CS251 Fall 2025

(cs251.stanford.edu)

Scaling the blockchain part II: Rollups

Dan Boneh

Scaling the blockchain: the problem

Transaction rates (Tx/sec):

- Bitcoin: can process about **7 (Tx/sec)**
- Ethereum: can process about **15 (Tx/sec)**

$\Big\}$ Tx Fees fluctuate:

2$ to 60$ for simple Tx

- The visa network: can process up to **24,000 (Tx/sec)**

Can we scale blockchains to visa speeds? … with low Tx fees

How to process more Tx per second

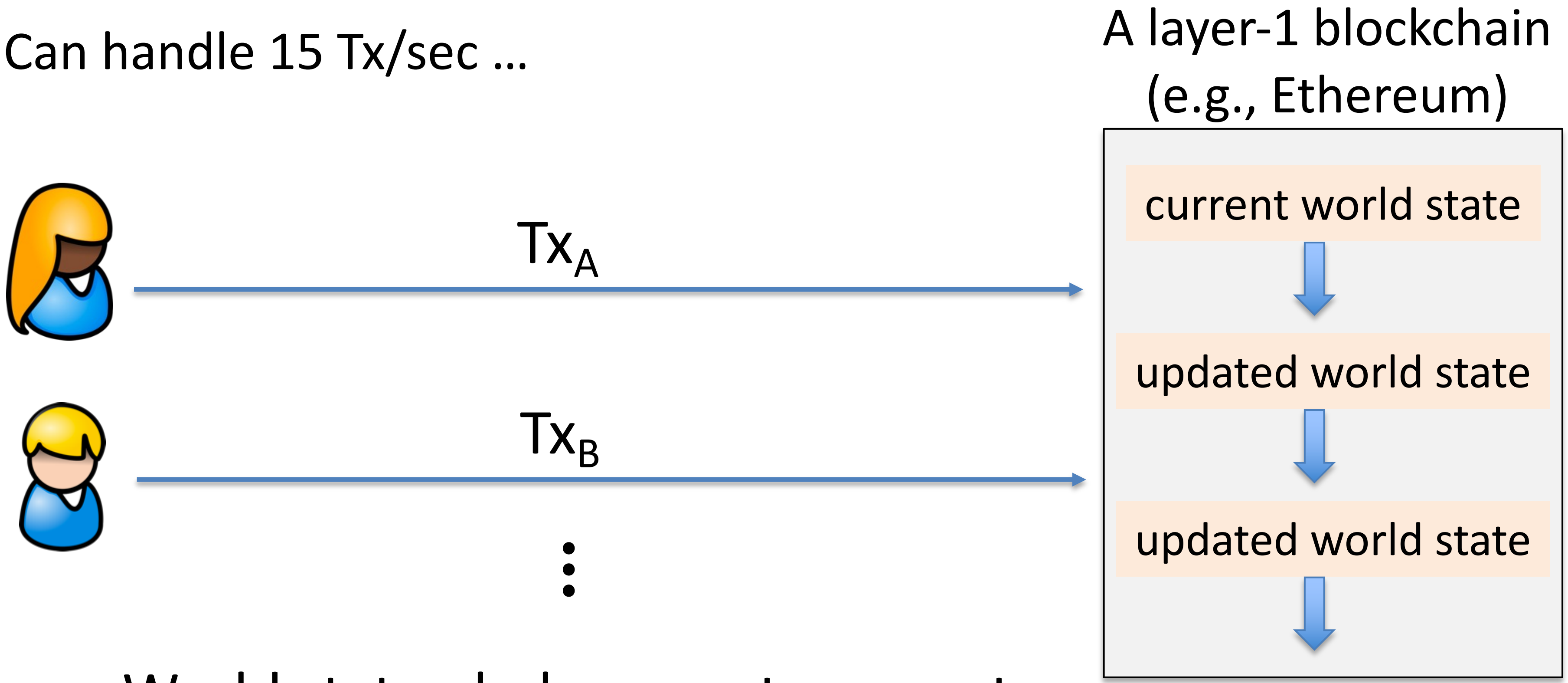
Many ideas:

reduces composability

- Use a faster consensus protocol
- Parallelize: split the chain into independent **shards**
- Today: Rollups, move the work somewhere else
- Payment channels: reduce the need to touch the chain
  - Requires locking up funds; mostly designed for payments.

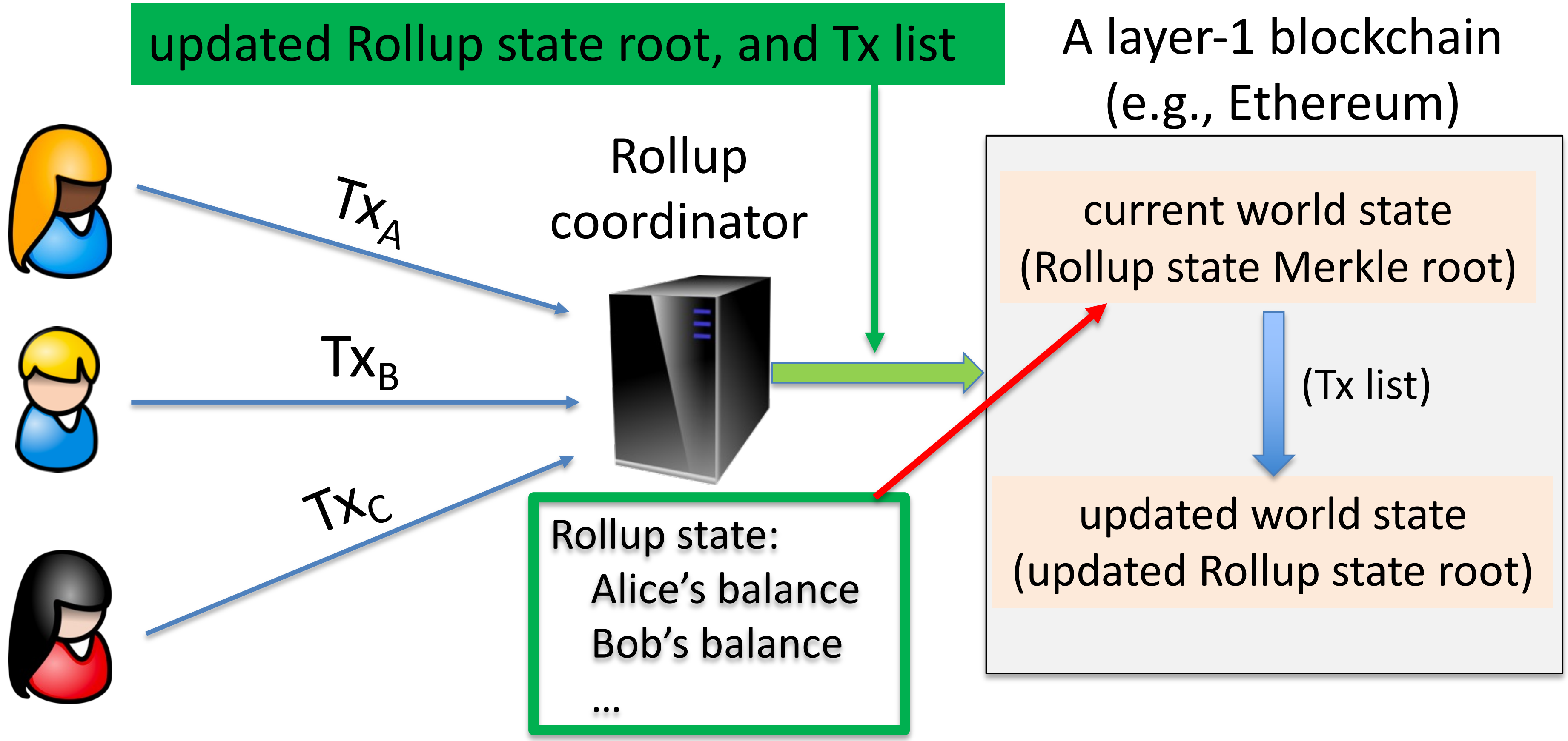
Recall: a basic layer-1 blockchain

Can handle 15 Tx/sec …



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
Tx_A
Tx_B
Tx_C
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:

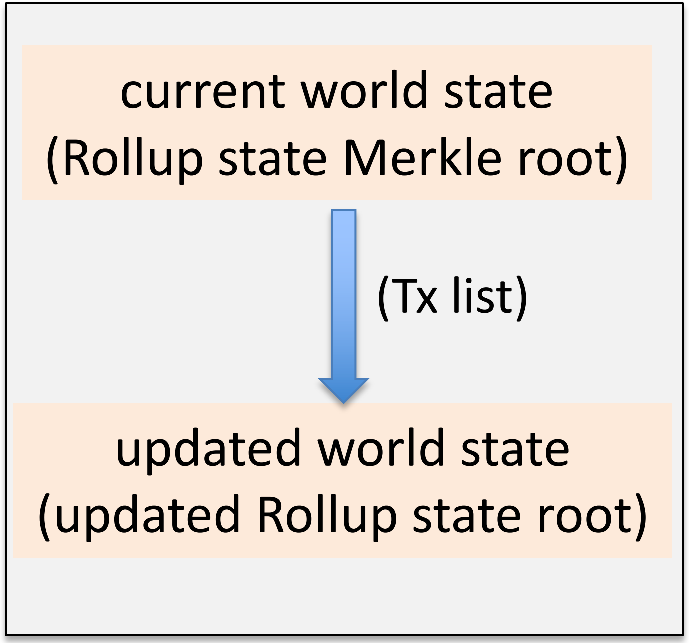
- *Hundreds* of transactions on Rollup state are batched into a *single* transaction on layer-1

  ⇒ 100x speed up in Tx/sec

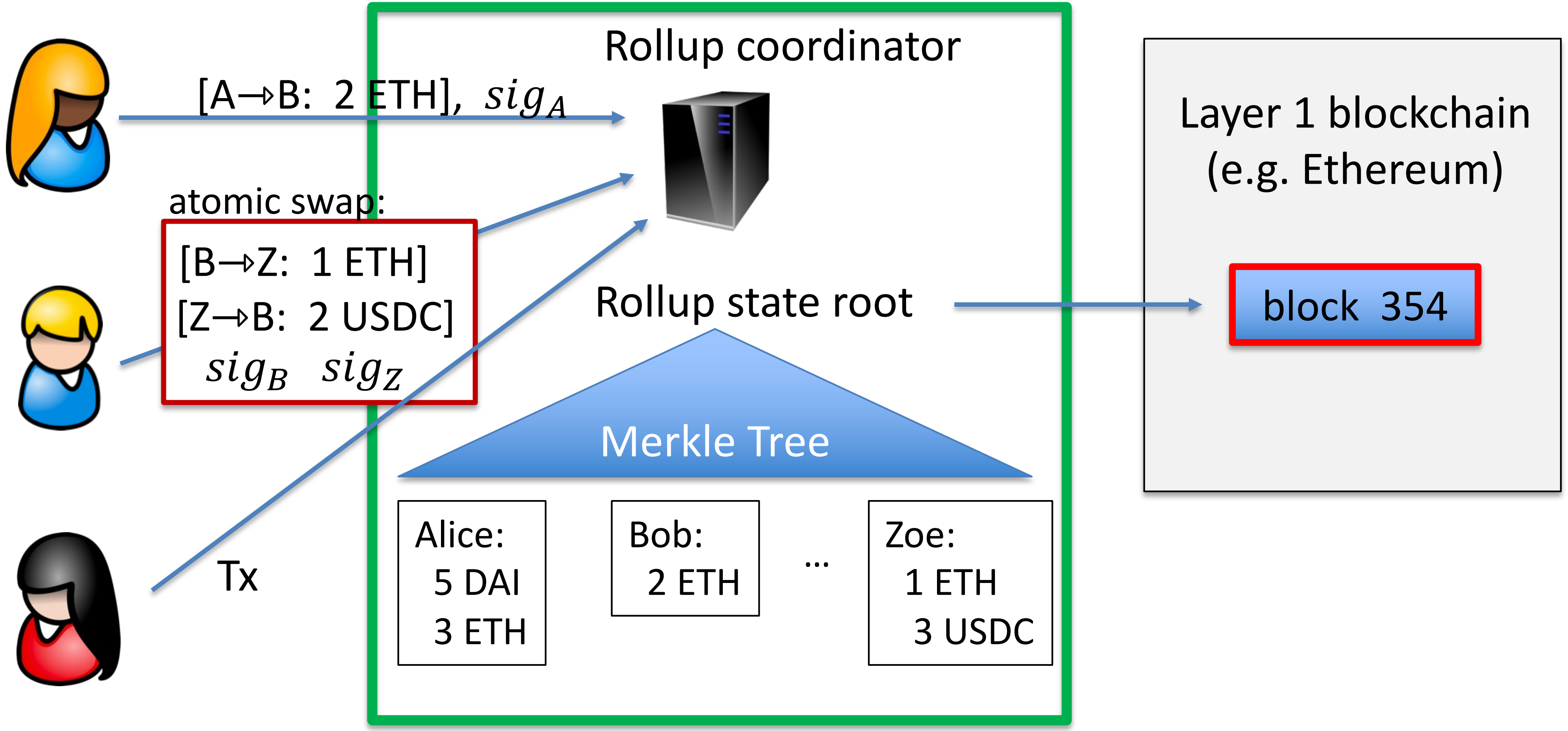
- Let's see how …

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

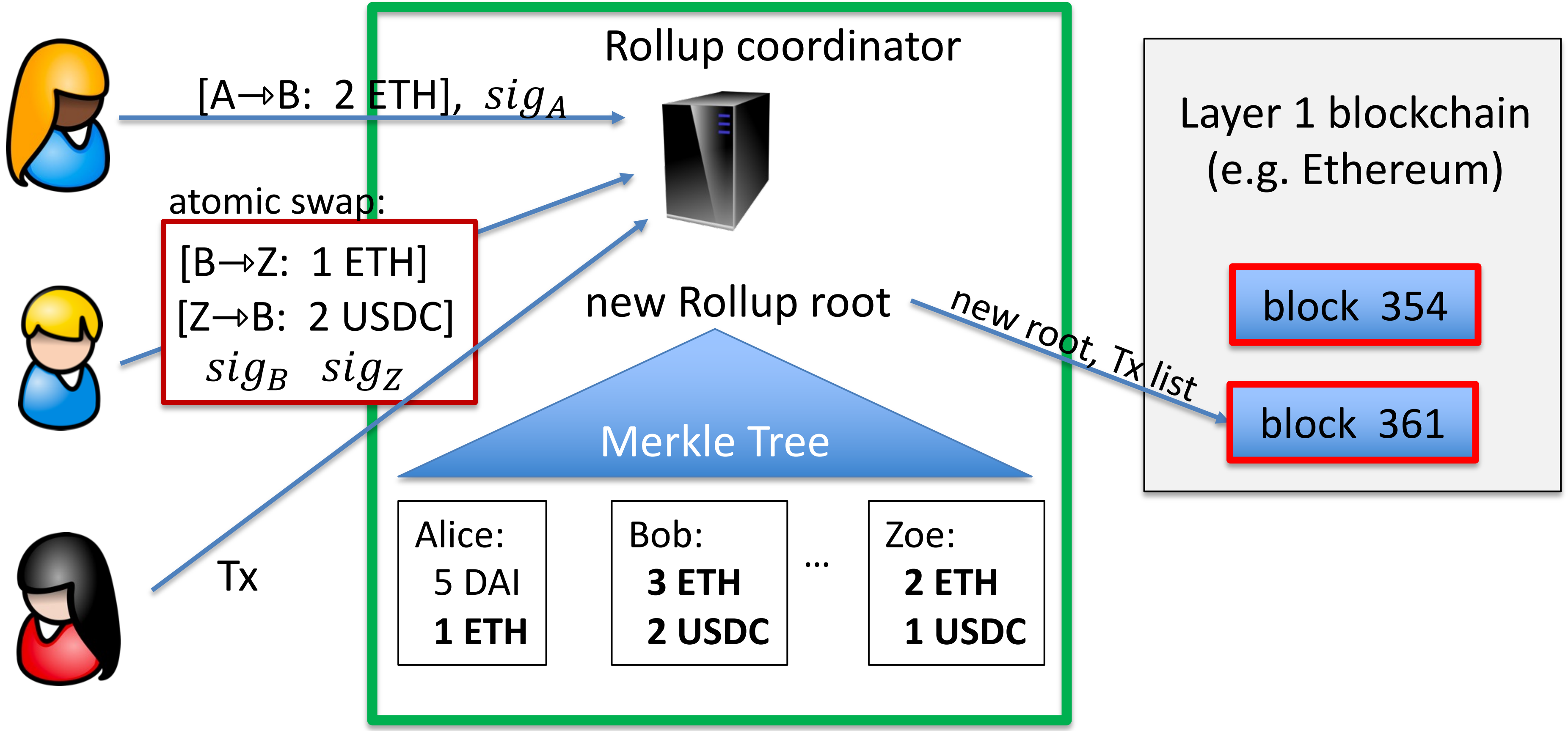
A layer-1 blockchain (e.g., Ethereum)



Rollup operation (simplified)

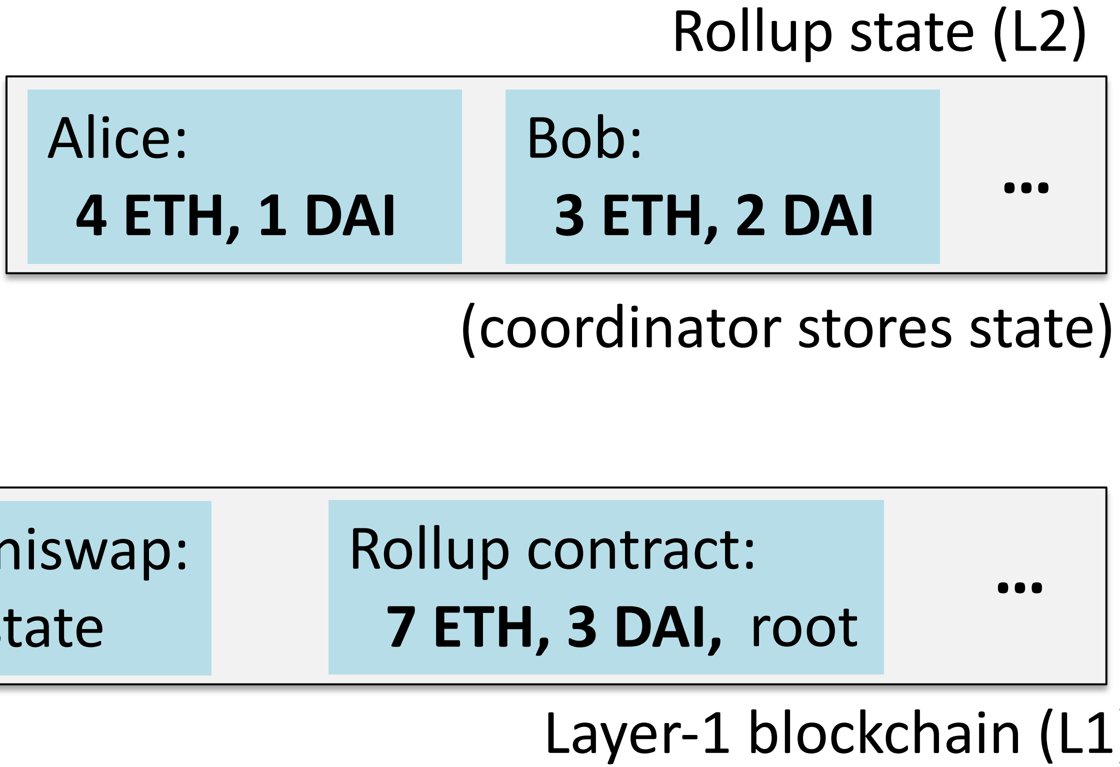


Rollup operation (simplified)

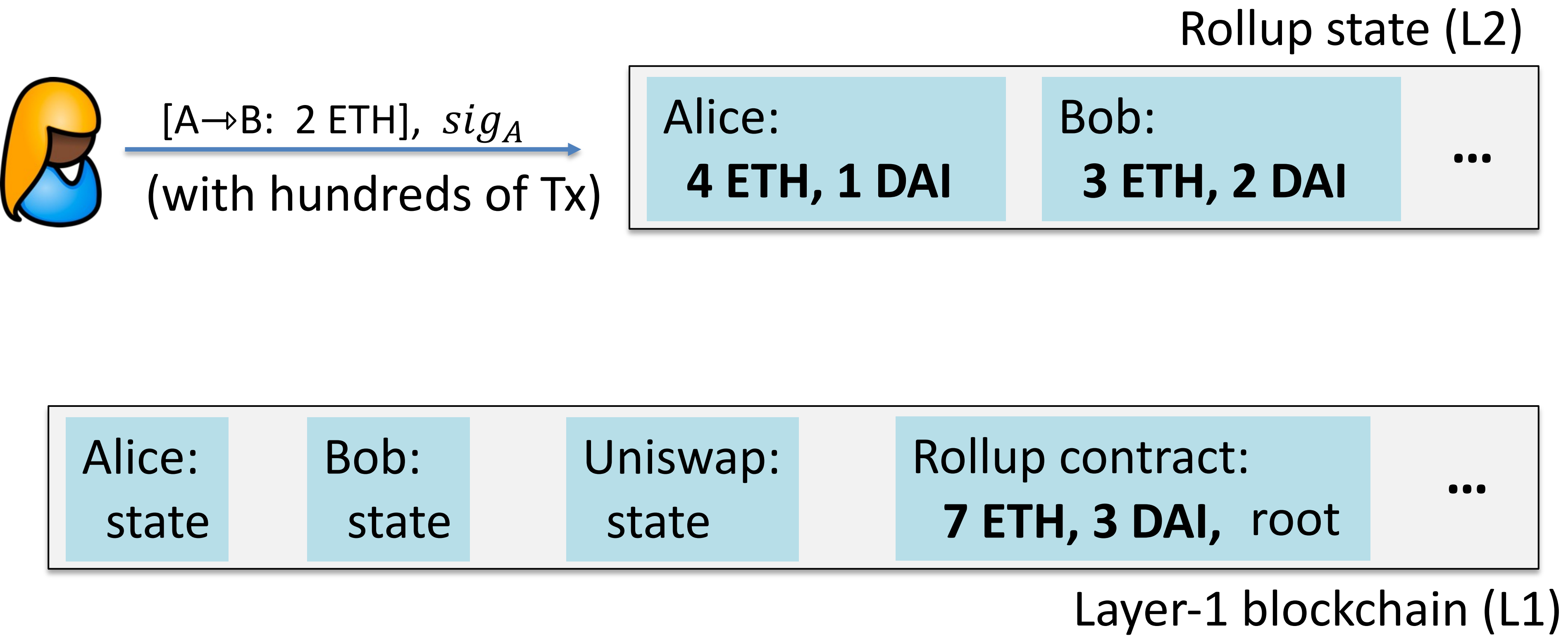


In more detail

Rollup contract on layer-1 holds assets of all Rollup accounts (and Merkle state root)

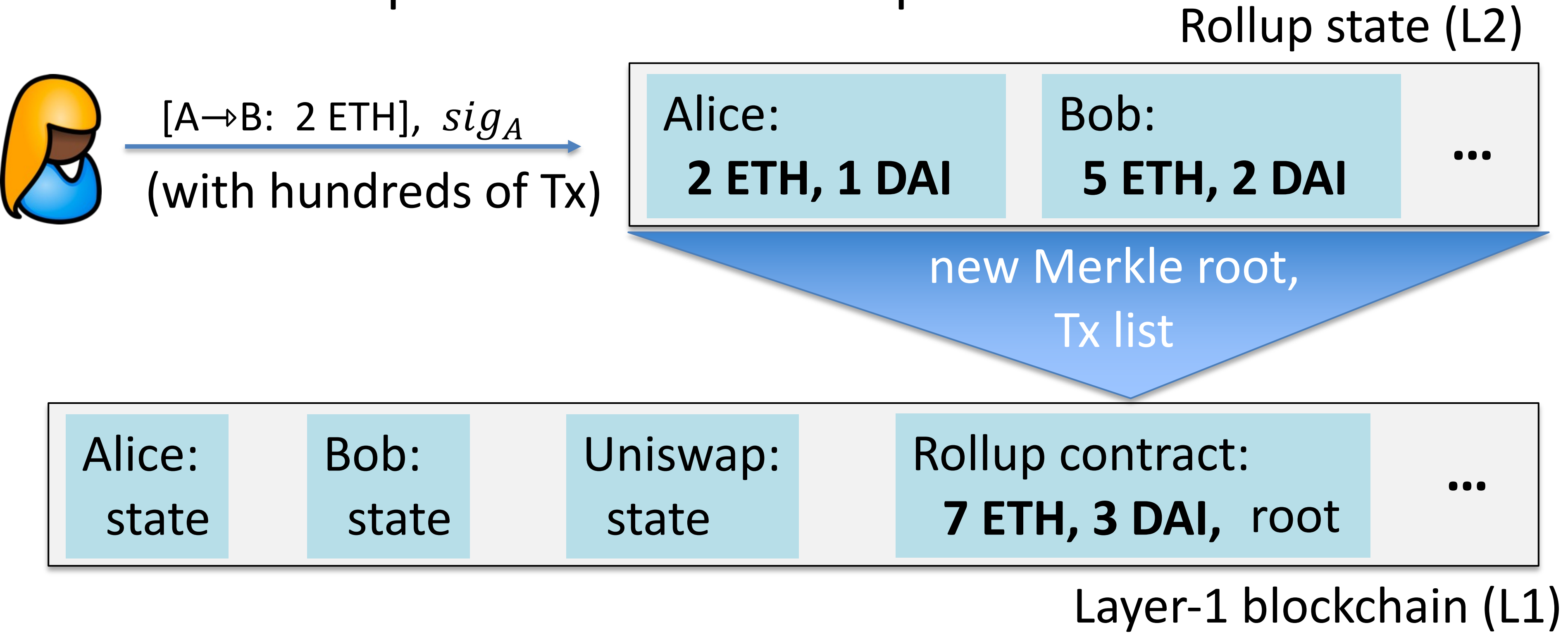


Transfers inside Rollup are easy (L2 → L2)



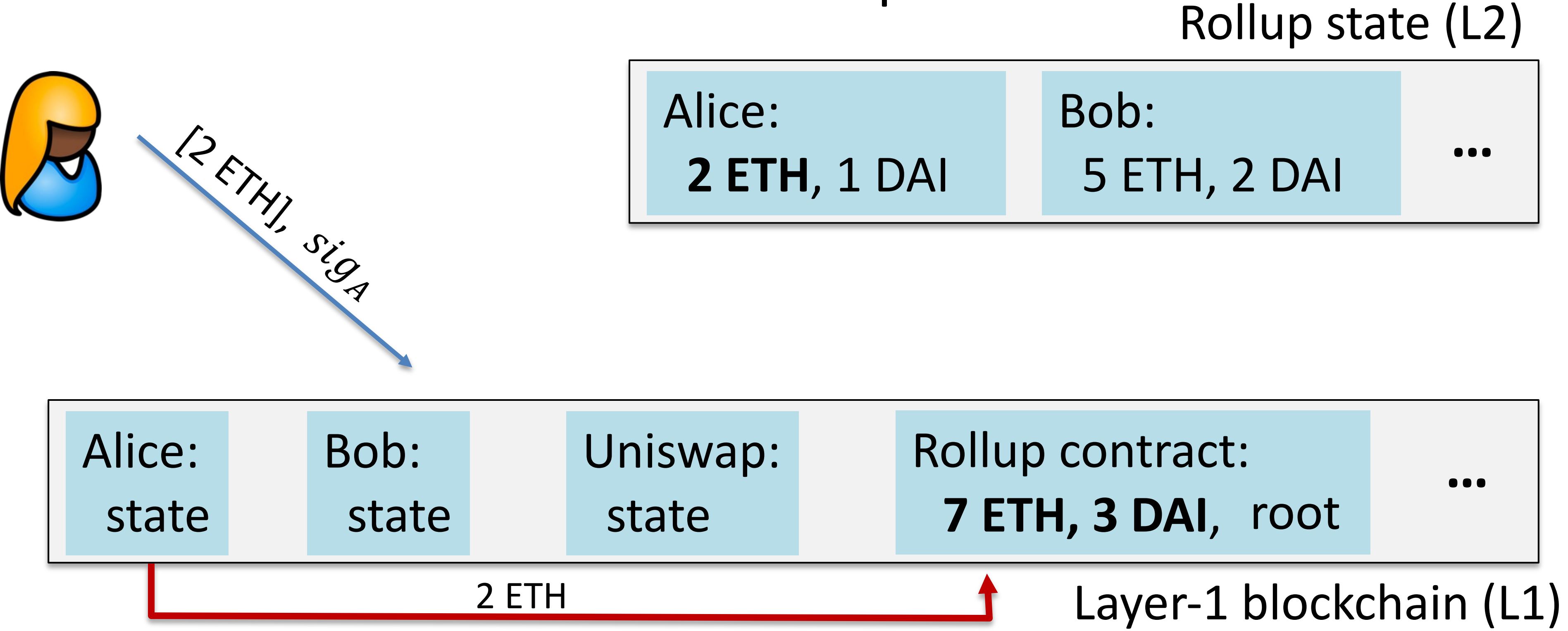
Transfers inside Rollup are easy (L2 → L2)

Coordinator updates root on Rollup contract



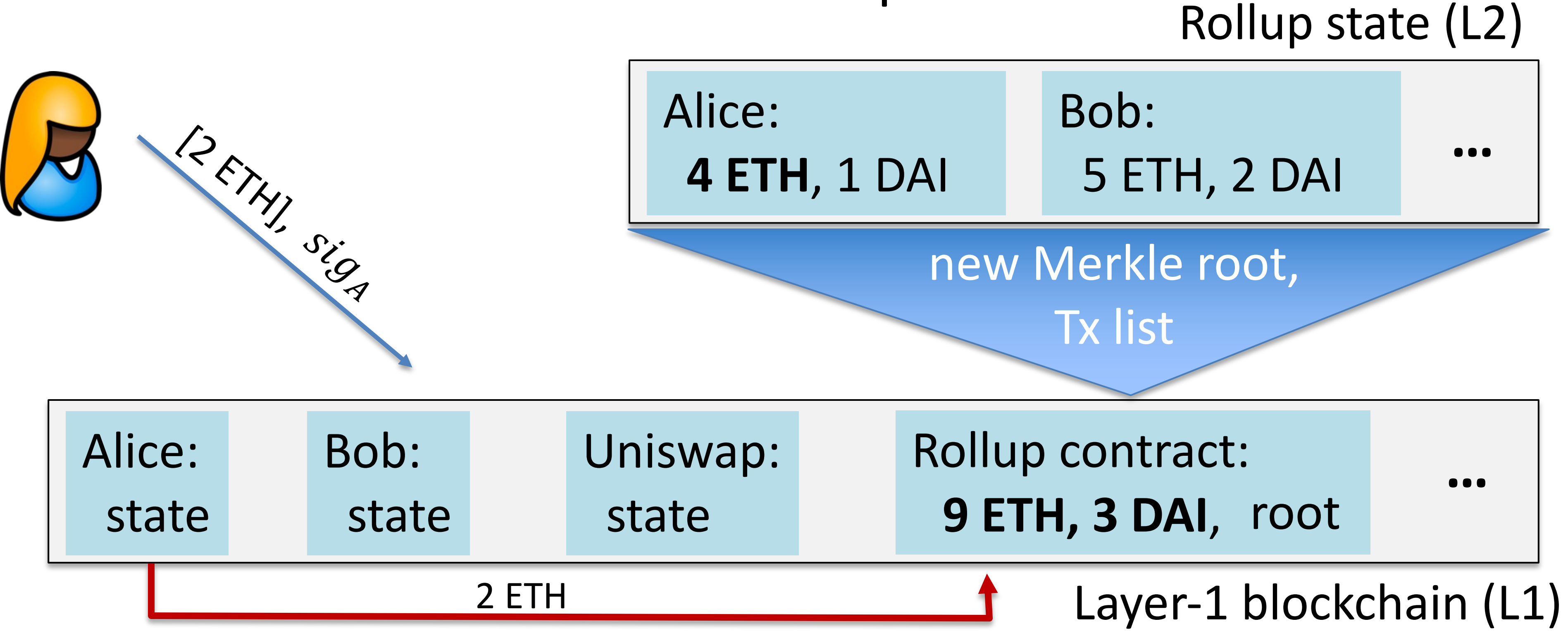
Transferring funds into Rollup (L1 → L2)

Alice issues an L1 Tx: slow and expensive



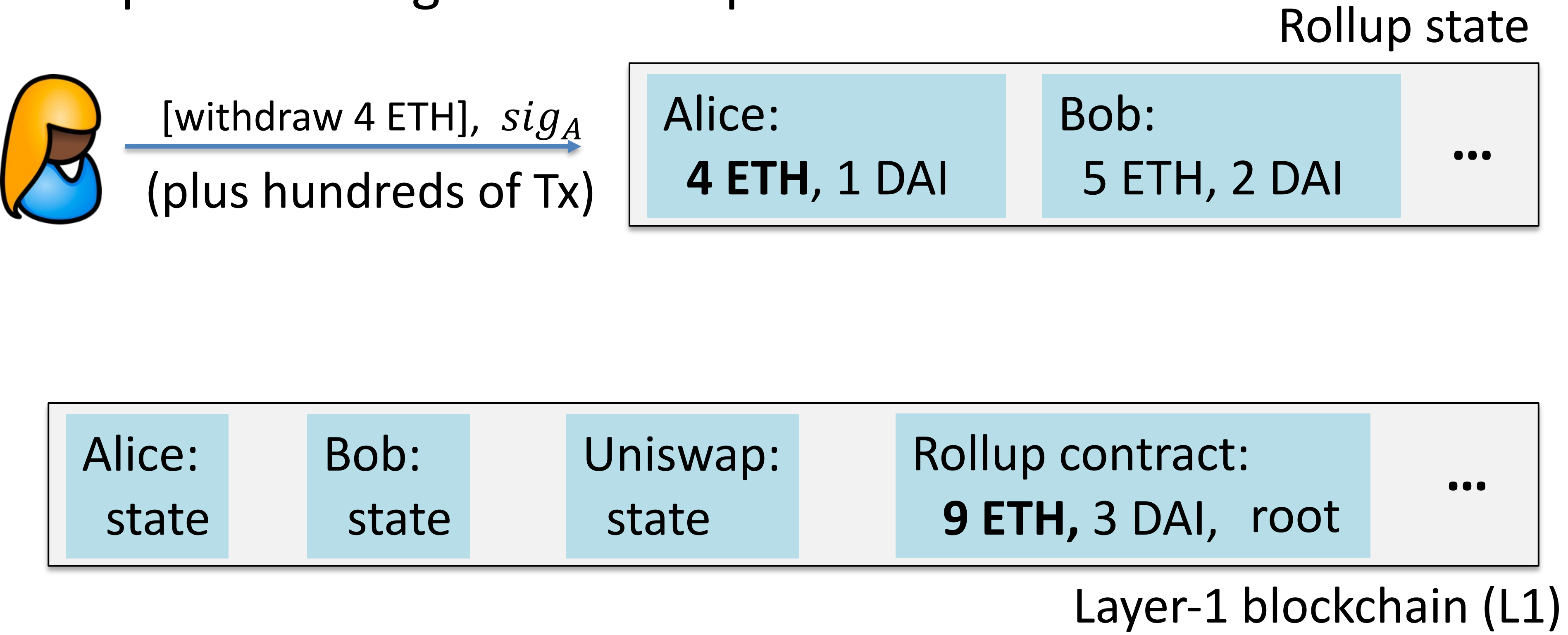
Transferring funds into Rollup (L1 → L2)

Alice issues an L1 Tx: slow and expensive



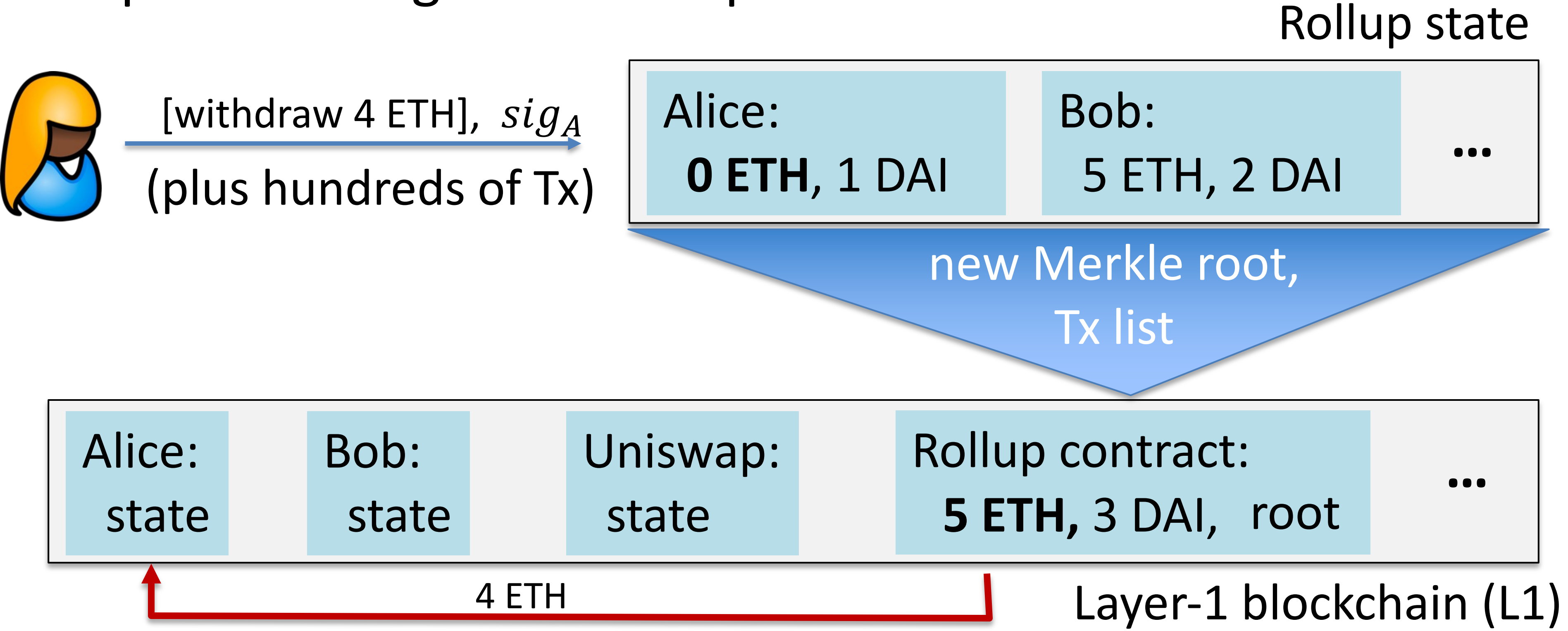
Transferring funds out of Rollup (L2 → L1)

Requires extra gas on L1 to process transfer



Transferring funds out of Rollup (L2 → L1)

Requires extra gas on L1 to process transfer



Summary: transferring Rollup assets

Transactions within a Rollup are easy:

- Batch settlement on L1 network (e.g., Ethereum)

Moving funds into or out of Rollup system (L1 $\Leftrightarrow$ L2) is expensive:

- Requires posting more data on L1 network $\Longrightarrow$ higher Tx fees.

Moving funds from one Rollup system to another (L2 $\Leftrightarrow$ L2)

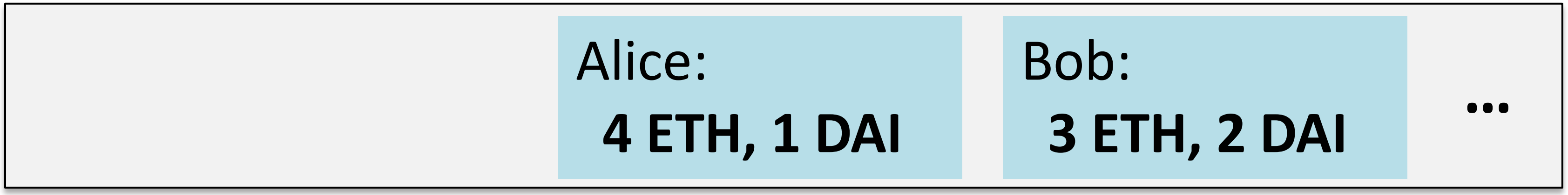
- Either via L1 network (expensive),

  or via a direct L2 $\Leftrightarrow$ L2 bridge (cheap)

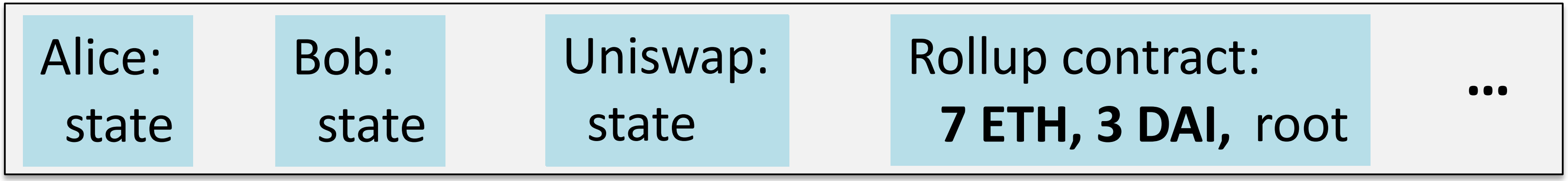
Running contracts on a Rollup?

Two copies of Uniswap

Rollup state (L2)

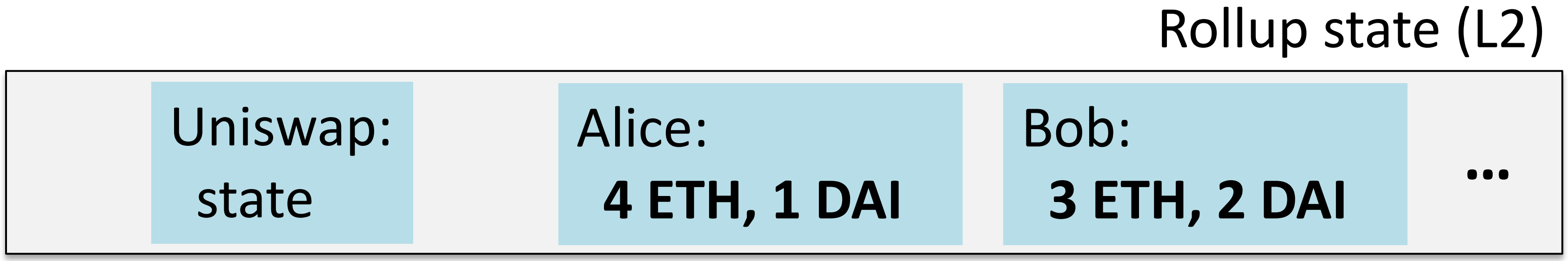


⇒ Rollup users can cheaply interact with Uniswap on Rollup



Layer-1 blockchain (L1)

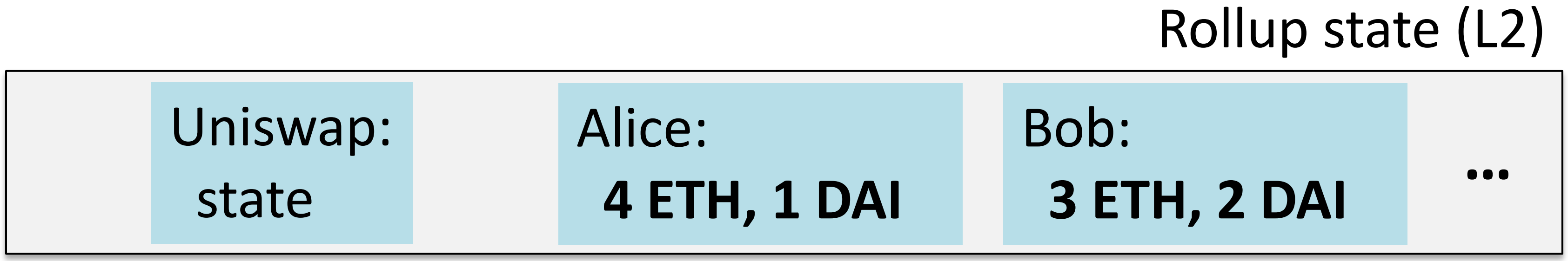
Running contracts on a Rollup?



Coordinator maintains state of all contracts on Rollup system:

- It updates the Uniswap Merkle leaf after every Tx to Uniswap
- Writes updated Rollup state Merkle root to L1 chain

Running contracts on a Rollup?



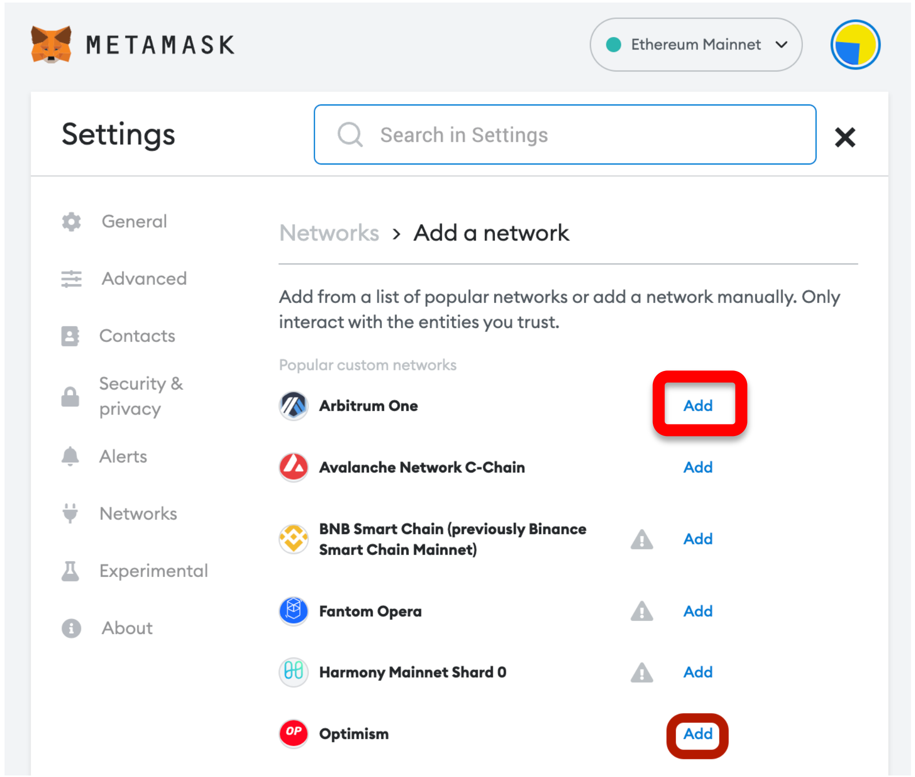
Rollup functions as Ethereum, but …

- It relies on the L1 chain to attest to the current Rollup state

How to send Tx to the coordinator

Enduser configures its wallet to send Tx to the RPC points of the selected Rollup.

(by default Metamask sends Tx to the Ethereum Mainnet RPC points)



METAMASK

Ethereum Mainnet

Settings

Search in Settings

General
Advanced
Contacts
Security & privacy
Alerts
Networks
Experimental
About

Networks › Add a network

Add from a list of popular networks or add a network manually. Only interact with the entities you trust.

Popular custom networks

Arbitrum One	Add
Avalanche Network C-Chain	Add
BNB Smart Chain (previously Binance Smart Chain Mainnet)	Add
Fantom Opera	Add
Harmony Mainnet Shard 0	Add
Optimism	Add

The role of the Coordinator

The Coordinator has multiple tasks:

- **Sequence** incoming Tx from Rollup users into a stream of Tx
  - ⇒ can extract MEV from searchers, in addition to Tx fees
  - ⇒ very profitable to be a Rollup coordinator
- **Execute** the stream of Tx on the latest Rollup state
- **Push updates** to the L1 chain

Shared coordinator: one coordinator for multiple Rollups (not today)

Coordinator architectures

A **centralized coordinator**:

- availability and censorship concerns,

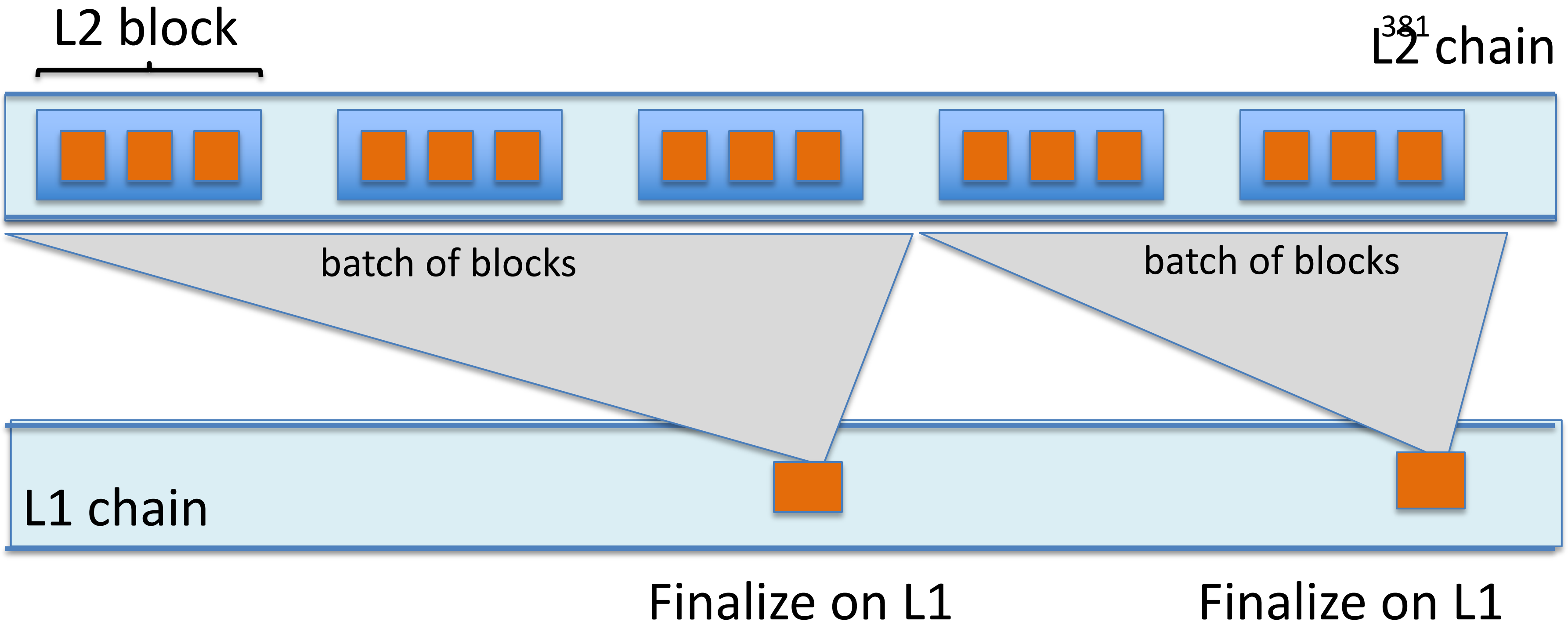
  … but cannot steal assets (as we will see)

A **decentralized coordinator**:

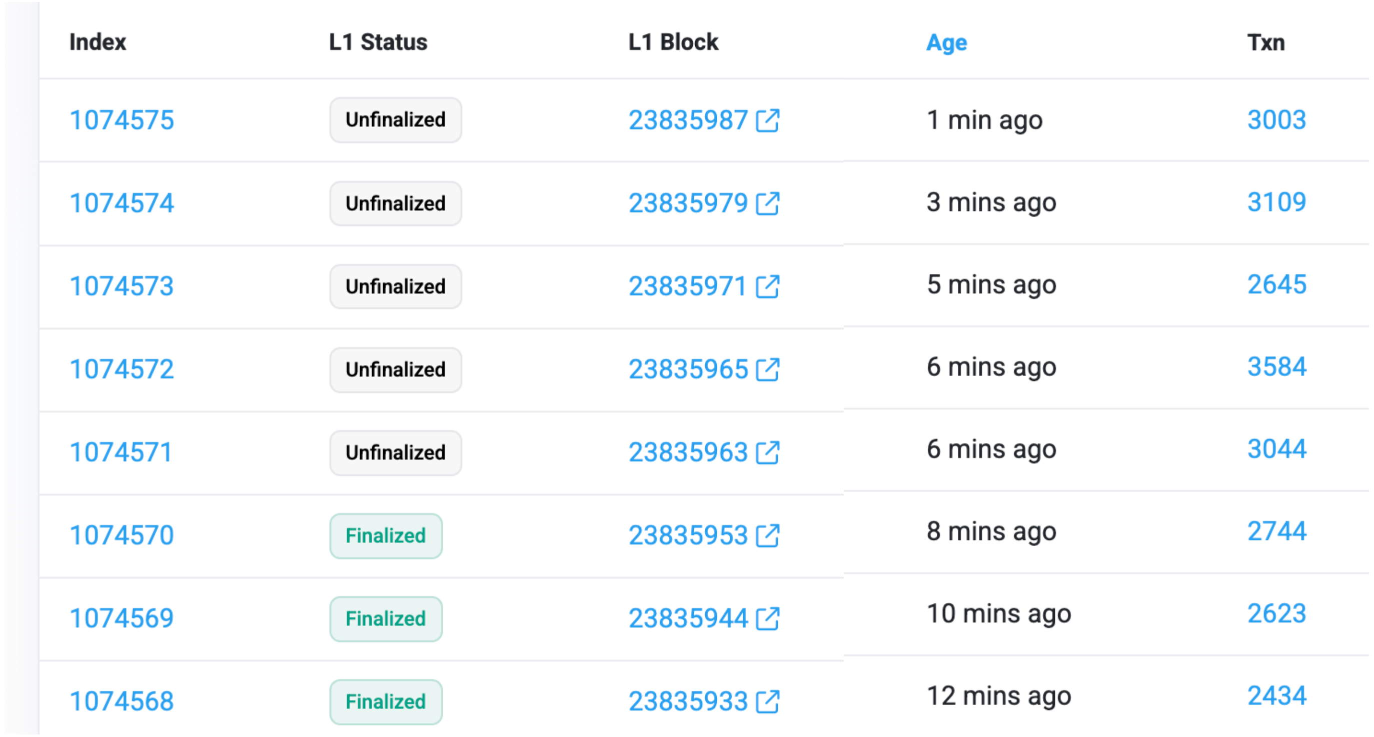
- a set of parties that run a fast consensus protocol
- At every epoch one party is chosen to sequence, execute, and push updates to the L1
- [Based Rollup (e.g., Tyco): L2 sequencing is done by L1 validators]

Importantly: L1 provides ground truth of the Rollup state

Tx rate on L2 is higher than on L1



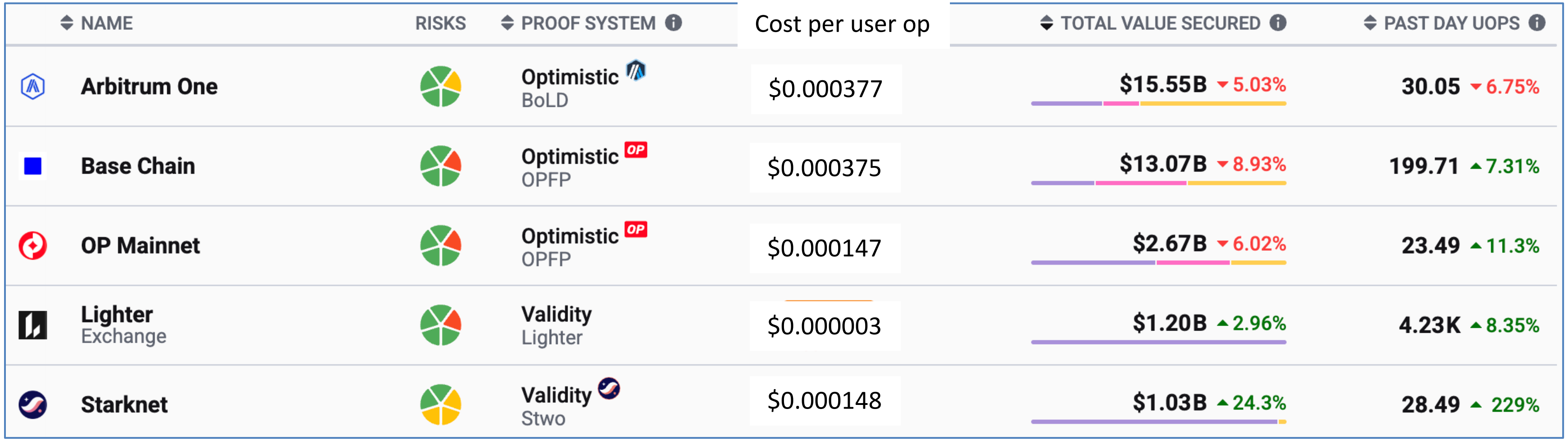
An example (Arbitrum – an optimisitic Rollup)



Index	L1 Status	L1 Block	Age	Txn	#blocks in batch
1074575	Unfinalized	23835987	1 min ago	3003	381
1074574	Unfinalized	23835979	3 mins ago	3109	416
1074573	Unfinalized	23835971	5 mins ago	2645	440
1074572	Unfinalized	23835965	6 mins ago	3584	110
1074571	Unfinalized	23835963	6 mins ago	3044	302
1074570	Finalized	23835953	8 mins ago	2744	492
1074569	Finalized	23835944	10 mins ago	2623	545
1074568	Finalized	23835933	12 mins ago	2434	581

A batch of blocks is posted on L1 (Ethereum) about every two minutes

Volume on some L2 systems (Nov. 2025)



NAME	RISKS	PROOF SYSTEM	Cost per user op	TOTAL VALUE SECURED	PAST DAY UOPS
Arbitrum One		Optimistic BoLD	$0.000377	$15.55B ▼5.03%	30.05 ▼6.75%
Base Chain		Optimistic OPFP	$0.000375	$13.07B ▼8.93%	199.71 ▲7.31%
OP Mainnet		Optimistic OPFP	$0.000147	$2.67B ▼6.02%	23.49 ▲11.3%
Lighter Exchange		Validity Lighter	$0.000003	$1.20B ▲2.96%	4.23K ▲8.35%
Starknet		Validity Stwo	$0.000148	$1.03B ▲24.3%	28.49 ▲ 229%

UOPS: User Operations Per Second

Ethereum cost per user op: 0.3 USD/Tx

Not so simple …

Problems …

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 users issue Tx?

  … can’t compute updated Rollup Merkle root.

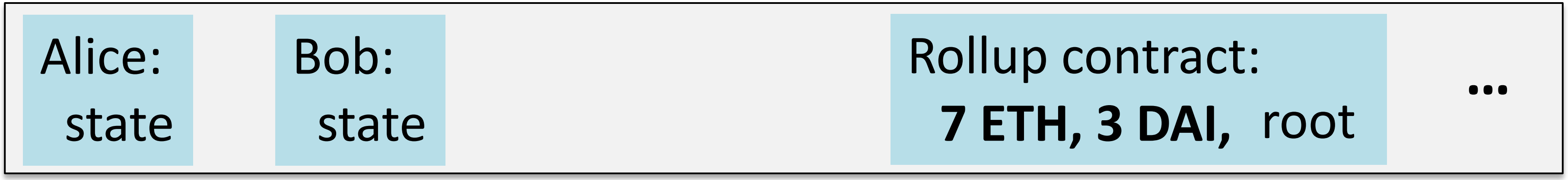
Problem 1: what if coordinator is dishonest?

Can coordinator steal funds from Rollup users?

No! L1 chain verifies that Rollup state updates are valid.

$\Rightarrow$ all Tx are valid and properly signed by the Rollup users

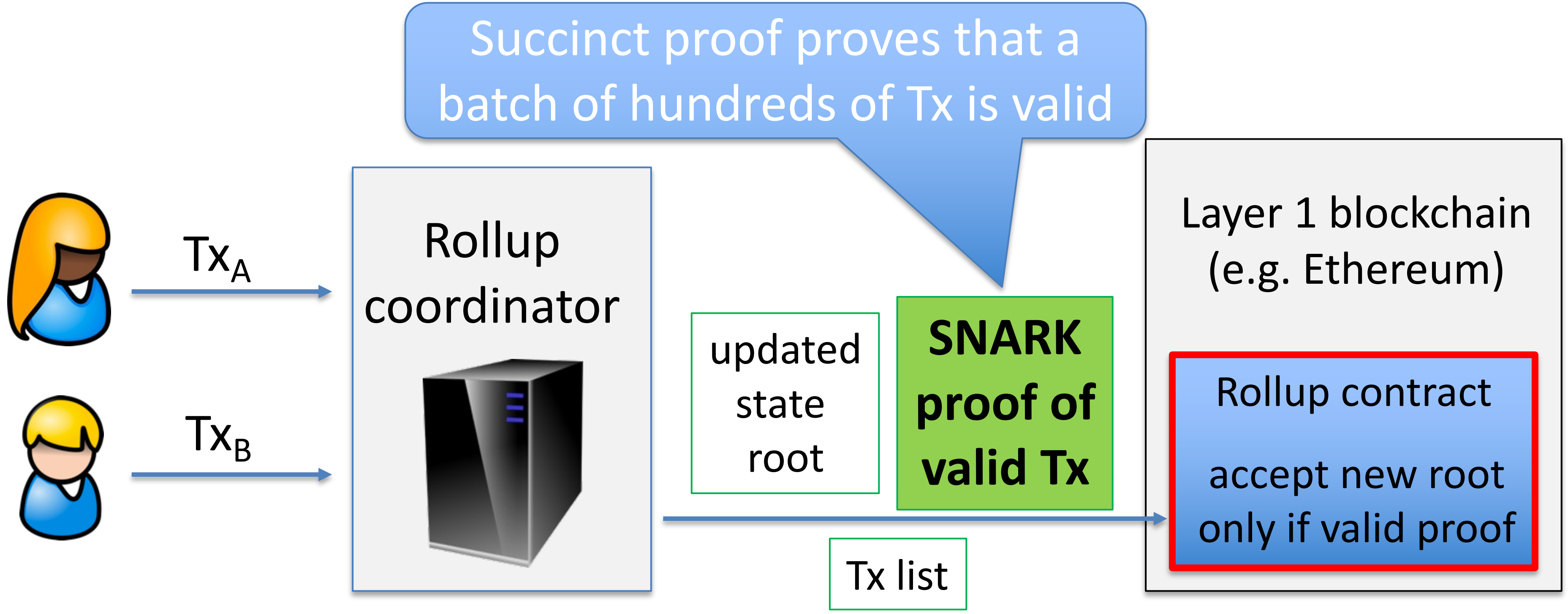
Challenge: how to do this cheaply ?? (with little gas on L1)



Layer-1 blockchain (L1)

Verifying Rollup state updates

Approach 1: **validity proofs** (called a **zk-Rollup**)


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)

(usually not a zero knowledge proof)

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

zkEVM

When a contract (e.g. Uniswap) runs on a Rollup:

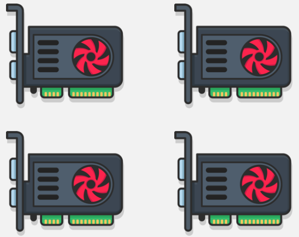
- coordinator builds a SNARK proof of correct execution of an EVM program ⇒ called a **zkEVM**
- Generating proof is a heavyweight computation

  … verifying proof is fast

Flavors of EVM block execution proofs:

- Directly prove that EVM bytecode ran correctly
  (Polygon zkEVM, Scroll)
- Compile Reth to RISC-V, prove in a RISC-V zkVM
  (MatterLabs, Succint, Zisk, Risc0, …)

Rollup coordinator



(lots of GPUs)

The end result

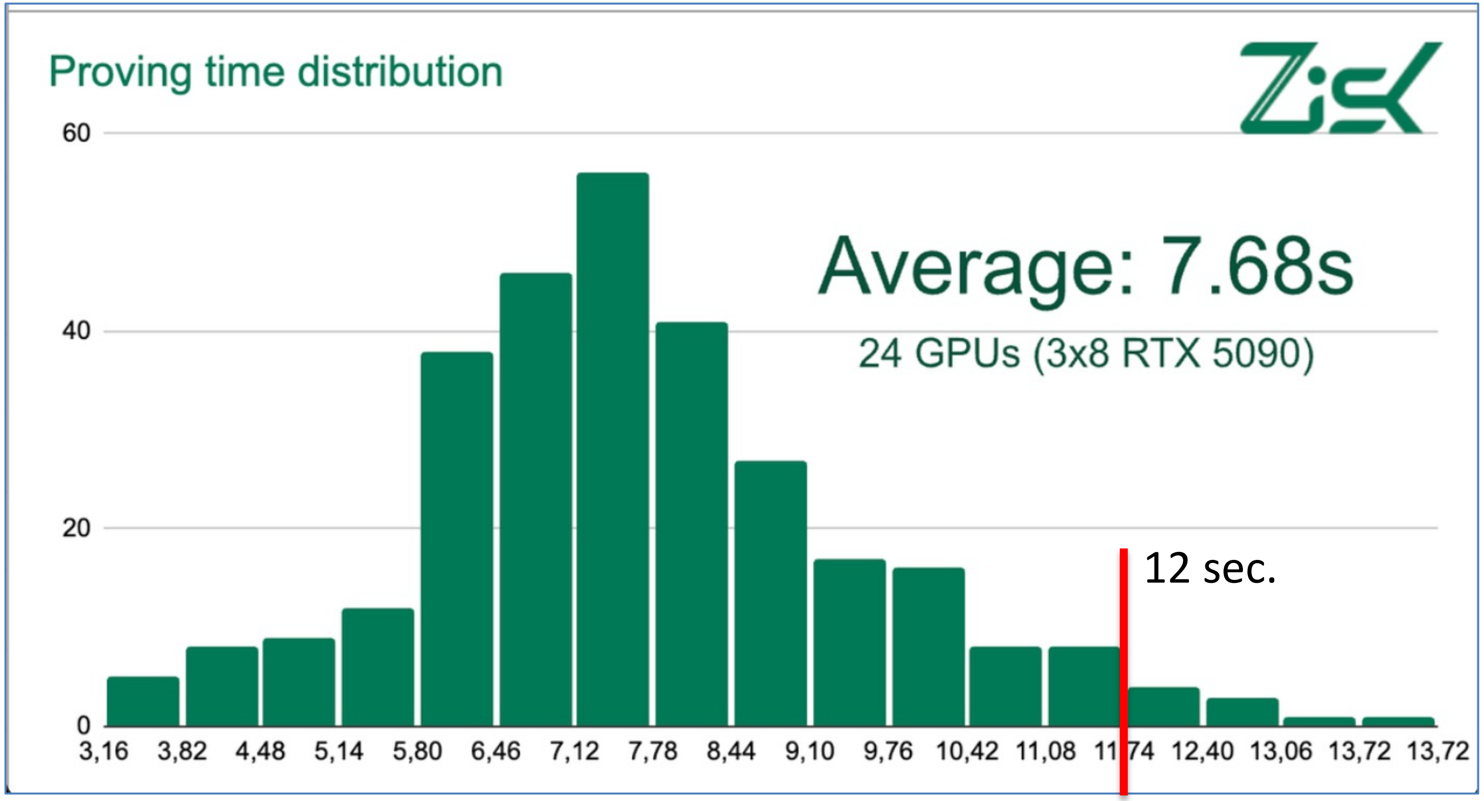
Rollup contract on L1 ensures coordinator cannot cheat:

- 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

⇒ Anyone can act as a coordinator (with enough compute power)

Generating a proof in under 12 seconds!

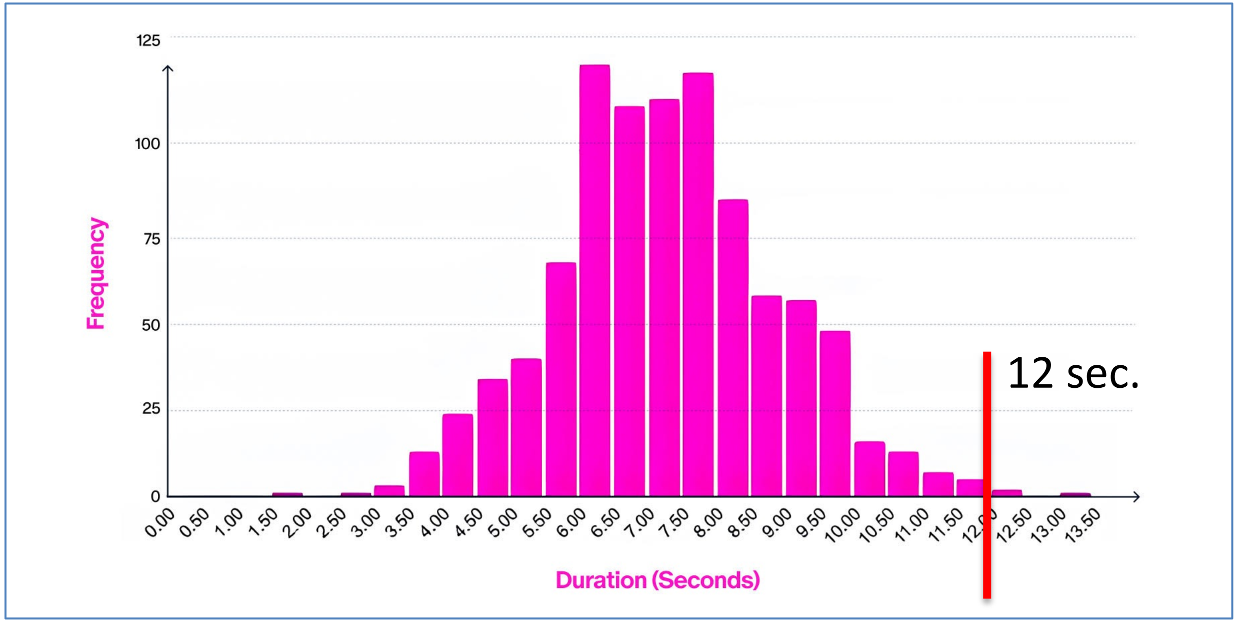


Distribution of the time to generate a proof for a block.

Oct. 28, 2025

Generating a proof in under 12 seconds!

SP1 Hypercube with only 16 GPUs



Distribution of the time to generate a proof for a block

Nov. 18, 2025

Verifying Rollup state updates

Approach 2: **fault proofs** (called an **optimistic Rollup**)

- Operation: Coordinator submits state updates to L1 w/o a proof
- If update is invalid: anyone has seven days to submit a fault proof
  - Successful fault proof reverts the update (and rewards validator)
  - Unsuccessful fault proof costs complainer a fee

Challenge: how to prove a fault to the Rollup contract on L1 ??

Naively: L1 can re-execute all Tx in batch $\Rightarrow$ expensive and slow

Fault Proof game

coordinator

claimed post-root

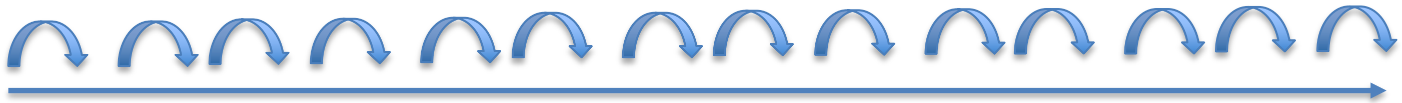
pre-root

Tx list

Coordinator computes Merkle tree of all states.
Sends Merkle root to L1

fault claim

different post-root



break computation into small steps

Fault Proof game

coordinator

claimed $state_n$

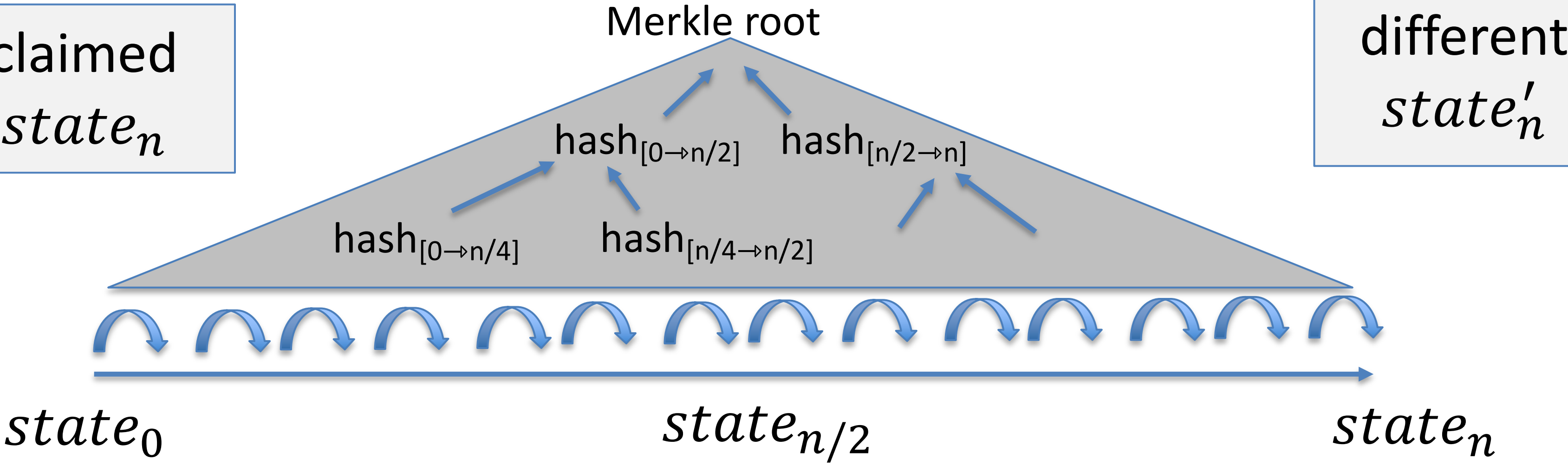
pre-root

Tx list

we know $state_n \neq state'_n$

fault claim

different $state'_n$



Fault Proof game: binary search

coordinator

claimed $state_n$

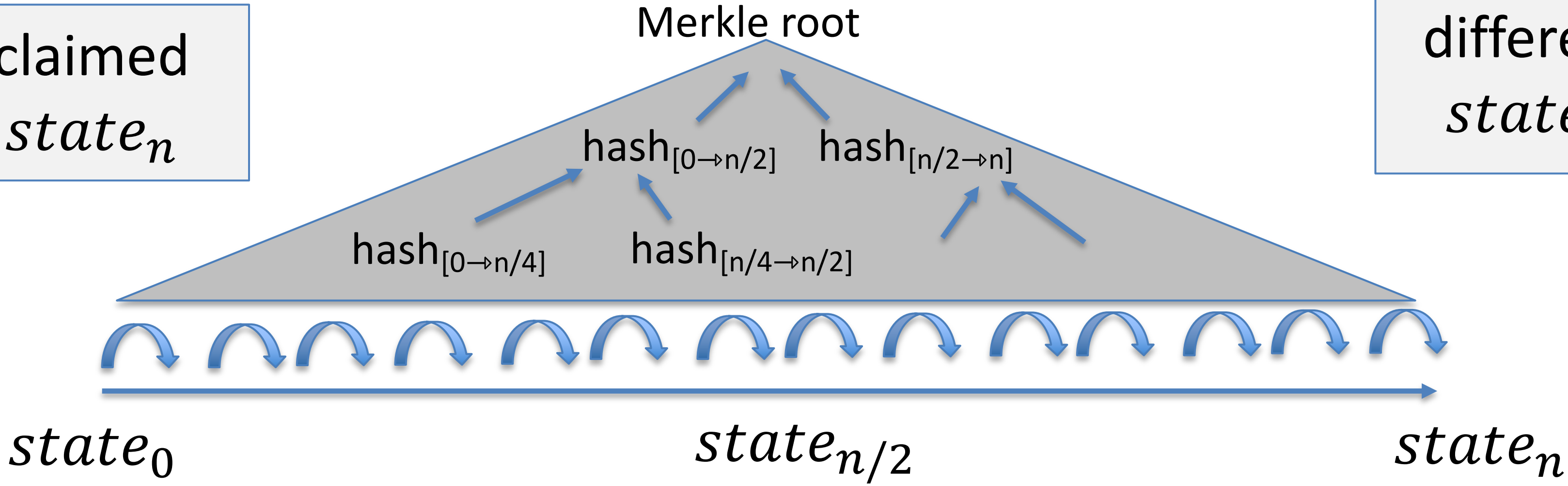
pre-root

Tx list

Suppose $state_{n/2} \neq state'_{n/2}$

fault claim

different $state'_n$



Merkle root
$\text{hash}_{[0 \to n/2]}$
$\text{hash}_{[n/2 \to n]}$
$\text{hash}_{[0 \to n/4]}$
$\text{hash}_{[n/4 \to n/2]}$
$state_0$
$state_{n/2}$
$state_n$

Fault Proof game: binary search

coordinator

claimed
$state_n$

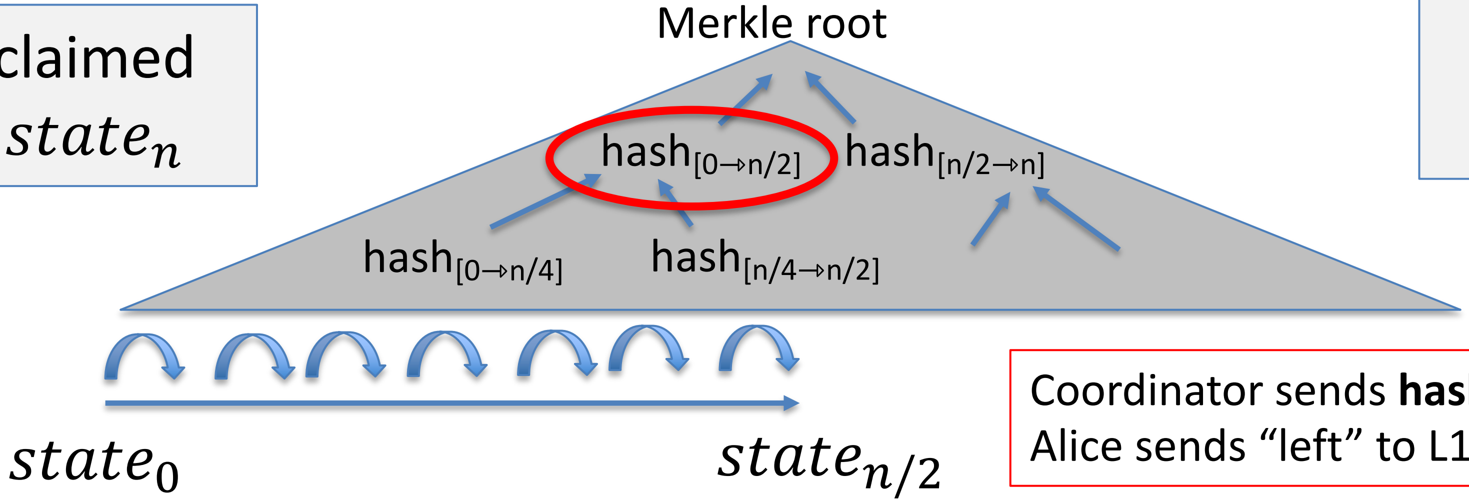
pre-root

Tx list

Suppose $state_{n/2} \neq state'_{n/2}$

fault claim

different
$state'_n$



Coordinator sends **hash$_{[0\rightarrow n/2]}$** to L1
Alice sends “left” to L1

Fault Proof game: binary search

coordinator

claimed
$state_n$

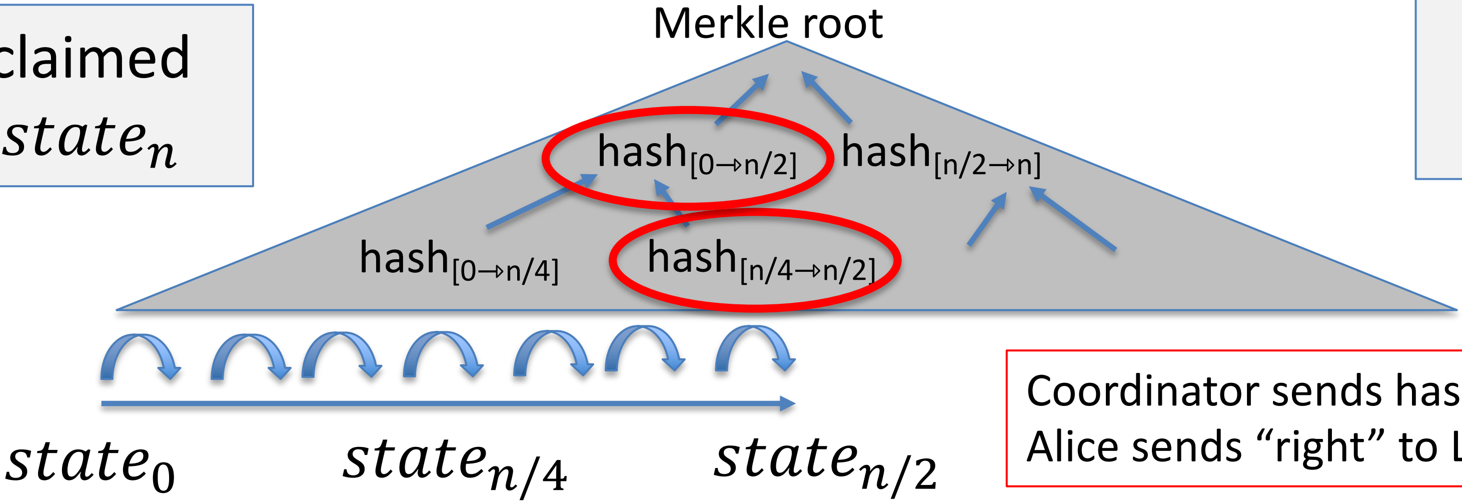
pre-root

Tx list

Suppose $state_{n/4} = state'_{n/4}$

fault claim

different
$state'_n$



Coordinator sends $\text{hash}_{[n/4\to n/2]}$ to L1
Alice sends "right" to L1

Fault Proof game: binary search

coordinator

claimed $state_n$

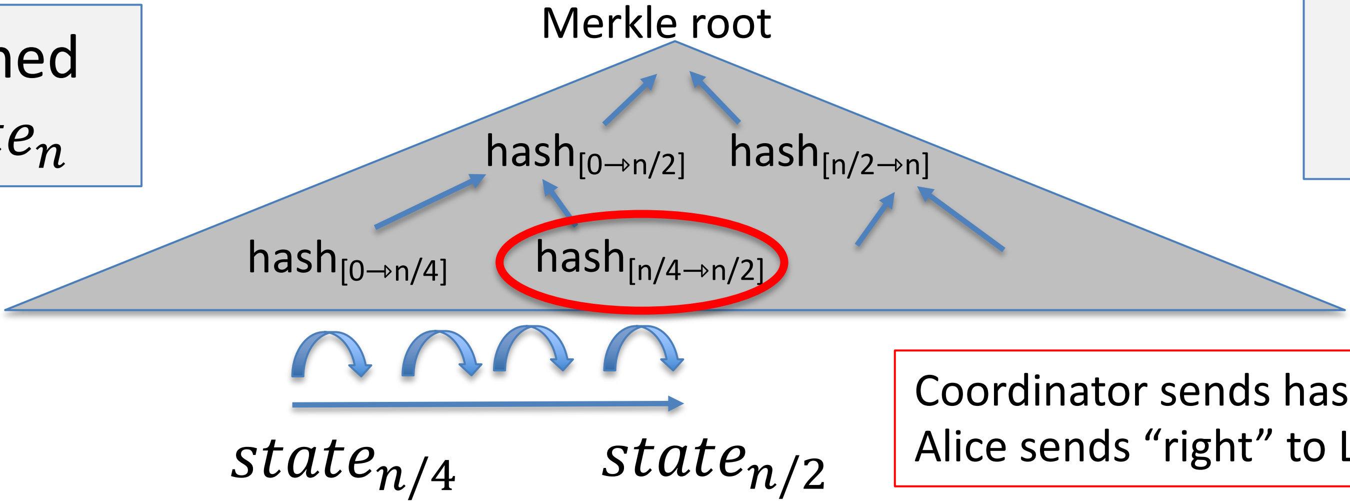
pre-root

Tx list

Suppose $state_{n/4} = state'_{n/4}$

fault claim

different $state'_n$



Merkle root
$\text{hash}_{[0\to n/2]}$
$\text{hash}_{[n/2\to n]}$
$\text{hash}_{[0\to n/4]}$
$\text{hash}_{[n/4\to n/2]}$
$state_{n/4}$
$state_{n/2}$

Coordinator sends $\text{hash}_{[n/4\to n/2]}$ to L1
Alice sends “right” to L1

Fault Proof game: binary search

coordinator

pre-root

Tx list

After $\log_2 n$ rounds:

- L1 has $state_i$ and $state_{i+1}$ from coordinator
- $state_i = state_i'$ and $state_{i+1} \neq state_{i+1}'$

or game times out because one player defects

$\Rightarrow$ Now L1 can verify fault proof by checking **one** computation step!

A simpler alternative

claimed $state_n$

pre-root

Tx list

different $state'_n$

- Alice submits to L1 contract a SNARK proof that $state_n$ is invalid
- L1 verifies SNARK, and if valid, slashes coordinator

$\Rightarrow$ SNARK is only needed in a rare fault event

Some difficulties with optimistic approach

(1) Transactions only settle after 7 days (after fault window expires)

- Alice needs to wait 7 days to withdraw funds from Rollup

  (Rollup contract will only send her the funds after 7 days)

[For fungible tokens, a 3rd party can advance the funds to Alice after checking validity of Alice's withdraw Tx. Does not apply to non-fungible tokens.]

(2) Suppose a successful fault proof 4 days after batch is posted

$\Rightarrow$ all subsequent Tx need to be reprocessed

The end result

Can easily port any smart contract to an optimistic Rollup

- The Rollup EVM can be enhanced with new features (opcodes)

High Tx throughput: in principle, up to 4000 tx/s

- No need for special hardware at the coordinator

Anyone can act as a coordinator and a fault verifier

Downside: 7 day finality delay

… ok, so coordinator cannot submit invalid Tx.

Problem 2: centralized coordinator, what if it stops providing service?

(censorship)

Solution: setup a new coordinator

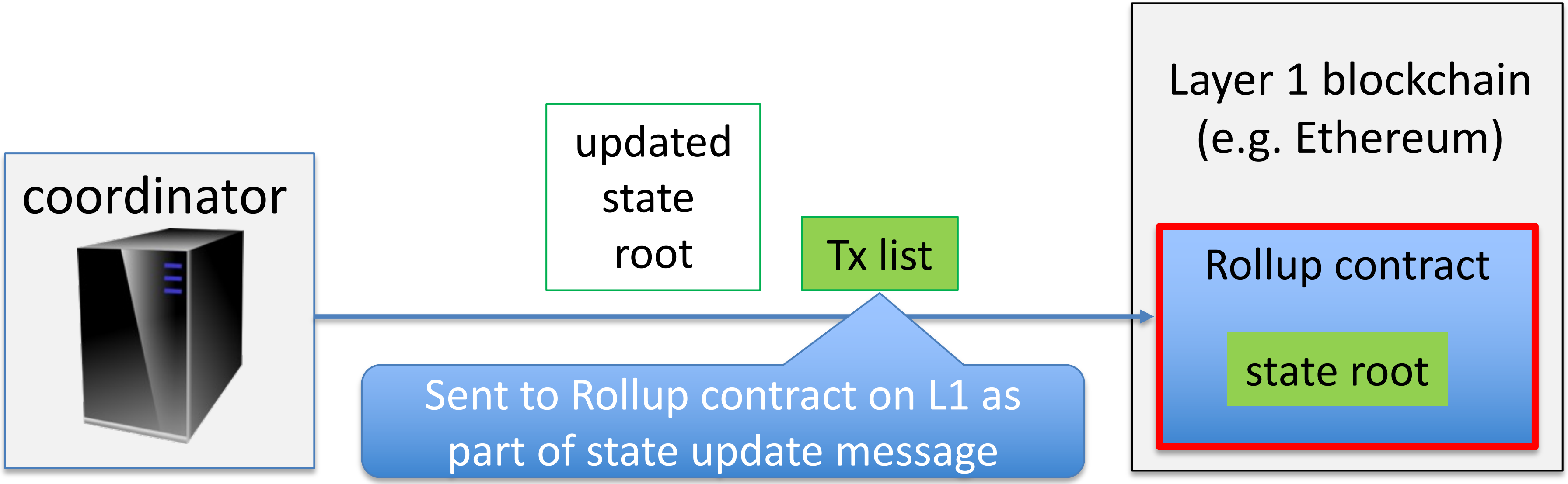
… but need the latest Rollup state

Where to get state?? The **data availability problem**

Ensuring Rollup state is always available

The definition of a Rollup:

Rollup state can always be reconstructed from data on the L1 chain


coordinator
updated state root
Tx list
Sent to Rollup contract on L1 as part of state update message
Layer 1 blockchain (e.g. Ethereum)
Rollup contract
state root

Ensuring Rollup state is always available

To reconstruct current Rollup state:

- Read all Rollup update messages and re-execute Tx.

  $\Rightarrow$ anyone can become a coordinator

- Rollups use L1 for data storage

What to store?

- For zk-Rollup: send Tx summary to L1, without user signatures

  (SNARK proof proves validity of signatures)

- For optimistic: need to send Tx summary *and* signatures to L1

Ensuring Rollup state is always available

The downside: **expensive**

- Tx list is sent as calldata: 16 gas per non-zero byte

Ethereum's solution: **blobs** [Fusaka upgrade, Dec. 3rd, 2025]

- Every block can contain up to 21 blobs (128 KB each)
- Validators store blobs for 30 days (sharded using PeerDAS)
- Blobs cost far less gas than calldata of the same size
  - Price determined by auction among Rollups (like EIP-1559)

What not store Tx data off chain?

Data Availability Committee (DAC)

To reduce fees due to Tx storage cost on L1:

- **Store L2 state root** (small) on the L1 chain
- **Store Tx data** (large) with a Data Availability Committee (**DAC**):
    - a set of nodes trusted to keep the data available
    - cheaper than storage on L1
    - L1 accepts an update only if <u>all</u> DAC members sign it

        $\Rightarrow$ ensures that all DAC members accepted Tx data

Setting up a new coordinator depends on availability of the DAC

Validium

Validium: an L2 using a DAC and validity proofs (SNARKs)

- Well suited for lower value assets.
- Potential privacy benefits … only DAC members see Tx data

An example: StarkEx uses a **<u>five</u>** member DAC

- Users can choose between Validium or Rollup modes

(Tx data off-L1-chain vs. Tx data on-L1-chain)

cheaper Tx fees, but only secure as DAC	More expensive Tx, but same as L1 security

Summary: types of L2

Scaling the blockchain: Payment channels and Rollups (L2 scaling)

security →	**SNARK validity proofs**	**Fraud proofs**
Tx data on L1 chain	**zkRollup**	optimistic Rollup, 7 day finality
Tx data in a DAC	Validium (reduced fees, but higher risk)	”Plasma”

availability

Can coordinator censor a Tx?

What if coordinators refuse to process a Tx?

What to do? One option:

- enduser can post Tx directly to the L1 Rollup contract
- The L1 Rollup contract will then refuse to accept updates until an update includes that Tx

$\Rightarrow$ censorship will cause the entire Rollup to freeze

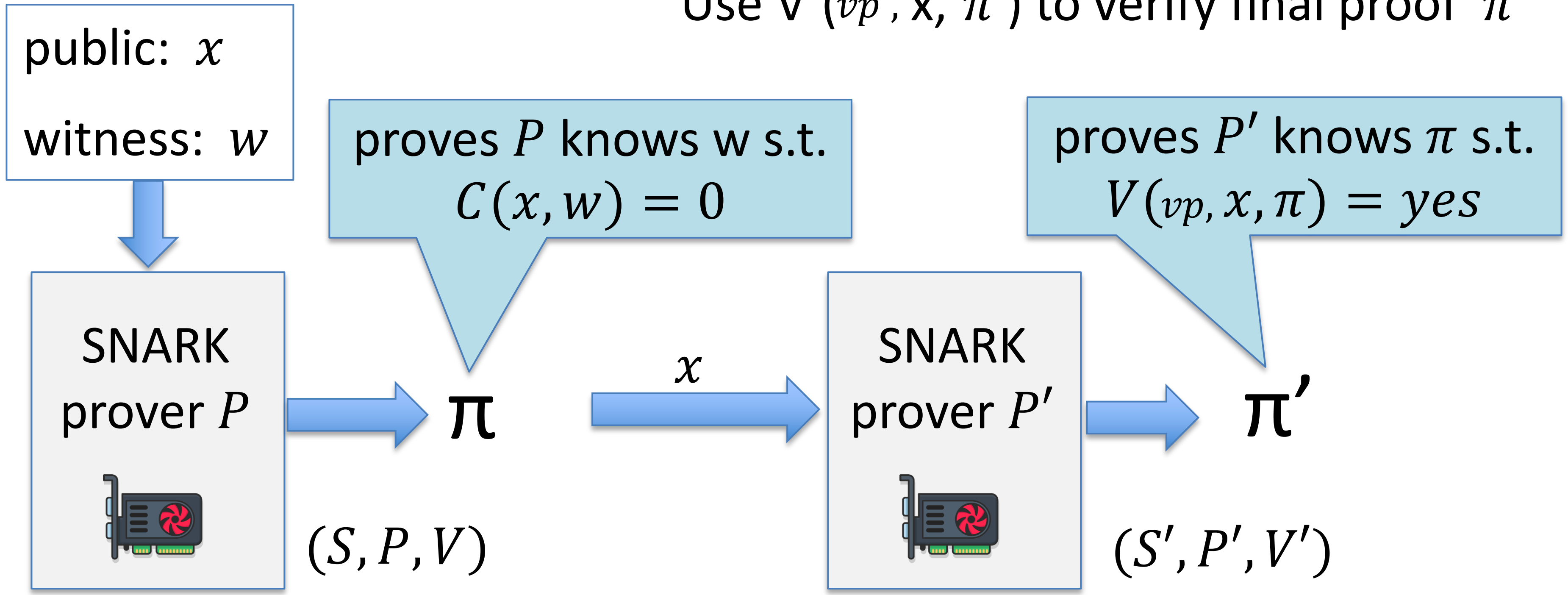
SNARK recursion

Layer 3 and beyond …

SNARK recursion

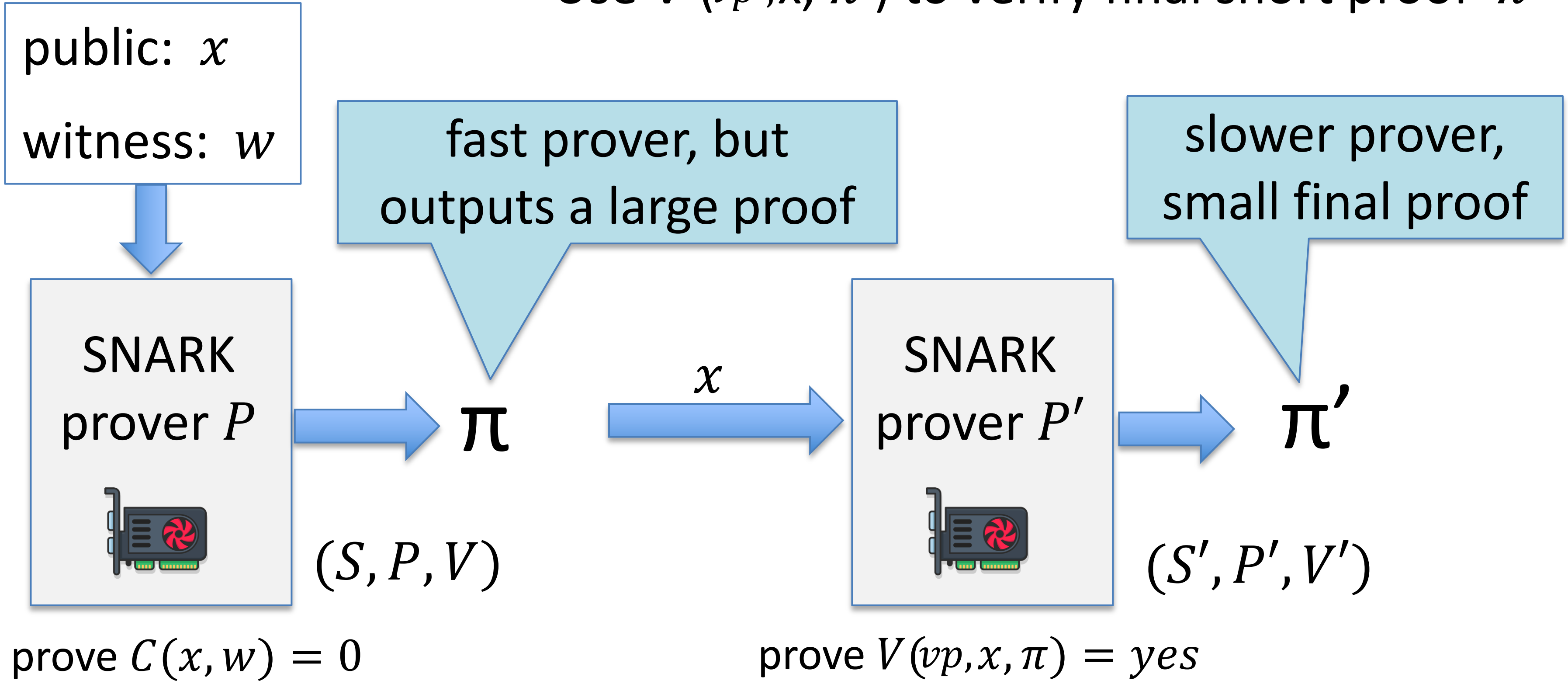
Two level recursion: $\boxed{\textbf{proving knowledge of a proof}}$

Use V'(vp', x, π') to verify final proof π'

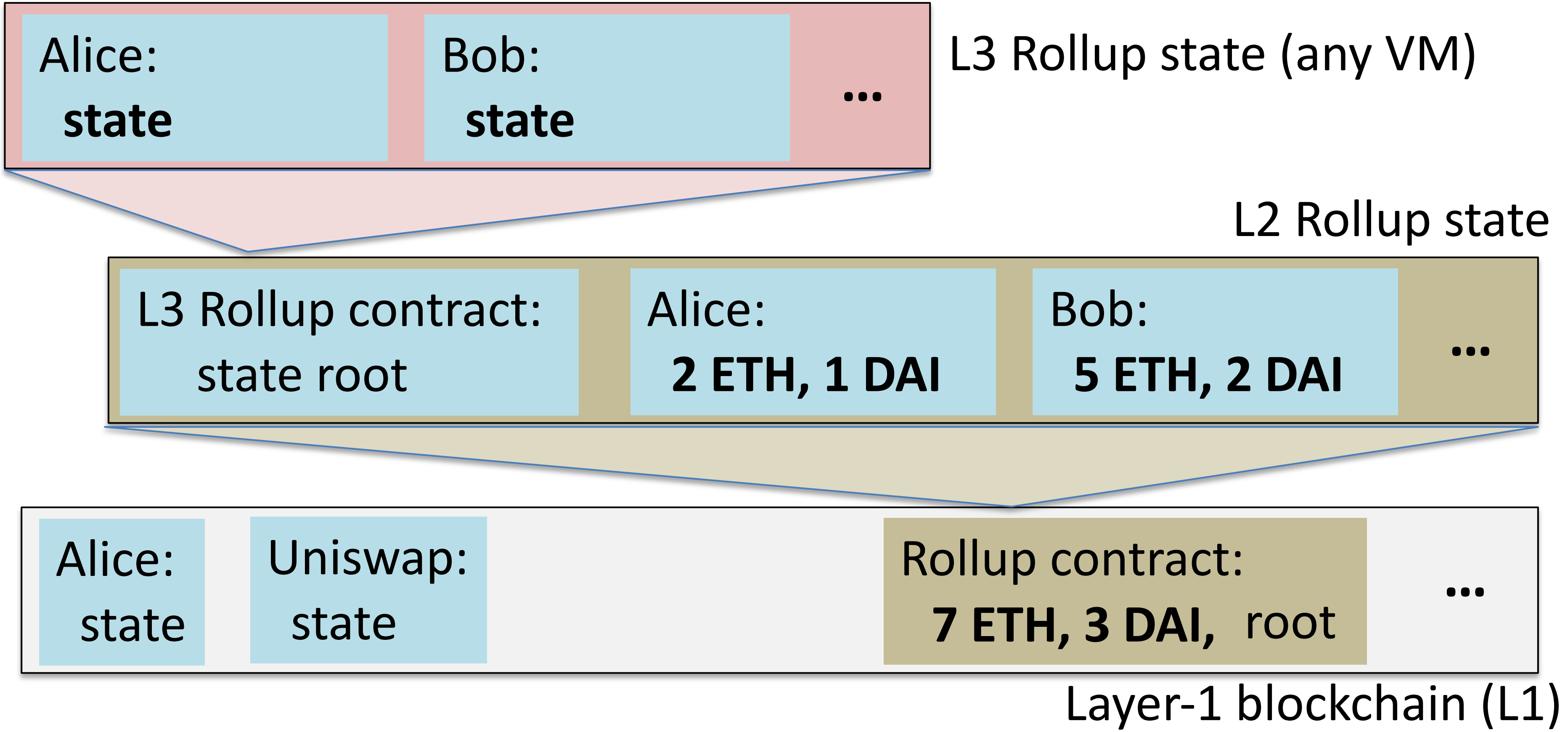

public: x
witness: w
proves P knows w s.t. $C(x,w) = 0$
proves P' knows π s.t. $V(vp,x,\pi) = yes$
SNARK prover P
π
x
SNARK prover P'
π'
(S,P,V)
(S',P',V')

Application 1: proof compression

Use V'(vp',x, π') to verify final short proof π'



Application 2: Layer three and beyond



Layer three and beyond

One L2 coordinator can support many L3s

- each L3 can run a custom VM with its own features
- L3 chains can communicate with each other through L2

Each L3 coordinator submits Tx list and SNARK proof to L2

- L2 coordinator: collects batch of proofs,
  - builds a proof π that it has a batch of valid proofs, and
  - submits the single proof π and updated root to L1 chain.

END OF LECTURE

Next lecture: final topics