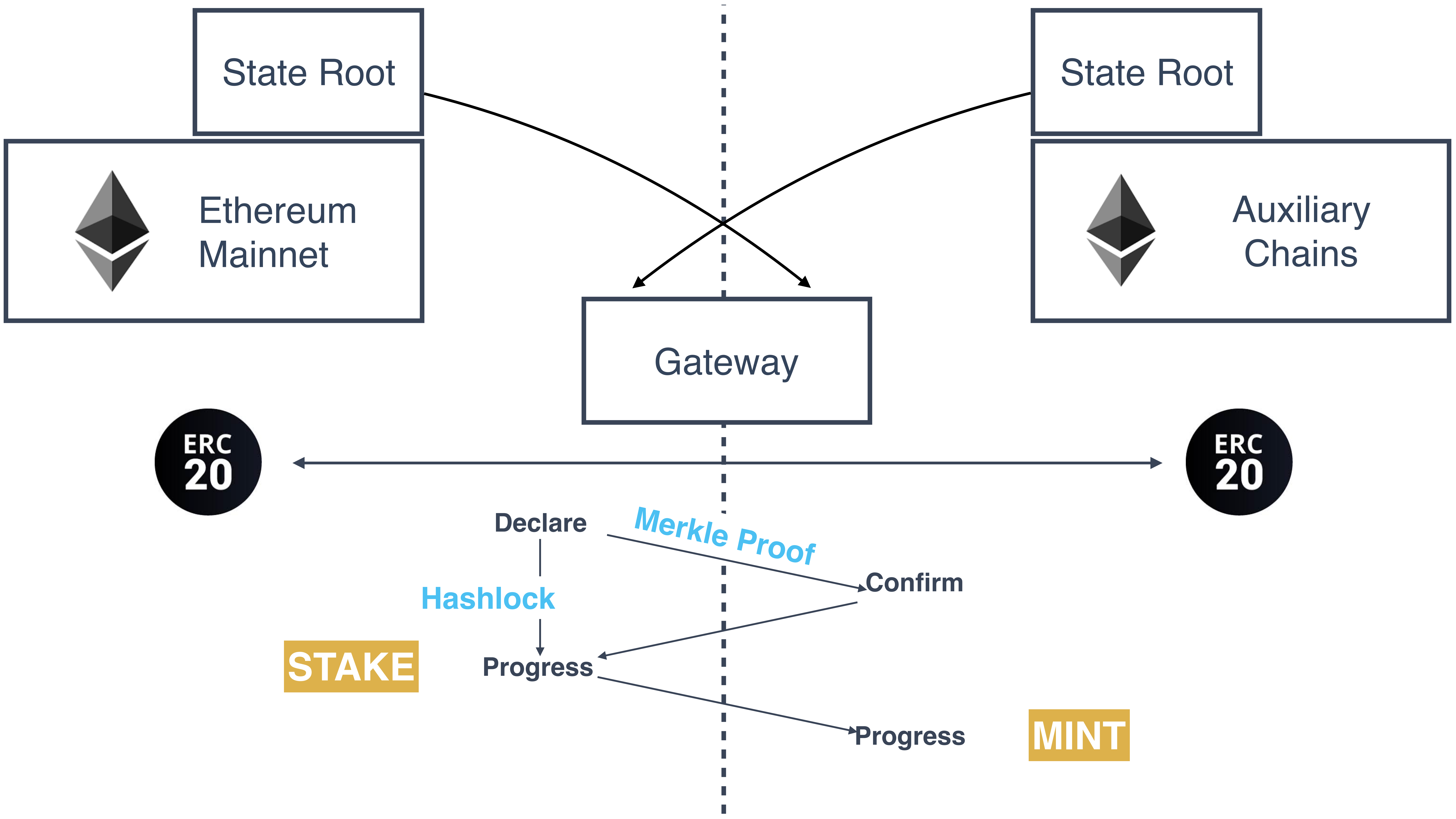
full **native EVM** interface
scalable, **on-chain smart contracts**

Token-Sharding



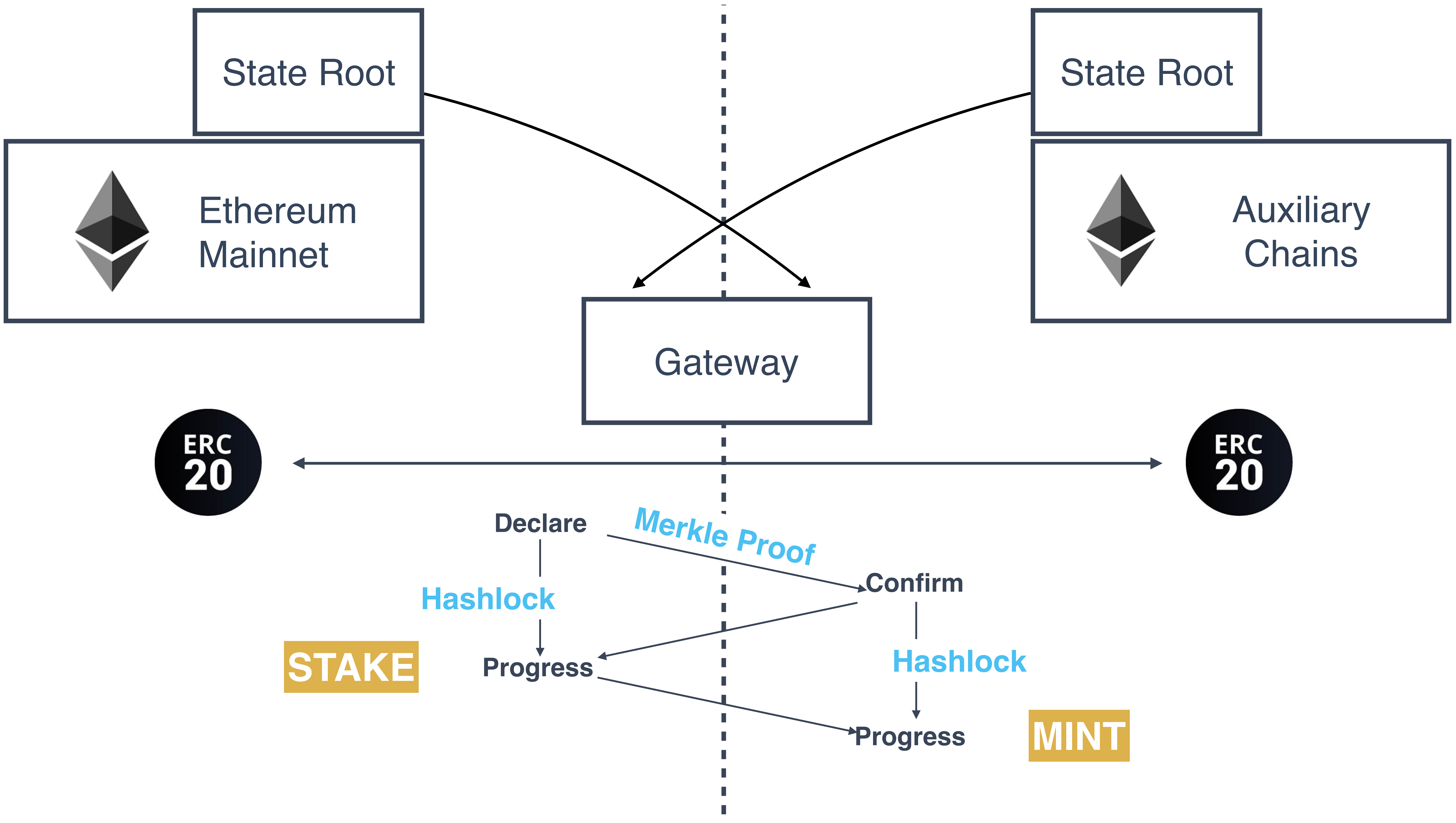
Message passing is **open** process with **bounties and fees** for nodes to complete the flow for all messages.

Token-Sharding



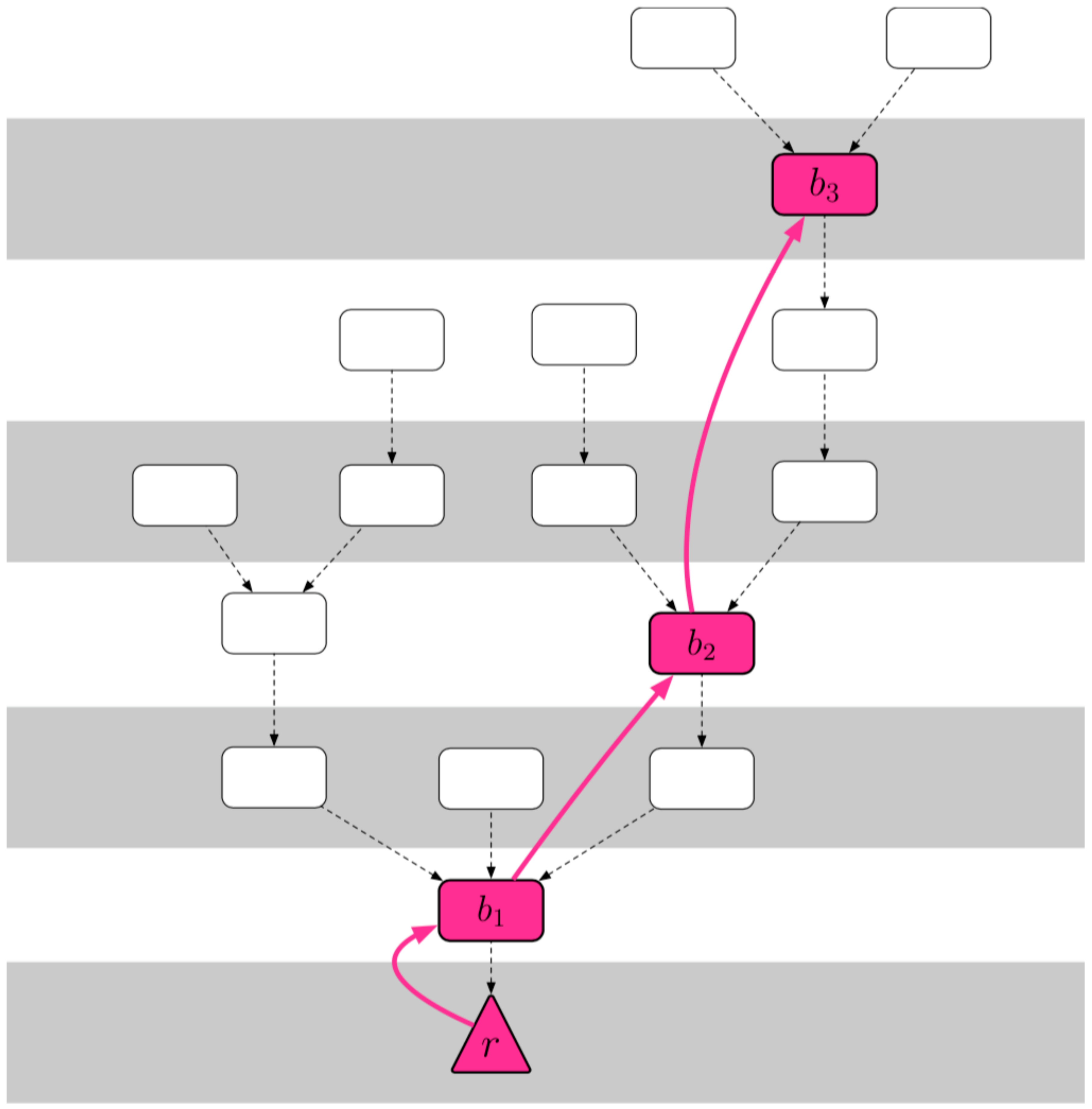
Message passing is **open** process with **bounties and fees** for nodes to complete the flow for all messages.

Token-Sharding



Message passing is **open** process with **bounties and fees** for nodes to complete the flow for all messages.

Intermezzo: Casper the Friendly Finality Gadget



(c) The justified chain $r \to b_1 \to b_2 \to b_3$

Casper the Friendly Finality Gadget,
V. Buterin, V. Griffith, nov 2017

validators can send vote messages:

$$\langle s, t, h(s), h(t) \rangle_v$$

checkpoint **justified** iff **exists supermajority link** from previously justified checkpoint

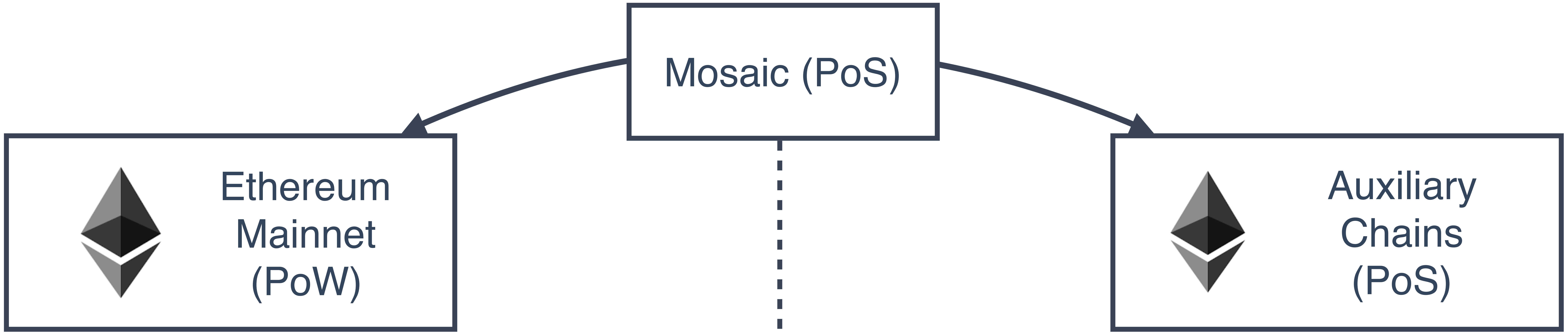
checkpoint **finalised** iff **direct child is justified**

slashing conditions, a validator **must not** publish

$$h(t_1) = h(t_2) \lor h(s_1) < h(s_2) < h(t_2) < h(t_1)$$

accountable **safety** and plausible **liveliness**

Proposing Meta-Blocks on Ethereum

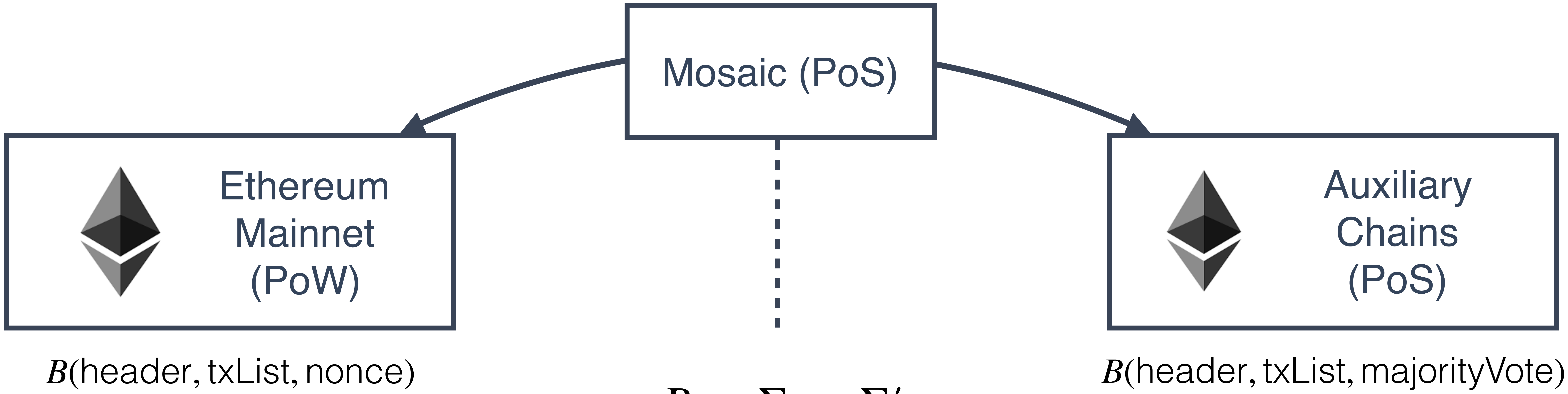


B(header, txList, nonce)

B(header, txList, majorityVote)

$$B_M : \Sigma \to \Sigma'$$

Proposing Meta-Blocks on Ethereum



$$B_M : \Sigma \rightarrow \Sigma'$$

Meta-Block

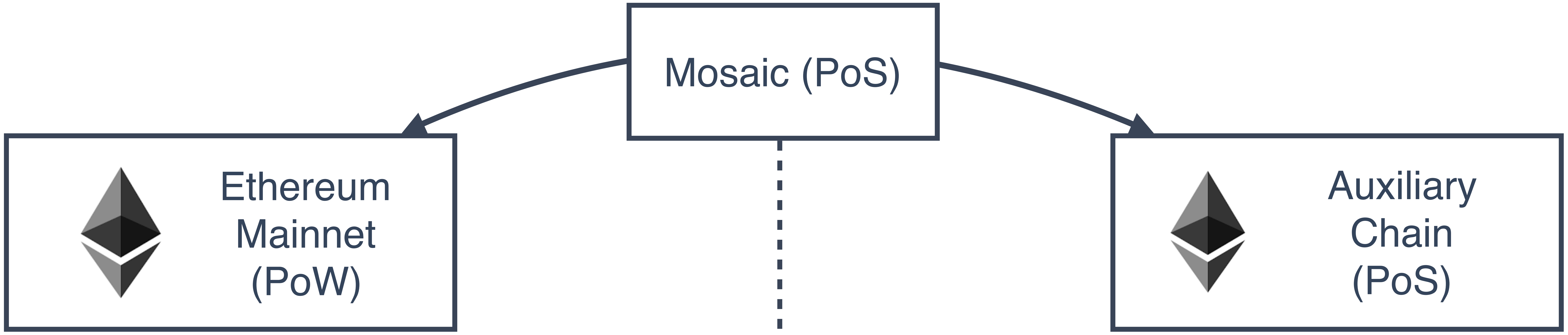
$$B_M(\text{kernel } K, \text{transition } T, \text{seal } S)$$

$K = \{\text{height, parent, } \Delta\text{validatorWeights, gasPrice}\}$

$T = \{\text{dynasty, txRoot, gasUsed, coreId}\}$

$S = \{\frac{+2}{3}\text{weightedVotes}\}$

Proposing Meta-Blocks on Ethereum



in Core contract meta-block is committed:

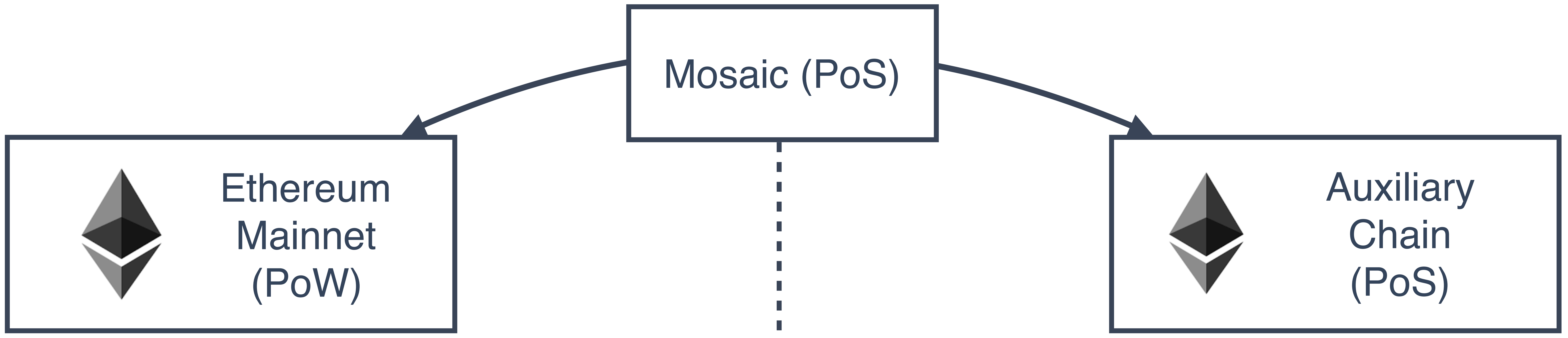
$$B_{h-1}(K_{h-1}, T_{d_{h-1}}, S_{h-1})$$

$$B_h(K_h, \cdot\,, \cdot\,)$$

$K = \{\text{height}, \text{parent}, \Delta\text{validatorWeights}, \text{gasPrice}\}$

$T = \{\text{dynasty}, \text{txRoot}, \text{gasUsed}, \text{coreId}\}$

Proposing Meta-Blocks on Ethereum

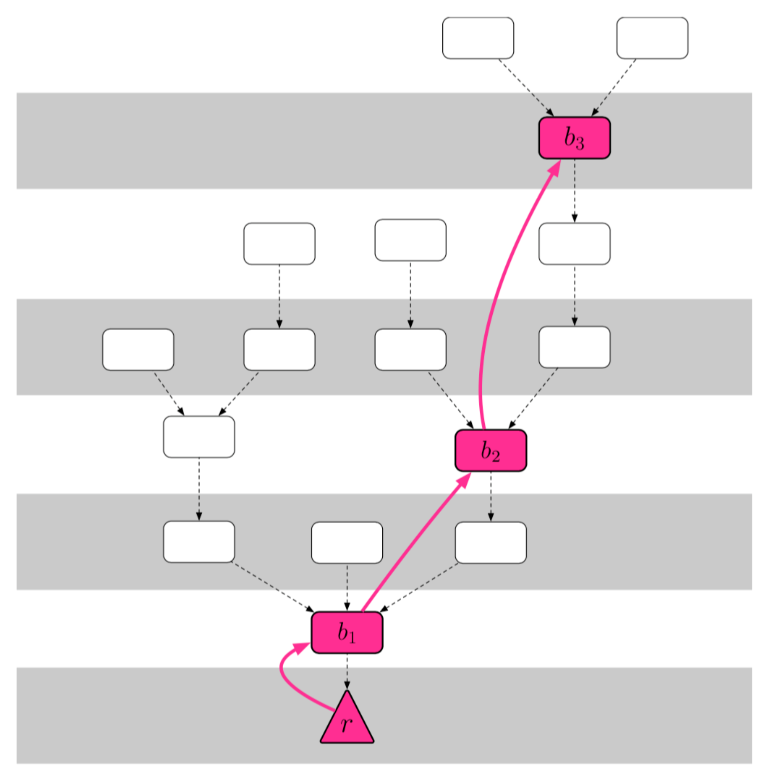


in Core contract meta-block is committed:

$$B_{h-1}(K_{h-1}, T_{d_{h-1}}, S_{h-1})$$

$$B_h(K_h, \cdot\,, \cdot\,)$$

finalise checkpoints:



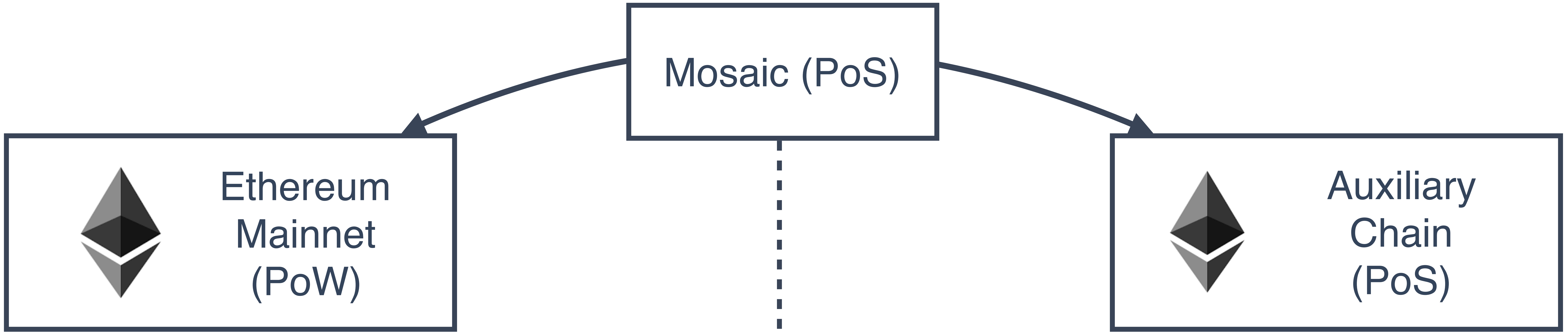
(c) The justified chain $r \to b_1 \to b_2 \to b_3$

in blockstore contract calculate T:
gas and txRoot for justified chain

$K = \{\text{height}, \text{parent}, \Delta\text{validatorWeights}, \text{gasPrice}\}$

$T = \{\text{dynasty}, \text{txRoot}, \text{gasUsed}, \text{coreId}\}$

Proposing Meta-Blocks on Ethereum

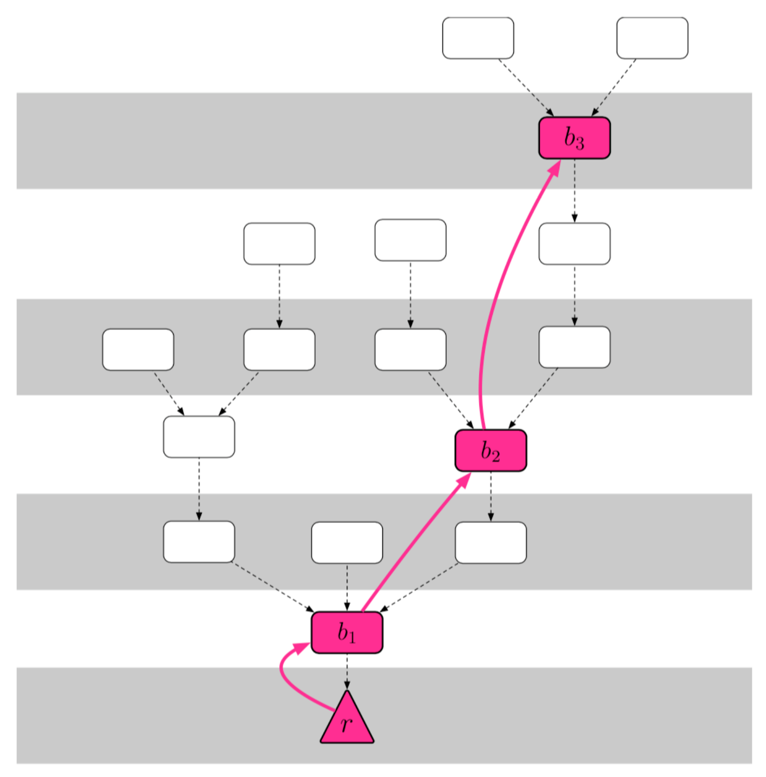


in Core contract meta-block is committed:

$$B_{h-1}(K_{h-1}, T_{d_{h-1}}, S_{h-1})$$

$$B_h(K_h, \cdot\ , \cdot\ )$$

finalise checkpoints:



(c) The justified chain $r \to b_1 \to b_2 \to b_3$

in blockstore contract calculate T: gas and txRoot for justified chain

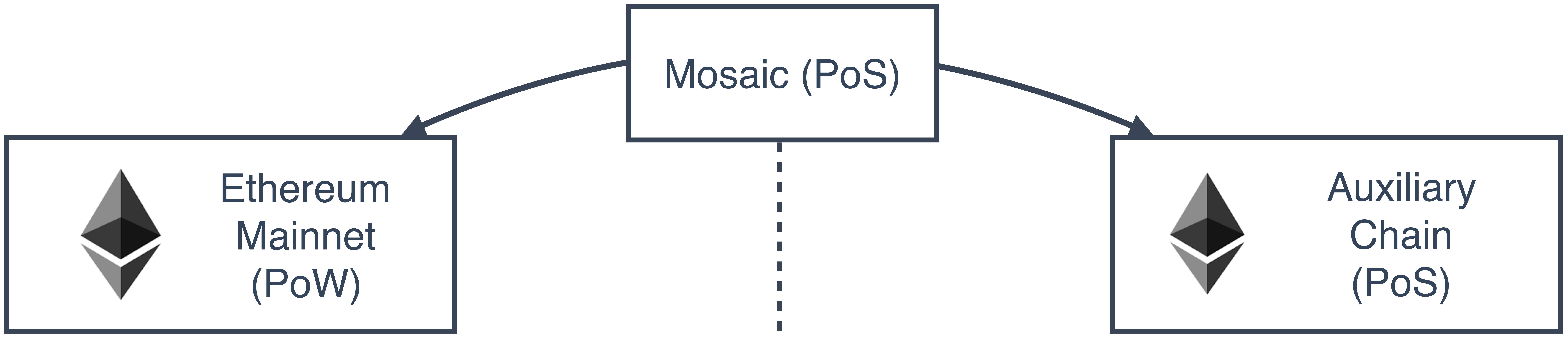
any **finalised checkpoint's T** can be used to **propose meta-block**

$$B_h(K_h, T_d, \cdot\ )$$

$K = \{\text{height}, \text{parent}, \Delta\text{validatorWeights}, \text{gasPrice}\}$

$T = \{\text{dynasty}, \text{txRoot}, \text{gasUsed}, \text{coreId}\}$

Proposing Meta-Blocks on Ethereum



in Core contract meta-block is committed:

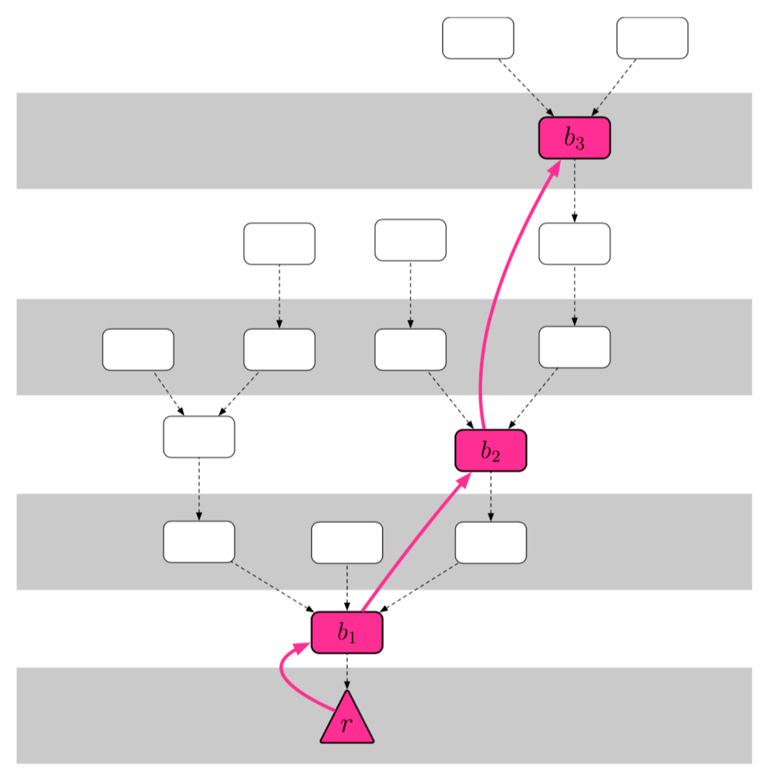
$$B_{h-1}(K_{h-1}, T_{d_{h-1}}, S_{h-1})$$

$$B_h(K_h, \cdot\, , \cdot\,)$$

propose new meta-block:

$$B_h(K_h, T_d, \cdot\,) \qquad B_h(K_h, T'_{d'}, \cdot\,)$$

finalise checkpoints:



(c) The justified chain $r \to b_1 \to b_2 \to b_3$

in blockstore contract calculate T: gas and txRoot for justified chain

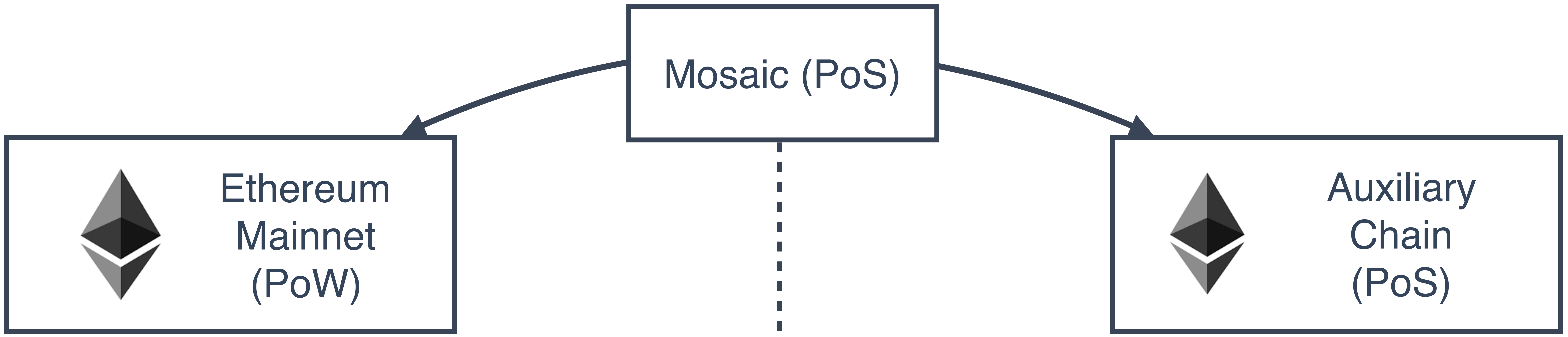
any **finalised checkpoint's T** can be used to **propose meta-block**

$$B_h(K_h, T_d, \cdot\,)$$

$K = \{\text{height}, \text{parent}, \Delta\text{validatorWeights}, \text{gasPrice}\}$

$T = \{\text{dynasty}, \text{txRoot}, \text{gasUsed}, \text{coreId}\}$

Committing Meta-Blocks on Ethereum



in Core contract meta-block is committed:

$$B_{h-1}(K_{h-1}, T_{d_{h-1}}, S_{h-1})$$

$$B_h(K_h, \cdot\,, \cdot\,)$$

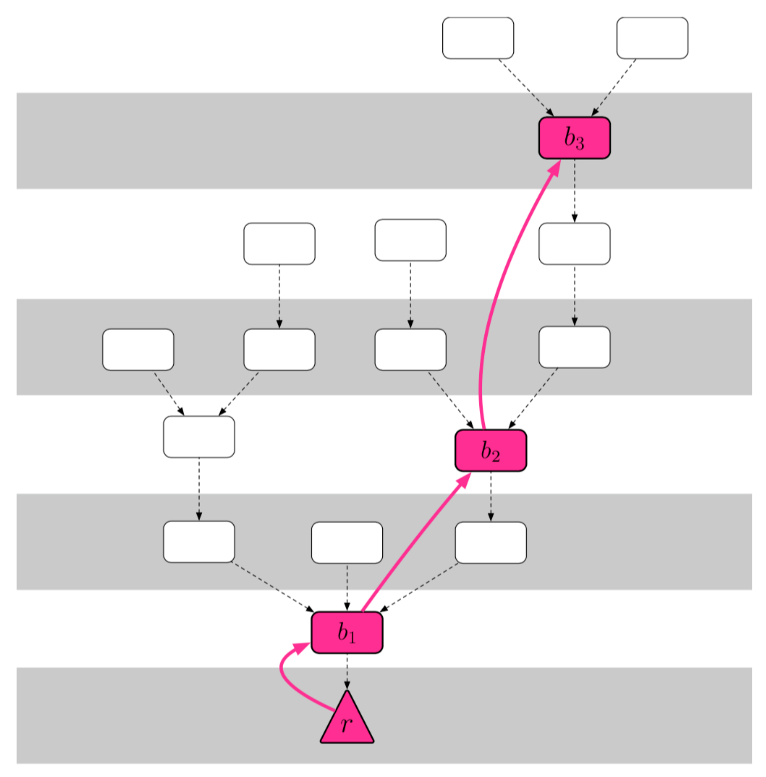
commit new meta-block:

$$B_h(K_h, T_{d_h}, S_h) \quad B_h(K_h, T'_{d'}, \cdot\,)$$

$$\text{with } S_h = \frac{+2}{3} Maj_v \left\{ \langle T_{d_h}, s, t, h(s), h(t) \rangle_v \right\}$$

$$K = \{\text{height}, \text{parent}, \Delta\text{validatorWeights}, \text{gasPrice}\}$$

finalise checkpoints:


(c) The justified chain $r \rightarrow b_1 \rightarrow b_2 \rightarrow b_3$

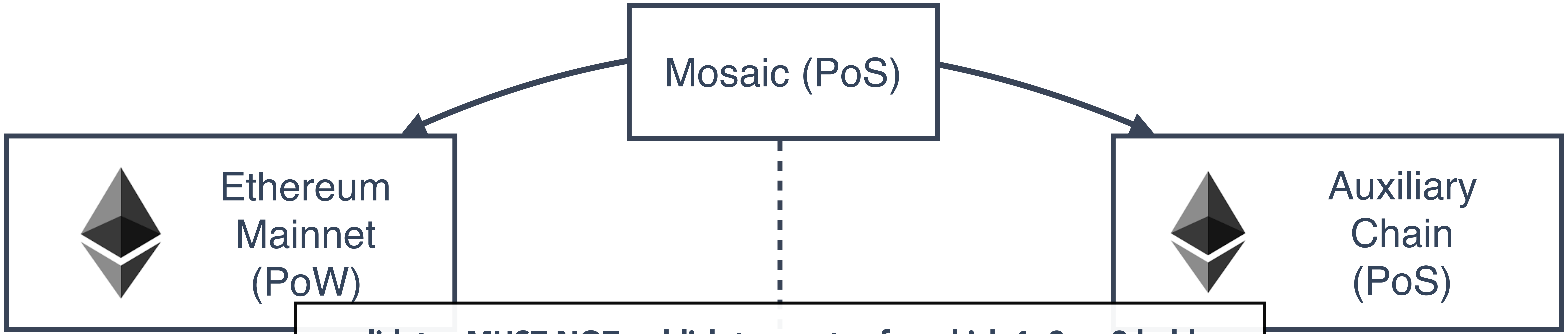
in blockstore contract calculate T: gas and txRoot for justified chain

any **finalised checkpoint's T** can be used to **propose meta-block**

$$B_h(K_h, T_d, \cdot\,)$$

$$T = \{\text{dynasty}, \text{txRoot}, \text{gasUsed}, \text{coreId}\}$$

Holding Validators Accountable



in Core contract meta-block is committed:

$B_{h-1}(K_{h-1}, T_{d_{h-1}}, S_{h-1})$

$B_h(K_h, \cdot\,, \cdot\,)$

commit new meta-block:

$B_h(K_h, T_{d_h}, S_h)$ $B_h(K_h, T'_{d'}, \cdot\,)$

with $S_h = \frac{+2}{3} Maj_v \left\{ \langle T_{d_h}, s, t, h(s), h(t) \rangle_v \right\}$

a validator MUST NOT publish two votes for which 1, 2 or 3 holds

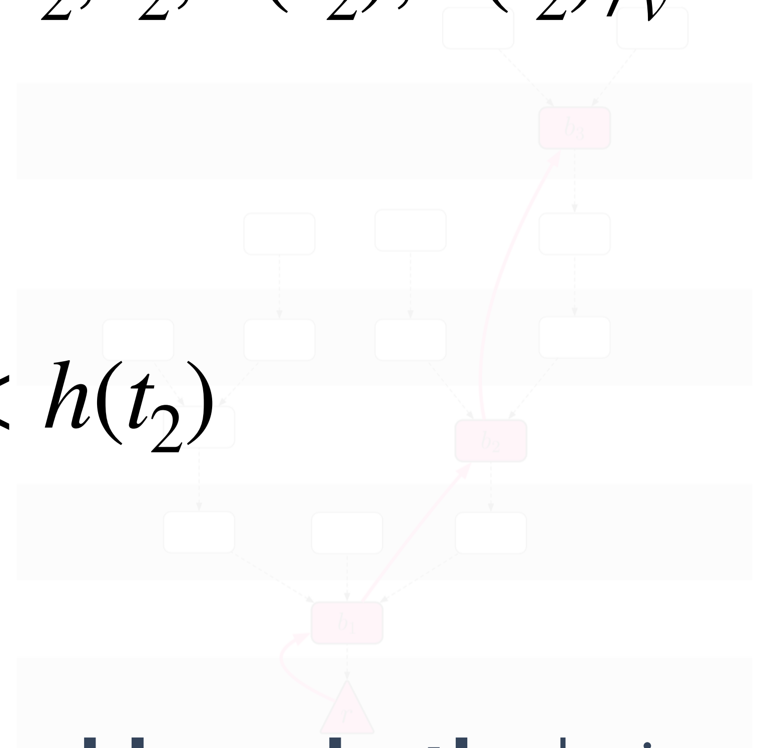
$\langle T_1, s_1, t_1, h(s_1), h(t_1) \rangle_v$ **and** $\langle T_2, s_2, t_2, h(s_2), h(t_2) \rangle_v$

1. $h(t_1) = h(t_2)$

2. $h(s_2) < h(s_1) < h(t_1) < h(t_2)$

3. $s_1 = s_2 \land T_1 \neq T_2$

slashing conditions are immediately **enforceable** on **both** chains.



finalise checkpoints:

in blockstore contract calculate T: gas and txRoot for justified chain

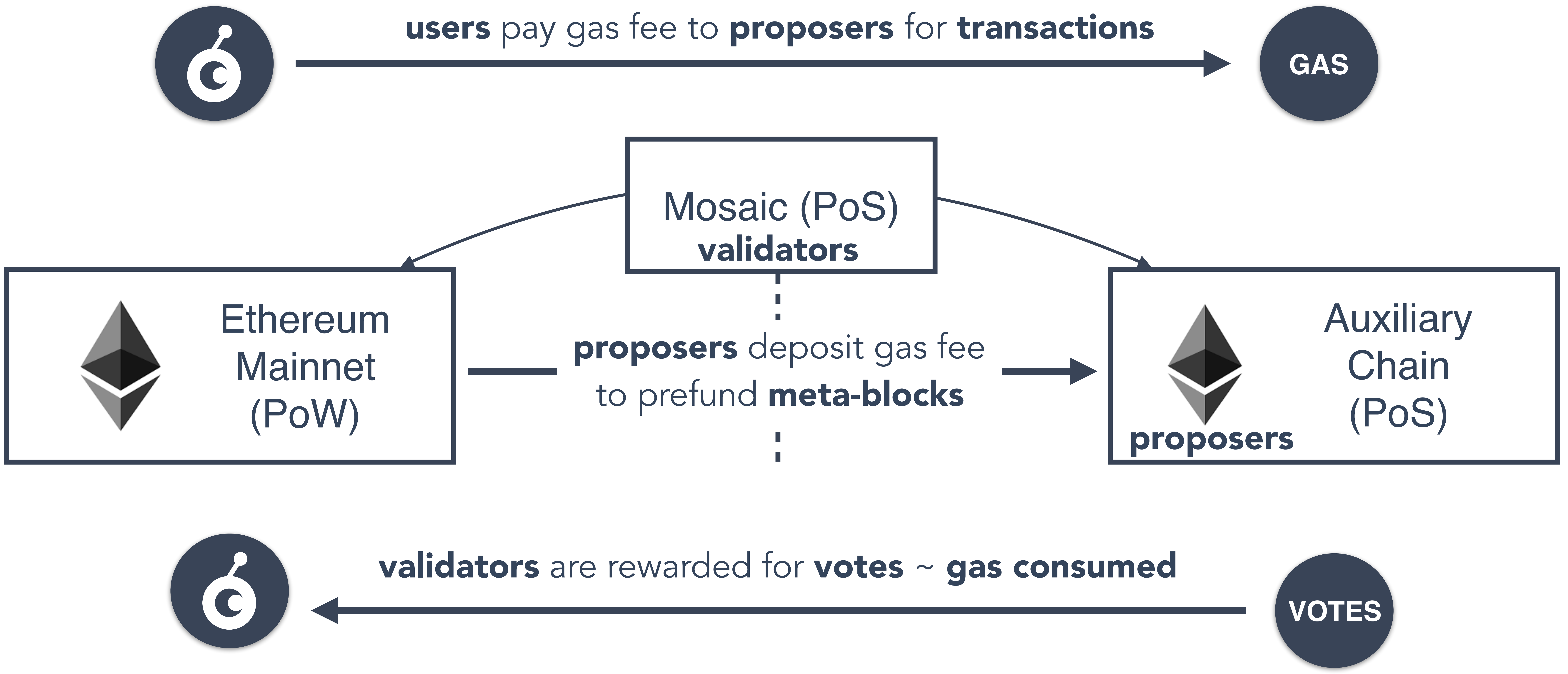
any **finalised checkpoint's T** can be used to **propose meta-block**

$B_h(K_h, T_d, \cdot\,)$

$K = \{\text{height}, \text{parent}, \Delta\text{validatorWeights}, \text{gasPrice}\}$

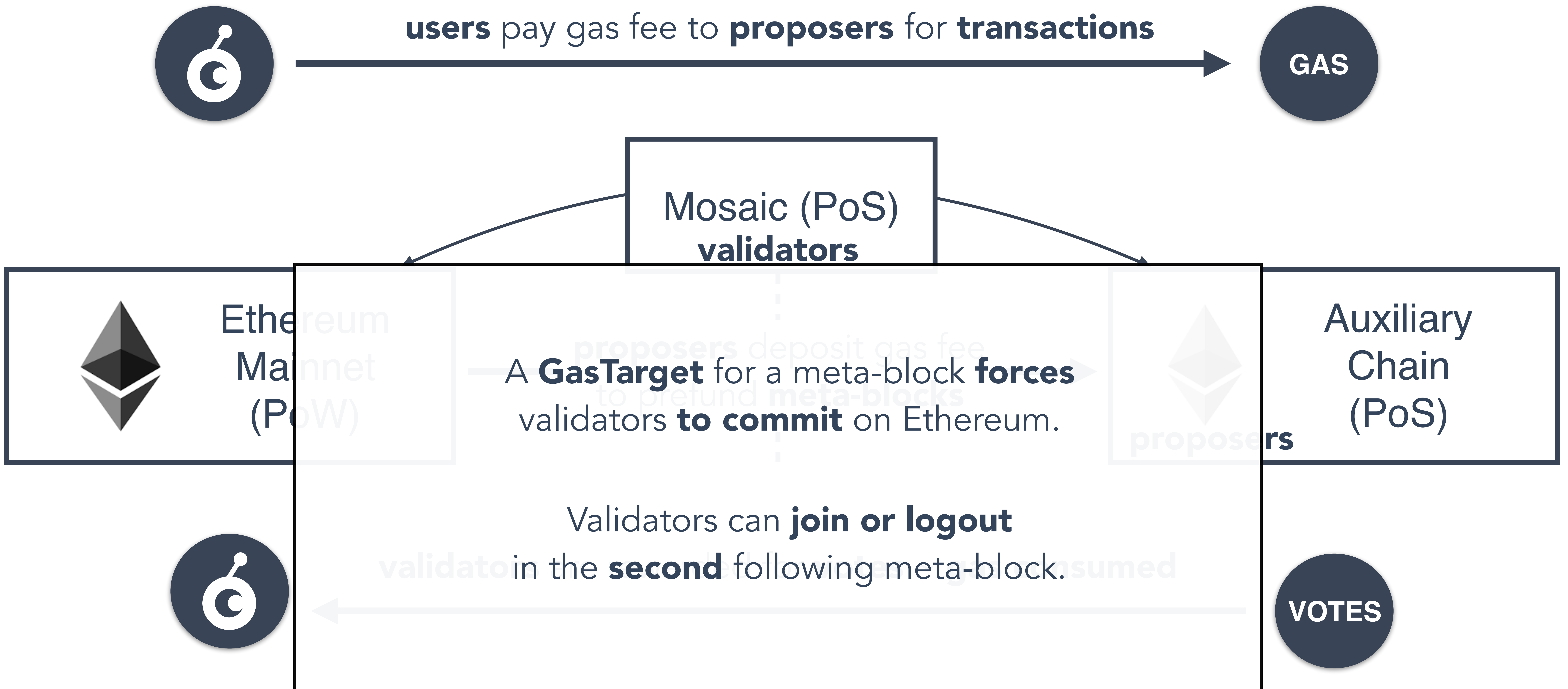
$T = \{\text{dynasty}, \text{txRoot}, \text{gasUsed}, \text{coreId}\}$

… but Validators Need Carrots too.


users pay gas fee to **proposers** for **transactions**
GAS
Mosaic (PoS)
validators
Ethereum
Mainnet
(PoW)
proposers deposit gas fee
to prefund **meta-blocks**
proposers
Auxiliary
Chain
(PoS)
validators are rewarded for **votes** ~ **gas consumed**
VOTES

validators must have **voted on auxiliary** and **committed on Ethereum** to get rewarded.

… but Validators Need Carrots too.



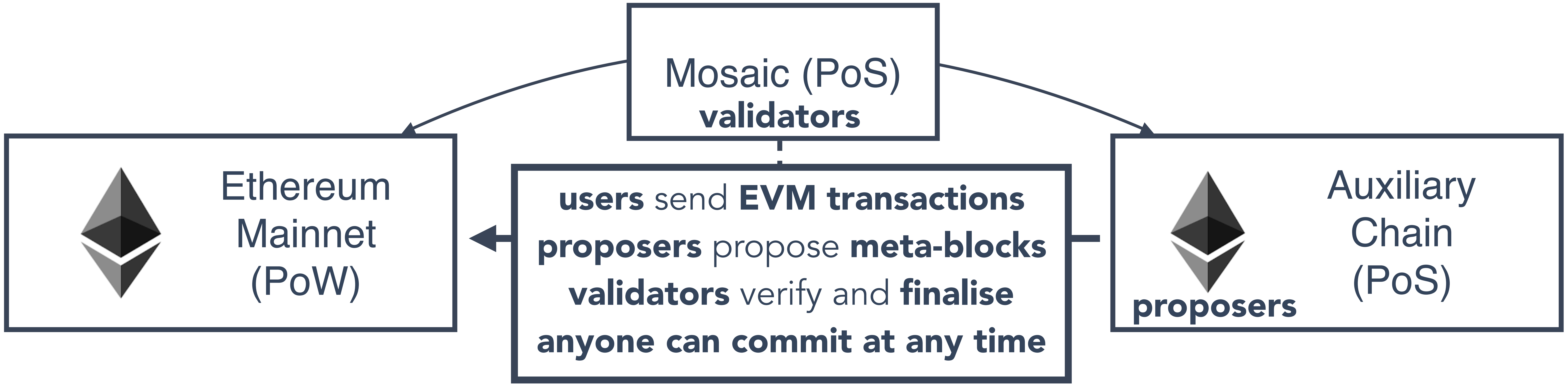
users pay gas fee to **proposers** for **transactions**
GAS
Mosaic (PoS) **validators**
Ethereum Mainnet (PoW)
proposers deposit gas fee to prefund **meta-blocks**
Auxiliary Chain (PoS)
proposers
VOTES

A **GasTarget** for a meta-block **forces** validators **to commit** on Ethereum.

Validators can **join or logout** in the **second** following meta-block.

validators must have **voted on auxiliary** and **committed on Ethereum** to get rewarded.

So what can we use this for?



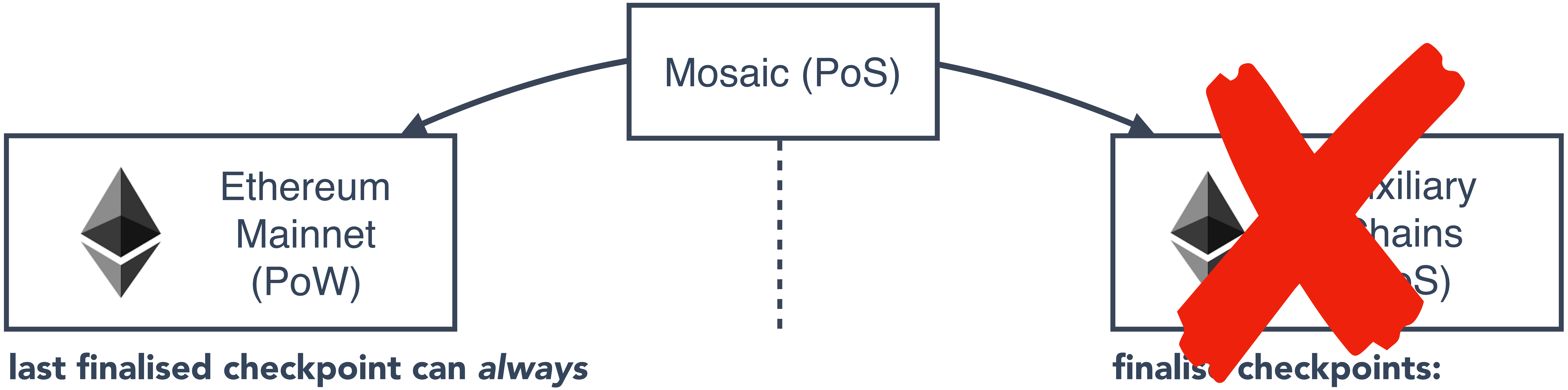
Mosaic (PoS)
validators
Ethereum Mainnet (PoW)
users send EVM transactions
proposers propose meta-blocks
validators verify and finalise
anyone can commit at any time
Auxiliary Chain (PoS)
proposers

- **Gateway** consumes the **state root finalised** by **validators**
  Users and **contracts** can pass messages (without a bridge); currently **ERC20-typed**

- for ERC20-tokens, once minted on the auxiliary chain,
  atomic swap is **fast** to move **across.**

- **DApps** (users) can **opt-in and -out** of auxiliary chains.

When it all goes wrong: Halting



last finalised checkpoint can *always* be committed by anyone.

finalis checkpoints:

validator set can be **challenged** that meta-blockchain has **halted.**

for halted core **all value held in gateways** on Ethereum can be recovered with ownership proofs against **latest** committed (equiv. **finalized**) meta-block. **No mass-exit required.**

Current Status - September 2018

- OST KIT & OpenST 0.9.2 used by **120+ external developer teams on Testnet**
- OST KIT & OpenST 0.9.2 used by **10+ external partners on Mainnet** — *3 months ahead of schedule*
- OpenST 0.9.3 **(cross-chain messaging)** — *completed*
- OpenST 0.9.4 **(decentralized keys engine & token rules engine)** — *in progress*
- **Introducing Mosaic at several events like ETH Berlin to solicit community feedback**

What’s Coming Next

Q3 2018:
- **Publish OpenST Mosaic Paper** for Community Feedback

Q4 2018:
- Continue **Mainnet testing** of OST, OpenST Protocol

2019:
- **Mosaic alpha**
- **Mosaic beta** - warm up validator pool
- **Widespread Deployments** of OST, OpenST, Mosaic to mainstream businesses and DApps
- **smart contract calls across auxiliary chains**
- **2nd-order meta-blocks** (aux-on-aux-on-ETHv1.0 or v2.0)

2020:
- **Mosaic fully released** to decentralised validator pool

Thank You !

Questions and feedback welcomed!

github.com/openstfoundation

ETHBERLIN
HACKATHON / WORKSHOPS / TALKS

7 September, 2018 ben@ost.com @benjaminbollen # ostmosaic