CS251 Fall 2022

(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)**

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:

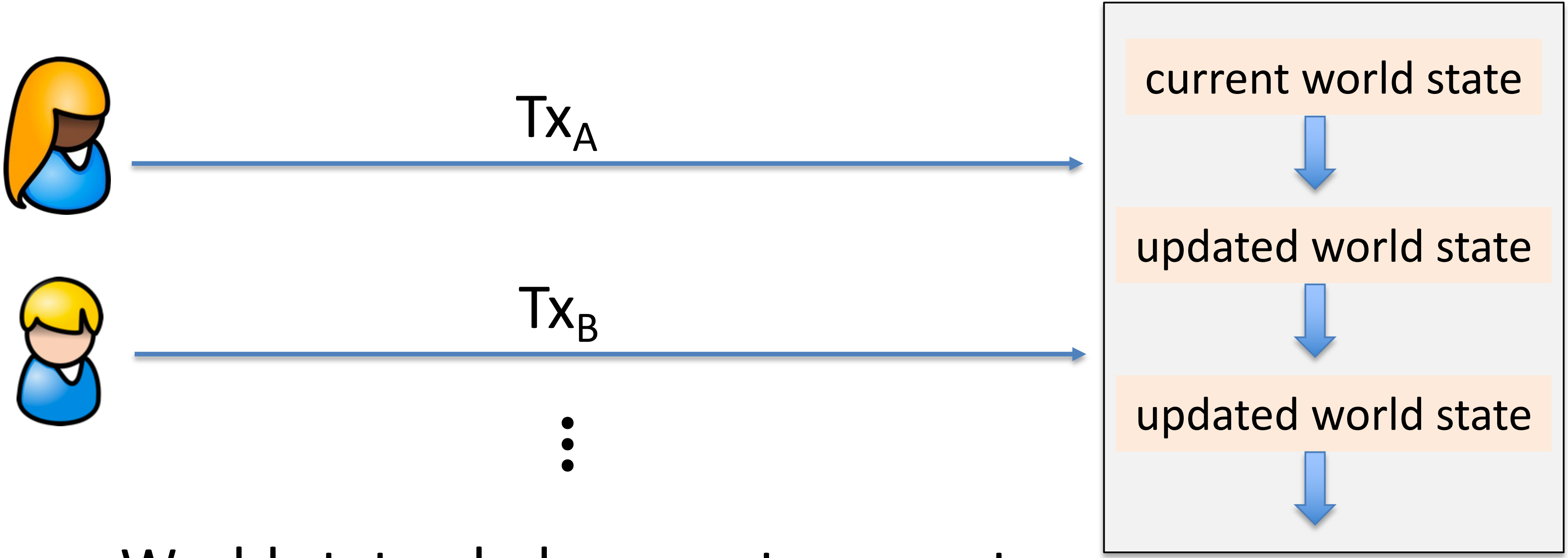
reduces composability

- 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 …

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



Tx_A

Tx_B

⋮

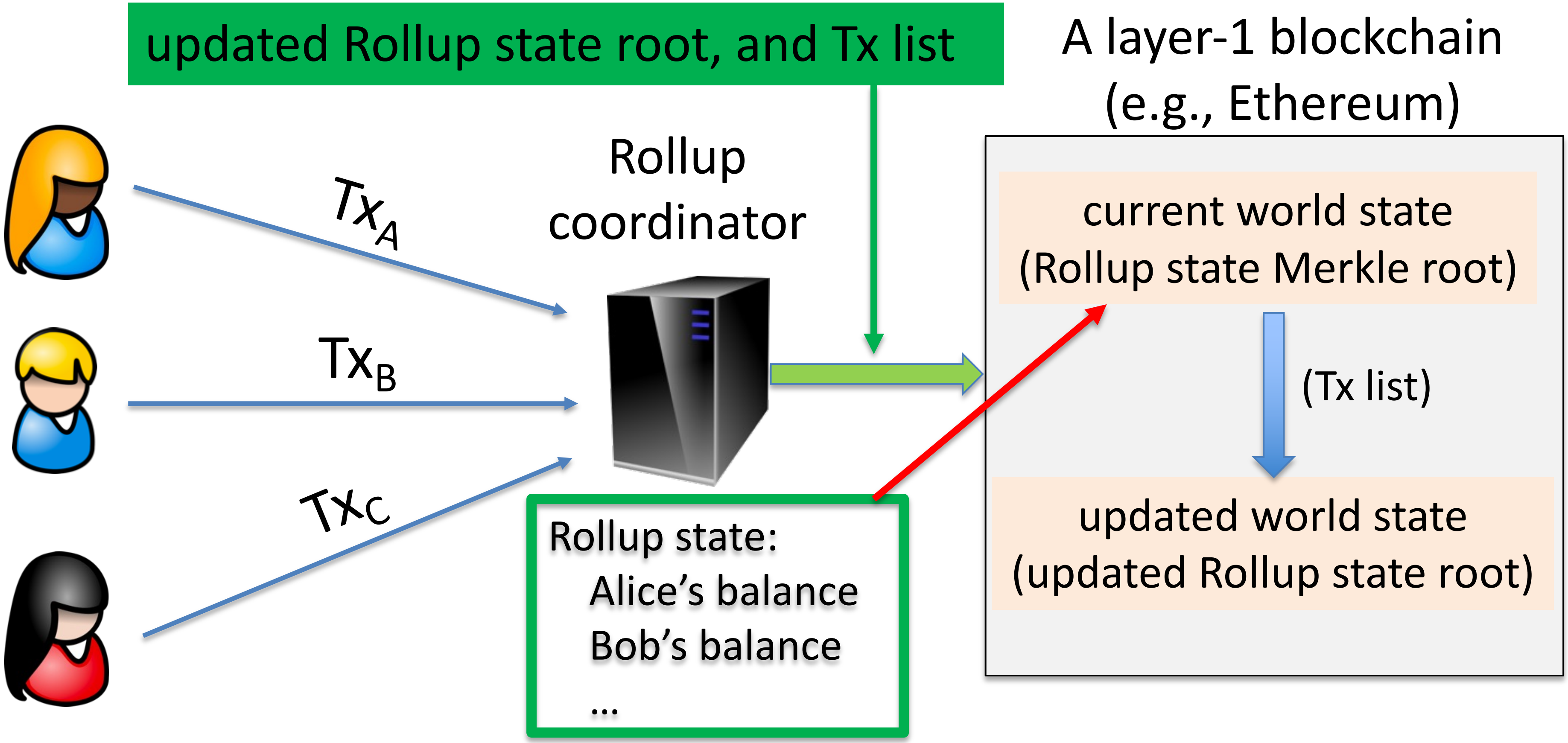
current world state

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)

Tx_A

Tx_B

Tx_C

Rollup
coordinator

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

current world state
(Rollup state Merkle root)

(Tx list)

updated world state
(updated Rollup state root)

Rollup idea 1: batch many Tx into one

Key point:

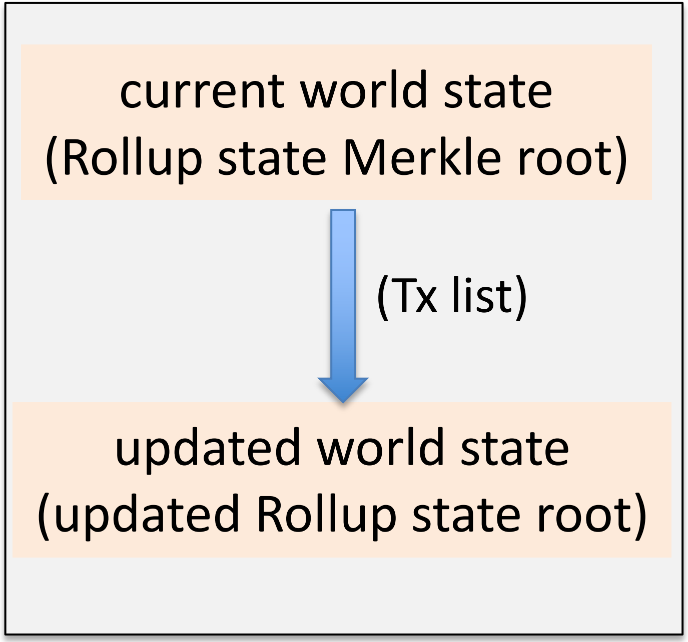
- *Hundreds* of transactions on Rollup state are batched into a *single* transaction on layer-1

  ⇒ 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)

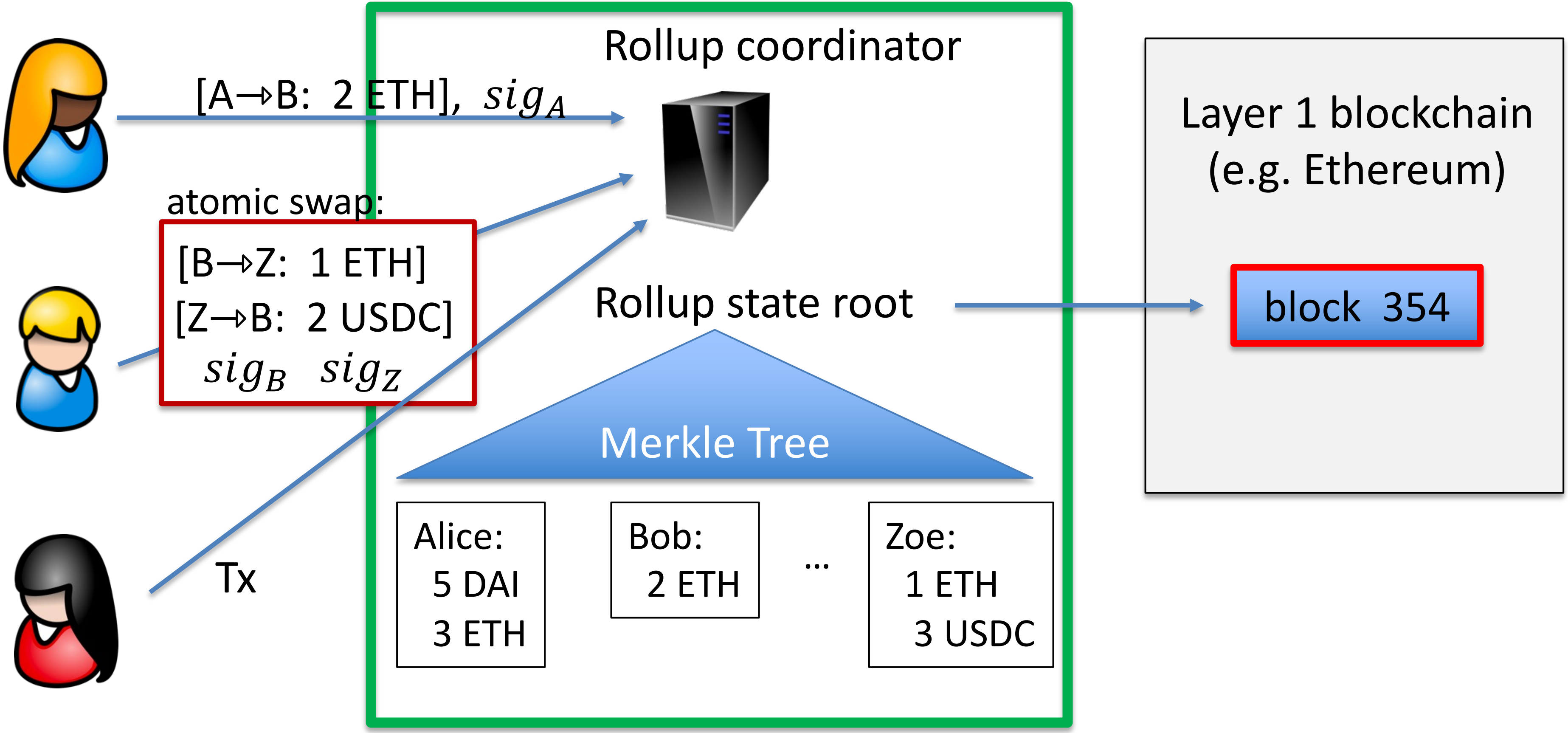


current world state
(Rollup state Merkle root)

(Tx list)

updated world state
(updated Rollup state root)

Rollup operation (simplified)



[A→B: 2 ETH], sig_A

atomic swap:

[B→Z: 1 ETH]

[Z→B: 2 USDC]

sig_B sig_Z

Tx

Rollup coordinator

Rollup state root

Merkle Tree

Alice:
5 DAI
3 ETH

Bob:
2 ETH

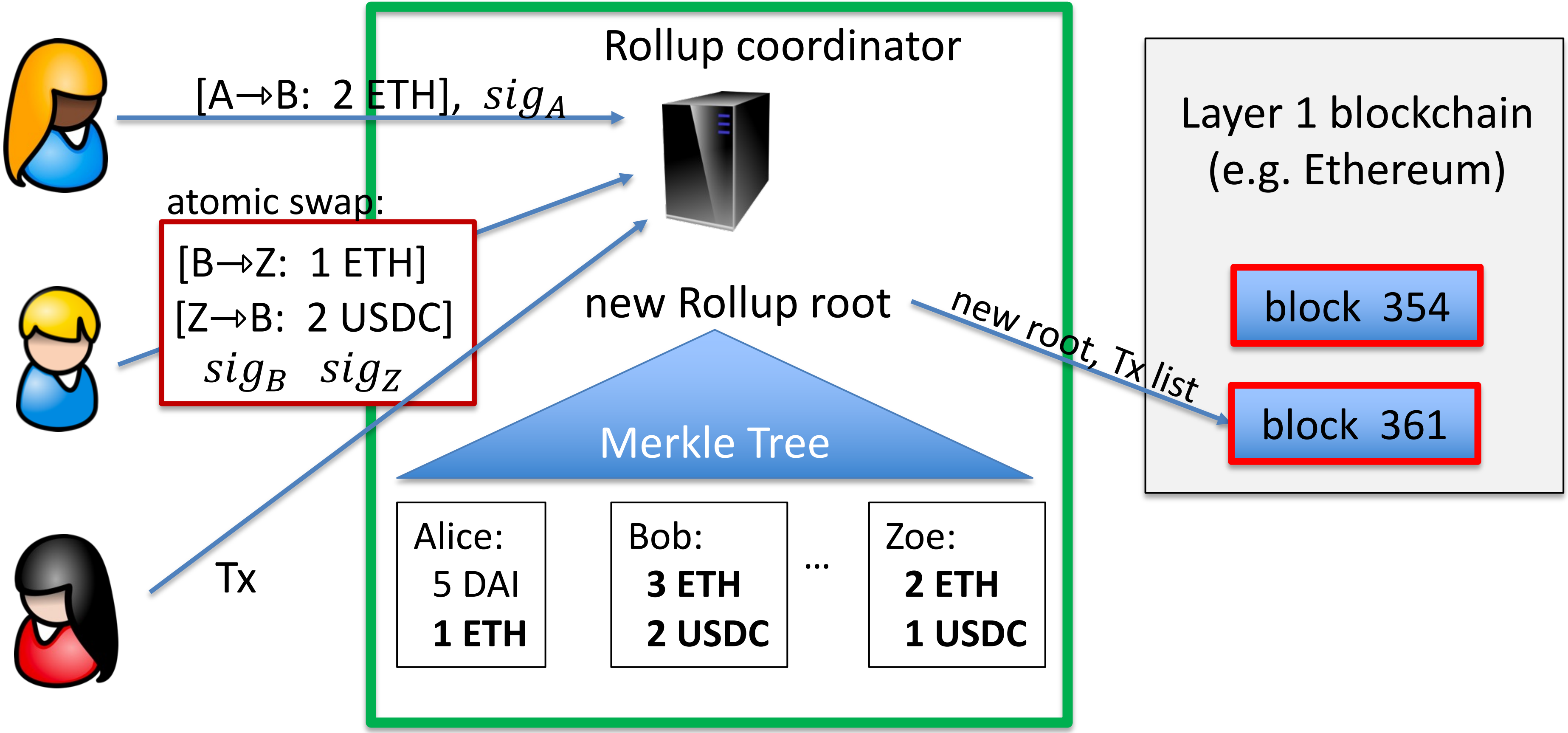
…

Zoe:
1 ETH
3 USDC

Layer 1 blockchain
(e.g. Ethereum)

block 354

Rollup operation (simplified)



Rollup coordinator

[A→B: 2 ETH], sig_A

atomic swap:

[B→Z: 1 ETH]
[Z→B: 2 USDC]
sig_B sig_Z

Tx

new Rollup root

Merkle Tree

Alice:
5 DAI
1 ETH

Bob:
3 ETH
2 USDC

…

Zoe:
2 ETH
1 USDC

new root, Tx list

Layer 1 blockchain
(e.g. Ethereum)

block 354

block 361

In more detail

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

Rollup state (L2)

Alice: **4 ETH, 1 DAI**

Bob: **3 ETH, 2 DAI**

…

(coordinator stores state)

Alice: state

Bob: state

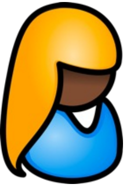
Uniswap: state

Rollup contract: **7 ETH, 3 DAI,** root

…

Layer-1 blockchain (L1)

Transfers inside Rollup are easy (L2 → L2)



[A→B: 2 ETH], sig_A

(with hundreds of Tx)

Rollup state (L2)

Alice:
4 ETH, 1 DAI

Bob:
3 ETH, 2 DAI

…

Alice:
state

Bob:
state

Uniswap:
state

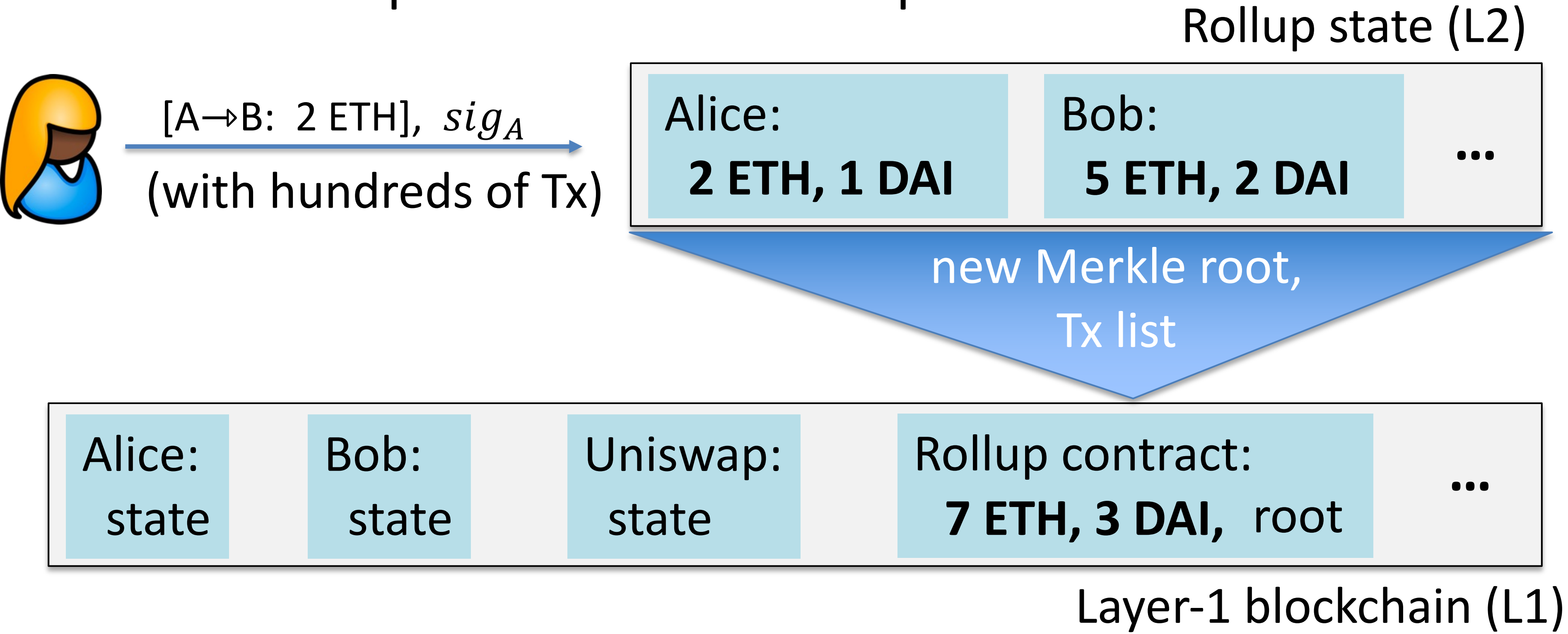
Rollup contract:
7 ETH, 3 DAI, root

…

Layer-1 blockchain (L1)

Transfers inside Rollup are easy (L2 → L2)

Coordinator updates root on Rollup contract



Rollup state (L2)

[A→B: 2 ETH], sig_A

(with hundreds of Tx)

Alice:
2 ETH, 1 DAI

Bob:
5 ETH, 2 DAI

…

new Merkle root,
Tx list

Alice:
state

Bob:
state

Uniswap:
state

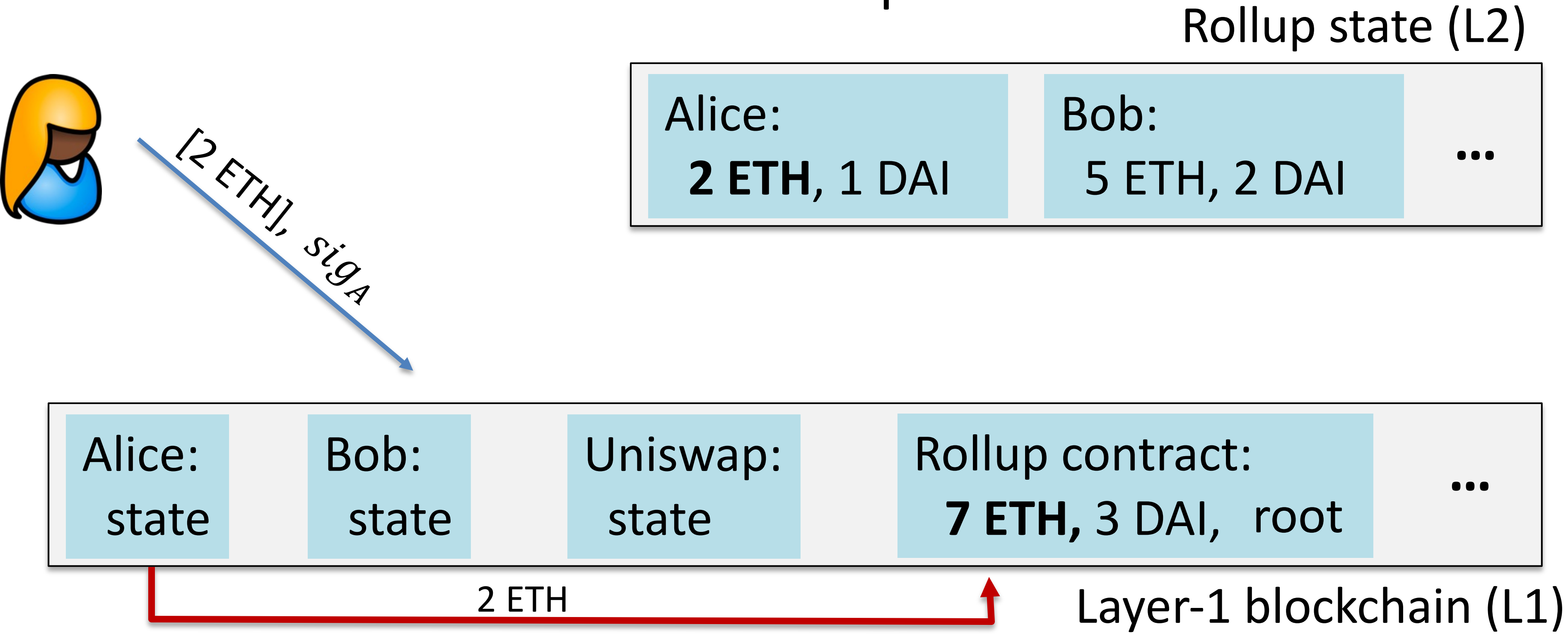
Rollup contract:
7 ETH, 3 DAI, root

…

Layer-1 blockchain (L1)

Transferring funds into Rollup (L1 → L2)

Alice issues an L1 Tx: slow and expensive



Rollup state (L2)

Alice:
2 ETH, 1 DAI

Bob:
5 ETH, 2 DAI

…

[2 ETH], sig_A

Alice:
state

Bob:
state

Uniswap:
state

Rollup contract:
7 ETH, 3 DAI, root

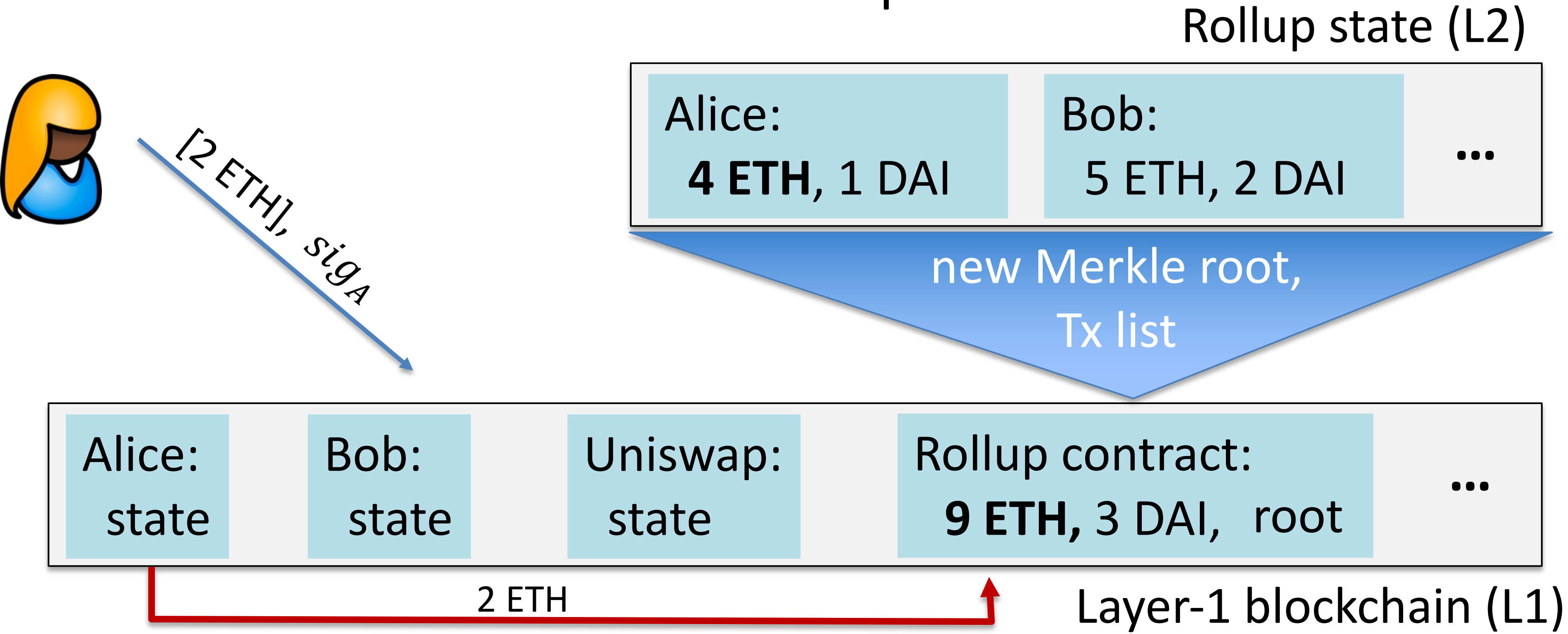
…

2 ETH

Layer-1 blockchain (L1)

Transferring funds into Rollup (L1 → L2)

Alice issues an L1 Tx: slow and expensive



Rollup state (L2)

Alice:
4 ETH, 1 DAI

Bob:
5 ETH, 2 DAI

…

new Merkle root,
Tx list

[2 ETH], sig_A

Alice:
state

Bob:
state

Uniswap:
state

Rollup contract:
9 ETH, 3 DAI, root

…

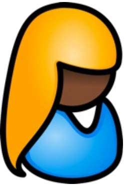
2 ETH

Layer-1 blockchain (L1)

Transferring funds out of Rollup (L2 → L1)

Requires extra gas on L1 to process transfer

Rollup state



[withdraw 4 ETH], sig_A

(plus hundreds of Tx)

Alice:
4 ETH, 1 DAI

Bob:
5 ETH, 2 DAI

…

Alice:
state

Bob:
state

Uniswap:
state

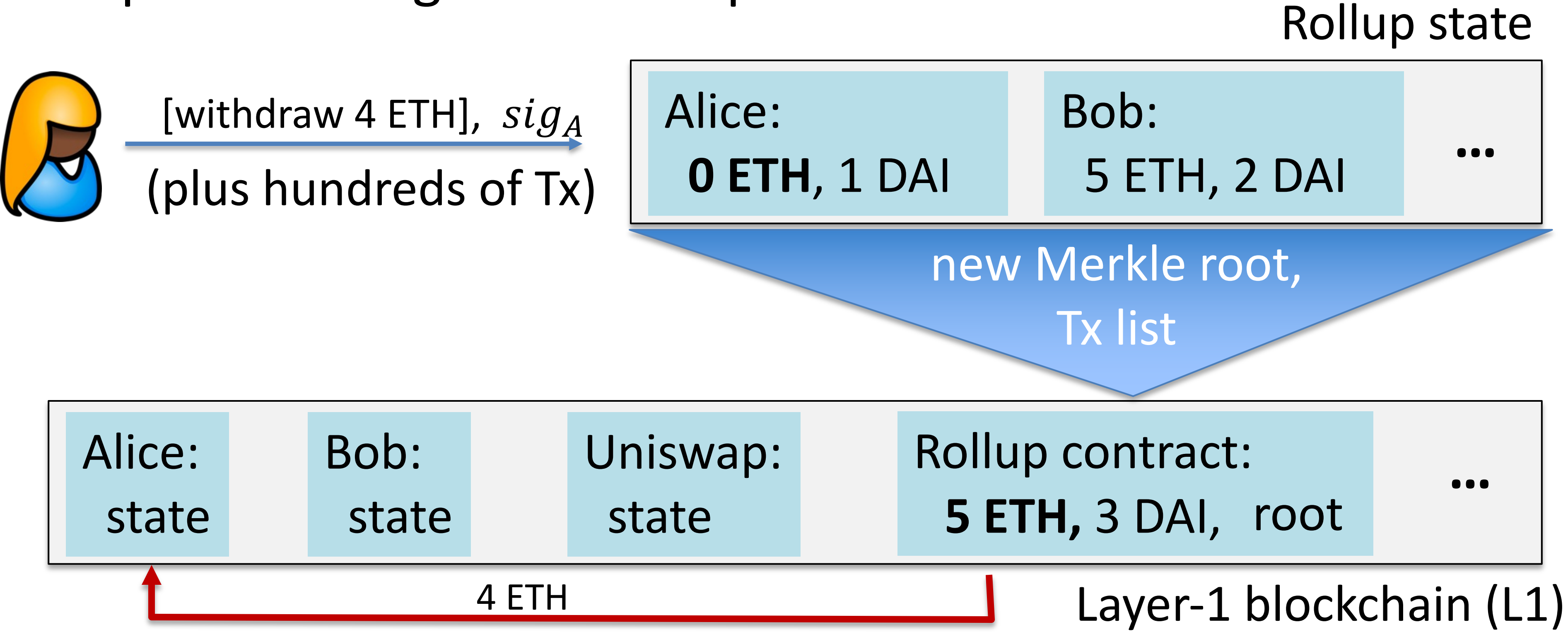
Rollup contract:
9 ETH, 3 DAI, root

…

Layer-1 blockchain (L1)

Transferring funds out of Rollup (L2 → L1)

Requires extra gas on L1 to process transfer



Rollup state

[withdraw 4 ETH], sig_A

(plus hundreds of Tx)

Alice:
0 ETH, 1 DAI

Bob:
5 ETH, 2 DAI

…

new Merkle root,
Tx list

Alice:
state

Bob:
state

Uniswap:
state

Rollup contract:
5 ETH, 3 DAI, root

…

4 ETH

Layer-1 blockchain (L1)

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 ⟺ L2) is expensive:

- Requires posting more data on L1 network ⟹ higher Tx fees.

Moving funds from one Rollup system to another (L2 ⟺ L2)

- Either via L1 network (expensive),
  or via a direct L2 ⟺ L2 bridge (cheap)

Running contracts on a Rollup?

Two copies of Uniswap

Rollup state (L2)

Alice:
4 ETH, 1 DAI

Bob:
3 ETH, 2 DAI

…

⇒ Rollup users can cheaply interact with Uniswap on Rollup

Alice:
state

Bob:
state

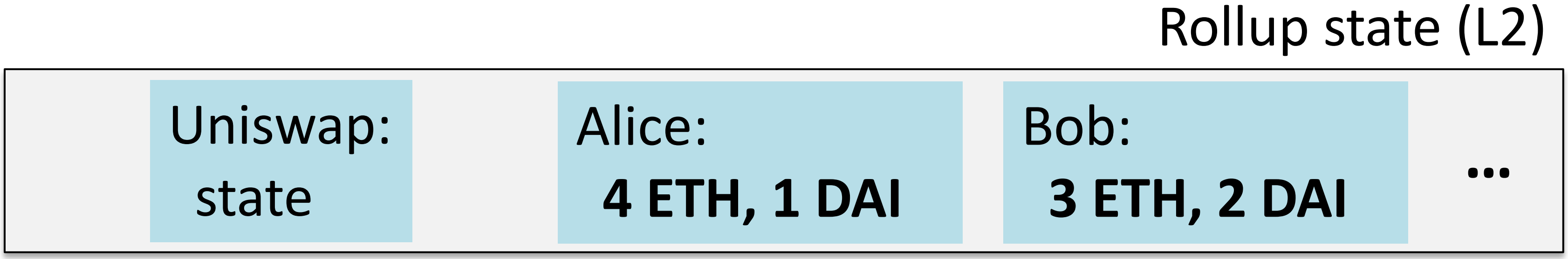
Uniswap:
state

Rollup contract:
7 ETH, 3 DAI, root

…

Layer-1 blockchain (L1)

Running contracts on a Rollup?



Rollup state (L2)

Uniswap:
state

Alice:
4 ETH, 1 DAI

Bob:
3 ETH, 2 DAI

…

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 state (L2)

Uniswap: state

Alice: **4 ETH, 1 DAI**

Bob: **3 ETH, 2 DAI**

…

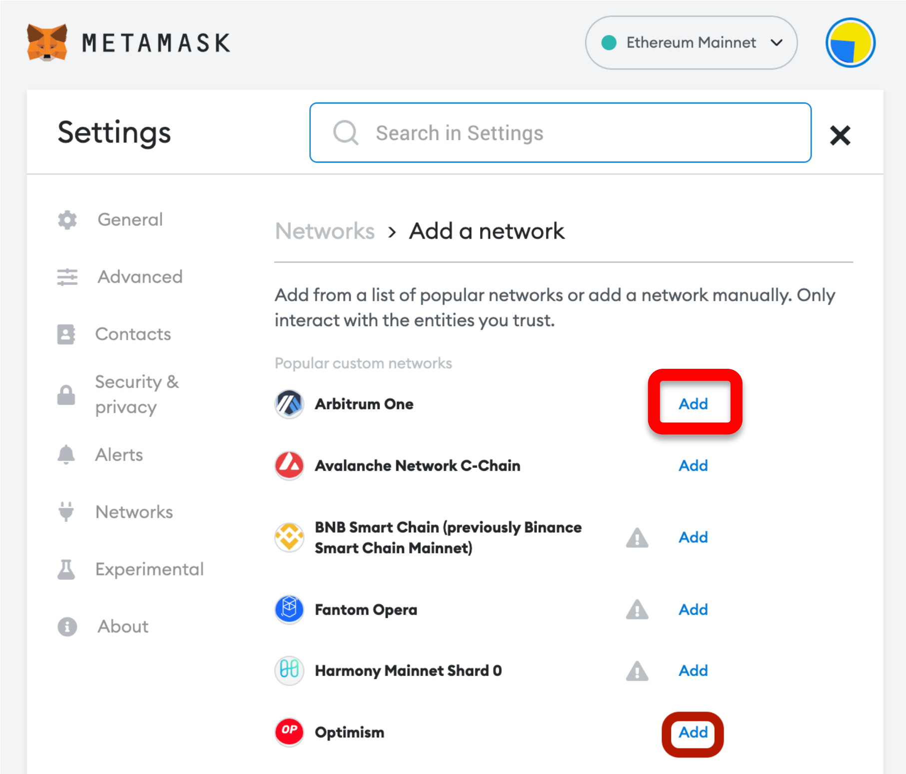
Rollup functions as Ethereum, but w/o a consensus protocol !!

- 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)



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

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 we initialize a new coordinator?

Problem 1: what if coordinator is dishonest?

Can coordinator steal funds from Rollup users?

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

⇒ all Tx are valid and properly signed by the Rollup users

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

Alice:
state

Bob:
state

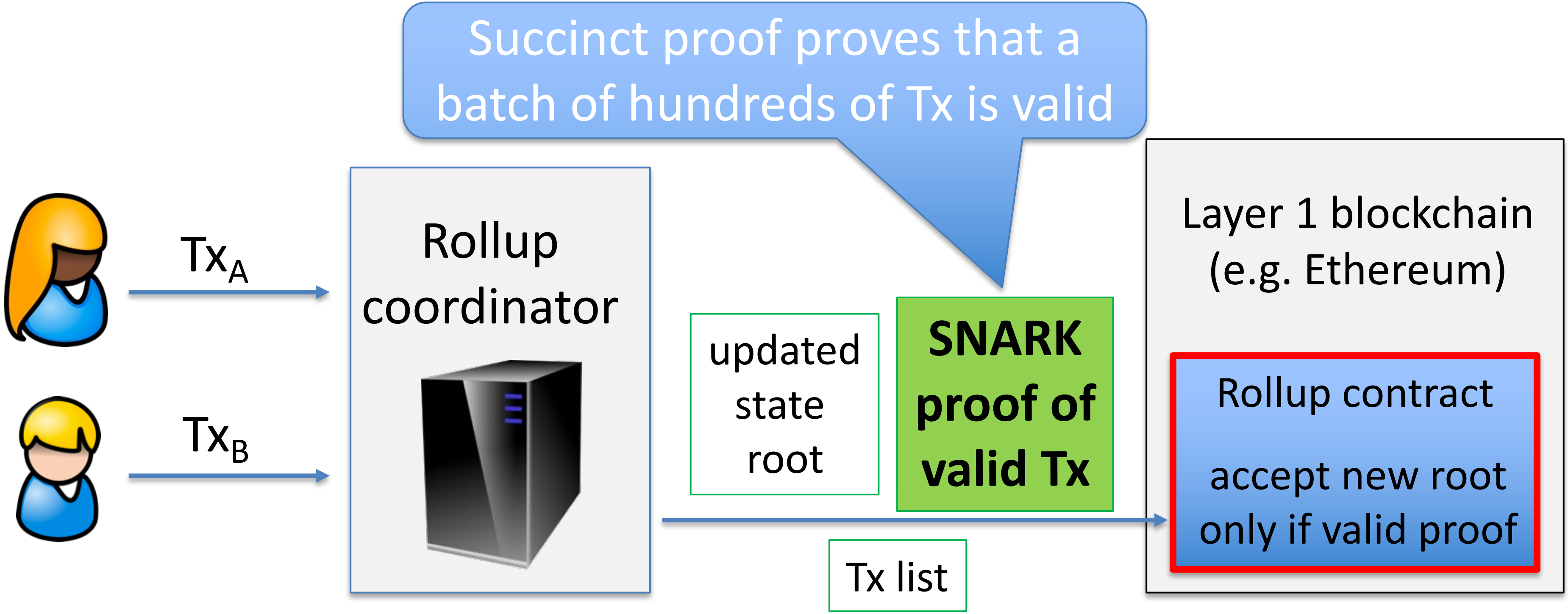
Rollup contract:
7 ETH, 3 DAI, root

…

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:

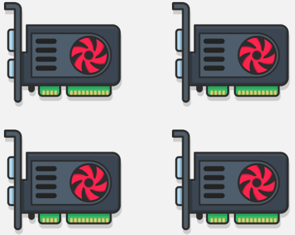
- coordinator builds a SNARK proof of correct execution of an EVM program ⇒ called a **zkEVM**
- Generating proof is a heavyweight computation

… verifying proof is fast

Two flavors of zkEVM:

- Prove that EVM bytecode ran correctly
  (Polygon zkEVM, Scroll)
- Compile Solidity to a SNARK-friendly circuit
  (MatterLabs)

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)

Verifying Rollup state updates

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

- Coordinator deposits stake in escrow on L1 Rollup contract

- Operation: Coordinator submits state updates to L1 w/o a proof
- If update is invalid: anyone has seven days to submit a fraud proof
  - Successful fraud proof means coordinator gets *slashed* on L1
  - Unsuccessful fraud proof costs complainer a fee

Challenge: how to prove fraud to Rollup contract on L1 ??

Naively: L1 can re-execute all Tx in batch ⇒ expensive and slow

Fraud Proof game

coordinator

pre-root

Tx list

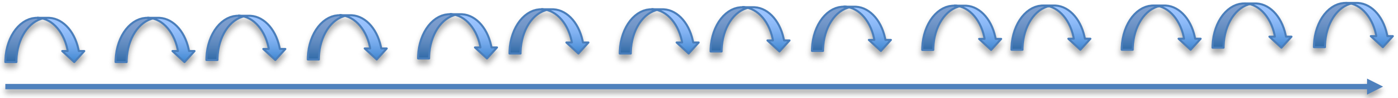
fraud claim

claimed post-root

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

different post-root

pre-state



post-state

break computation into small steps

Fraud Proof game

coordinator

claimed $state_n$

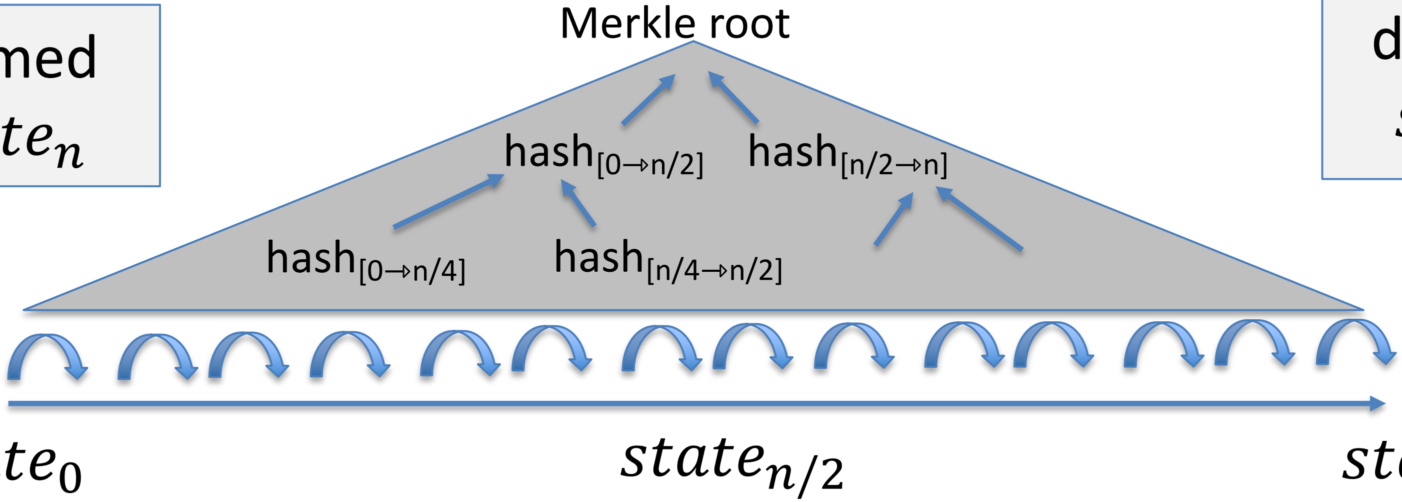
pre-root

Tx list

we know $state_n \neq state'_n$

fraud claim

different $state'_n$



Merkle root

hash$_{[0\rightarrow n/2]}$

hash$_{[n/2\rightarrow n]}$

hash$_{[0\rightarrow n/4]}$

hash$_{[n/4\rightarrow n/2]}$

$state_0$

$state_{n/2}$

$state_n$

Fraud Proof game: binary search

coordinator

claimed
$state_n$

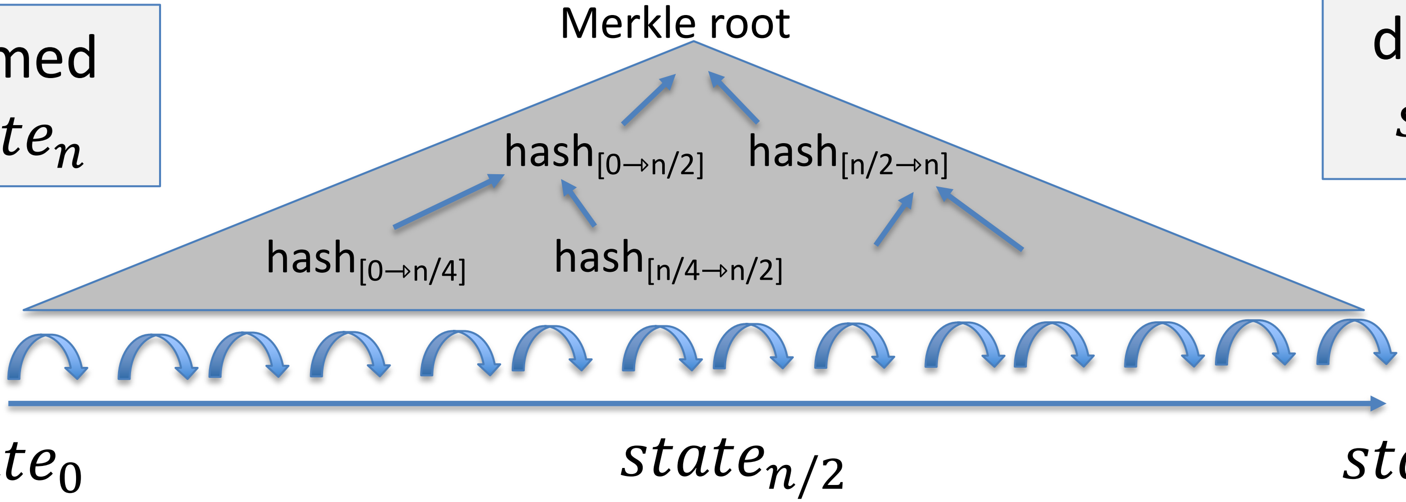
pre-root

Tx list

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

fraud claim

different
$state'_n$



Merkle root

$hash_{[0 \to n/2]}$

$hash_{[n/2 \to n]}$

$hash_{[0 \to n/4]}$

$hash_{[n/4 \to n/2]}$

$state_0$

$state_{n/2}$

$state_n$

Fraud Proof game: binary search

coordinator

claimed $state_n$

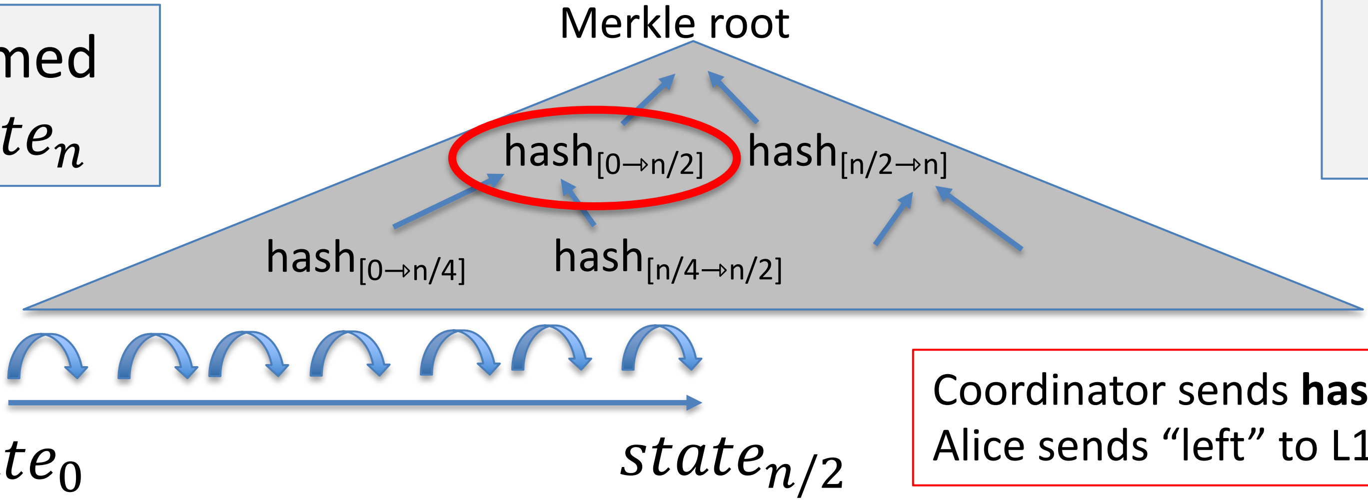
pre-root

Tx list

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

fraud claim

different $state'_n$



Merkle root

hash$_{[0\to n/2]}$

hash$_{[n/2\to n]}$

hash$_{[0\to n/4]}$

hash$_{[n/4\to n/2]}$

$state_0$

$state_{n/2}$

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

Fraud Proof game: binary search

coordinator

claimed $state_n$

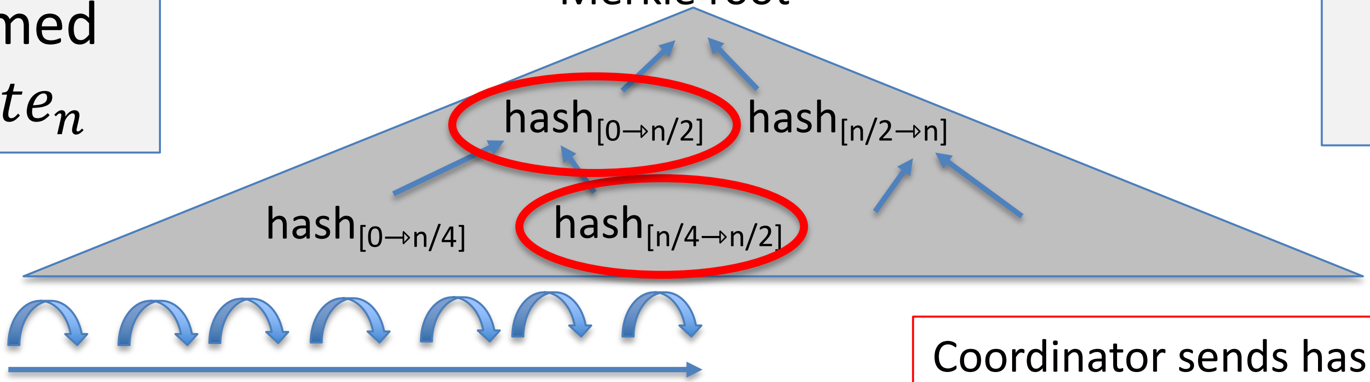
pre-root

Tx list

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

fraud claim

different $state'_n$



Merkle root

$\text{hash}_{[0 \to n/2]}$ $\text{hash}_{[n/2 \to n]}$

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

$state_0$ $state_{n/4}$ $state_{n/2}$

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

Fraud Proof game: binary search

coordinator

claimed $state_n$

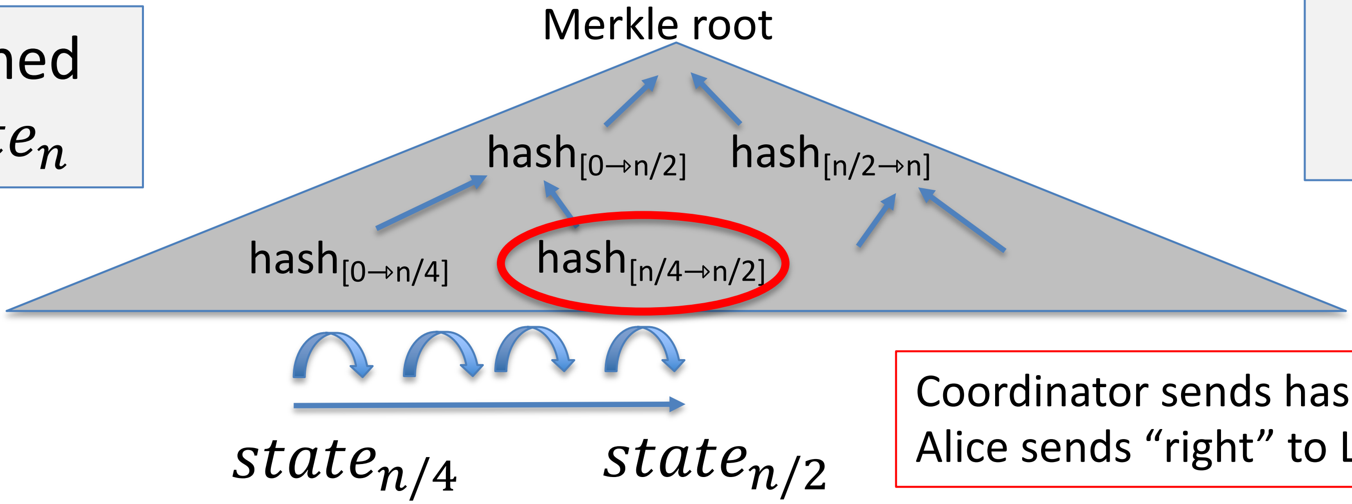
pre-root

Tx list

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

fraud claim

different $state'_n$



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

Fraud 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

⇒ Now L1 can verify fraud proof by checking **one** computation step!

Some difficulties

(1) Transactions only settle after 7 days (after fraud 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 fraud proof 4 days after batch is posted

⇒ all subsequent Tx need to be resubmitted

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 verifier

Downside: 7 day finality delay

An example (StarkNet -- using validity proofs)

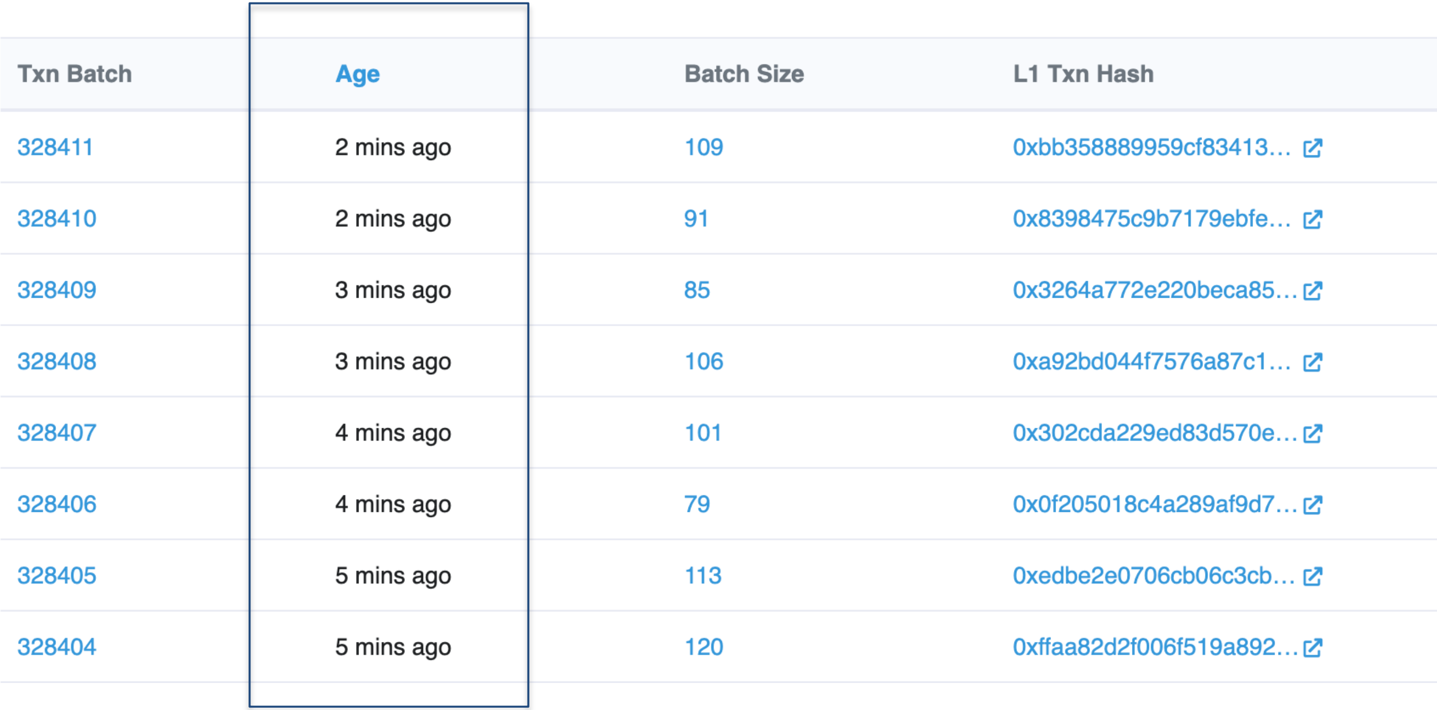
Block

Number	Hash	Status	Num. of Txs	Age
PENDING	PENDING	PENDING	64	3min
13011	0x0432…2380	ACCEPTED_ON_L2	82	8min
13010	0x0492…f0d1	ACCEPTED_ON_L2	122	15min
13009	0x0081…b7af	ACCEPTED_ON_L2	127	24min
		• • •		
12868	0x060c…15eb	ACCEPTED_ON_L2	58	8h
12867	0x0654…3b0f	ACCEPTED_ON_L1	72	9h
12866	0x0779…57d6	ACCEPTED_ON_L1	63	9h
12865	0x06ae…943f	ACCEPTED_ON_L1	97	9h

Tx posted on L1 (Ethereum) about every eight hours

Source: starkscan.co

An example (Optimism -- using fraud proofs)



Txn Batch	Age	Batch Size	L1 Txn Hash
328411	2 mins ago	109	0xbb358889959cf83413…
328410	2 mins ago	91	0x8398475c9b7179ebfe…
328409	3 mins ago	85	0x3264a772e220beca85…
328408	3 mins ago	106	0xa92bd044f7576a87c1…
328407	4 mins ago	101	0x302cda229ed83d570e…
328406	4 mins ago	79	0x0f205018c4a289af9d7…
328405	5 mins ago	113	0xedbe2e0706cb06c3cb…
328404	5 mins ago	120	0xffaa82d2f006f519a892…

Shows batch posted on L1 (Ethereum)

Source: optimistic.etherscan.io

… ok, so coordinator cannot submit invalid Tx.

Problem 2: What if coordinator stops providing service?

Solution: setup a new coordinator

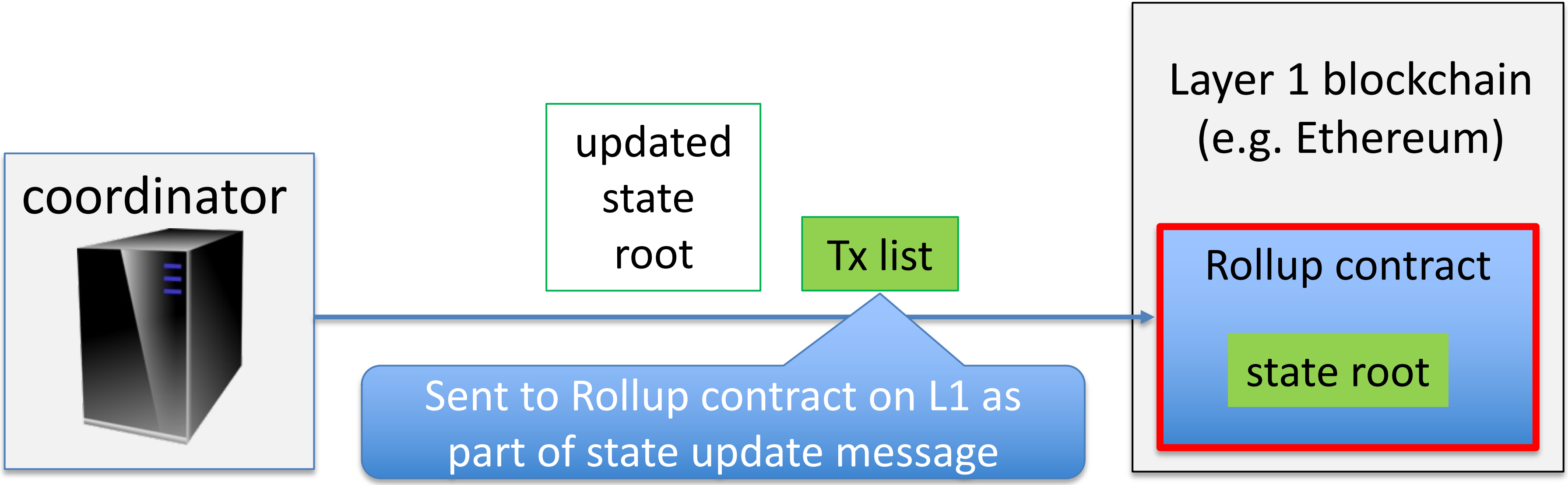
… but need the latest Rollup state

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



Ensuring Rollup state is always available

To reconstruct current Rollup state:

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

  ⇒ 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

(EIP-4488 aims to support Rollups by reducing to 3 gas/byte)

In practice:

- Optimistic Rollups fee/Tx: 3-8 times lower than Ethereum L1
- zk-Rollup fee/Tx: 40-100 times lower than Ethereum L1

Can we do even better?

Data Availability Committee (DAC)

To further reduce Tx fees:

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

    ⇒ 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 **five** 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

availability

	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”

Volume of some L2 systems

	Tx Volume/day	average fee/tx	(on Nov. 15, 2022)
Ethereum:	1013K Tx	**2.71 USD/Tx**	
Arbitrum:	345K Tx	0.08 USD/Tx	(optimistic Rollup)
Optimism:	303K Tx	0.13 USD/Tx	(optimistic Rollup)
StarkNet:	14K Tx	0.22 USD/Tx	(zkRollup)

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

⇒   censorship will cause the entire Rollup to freeze

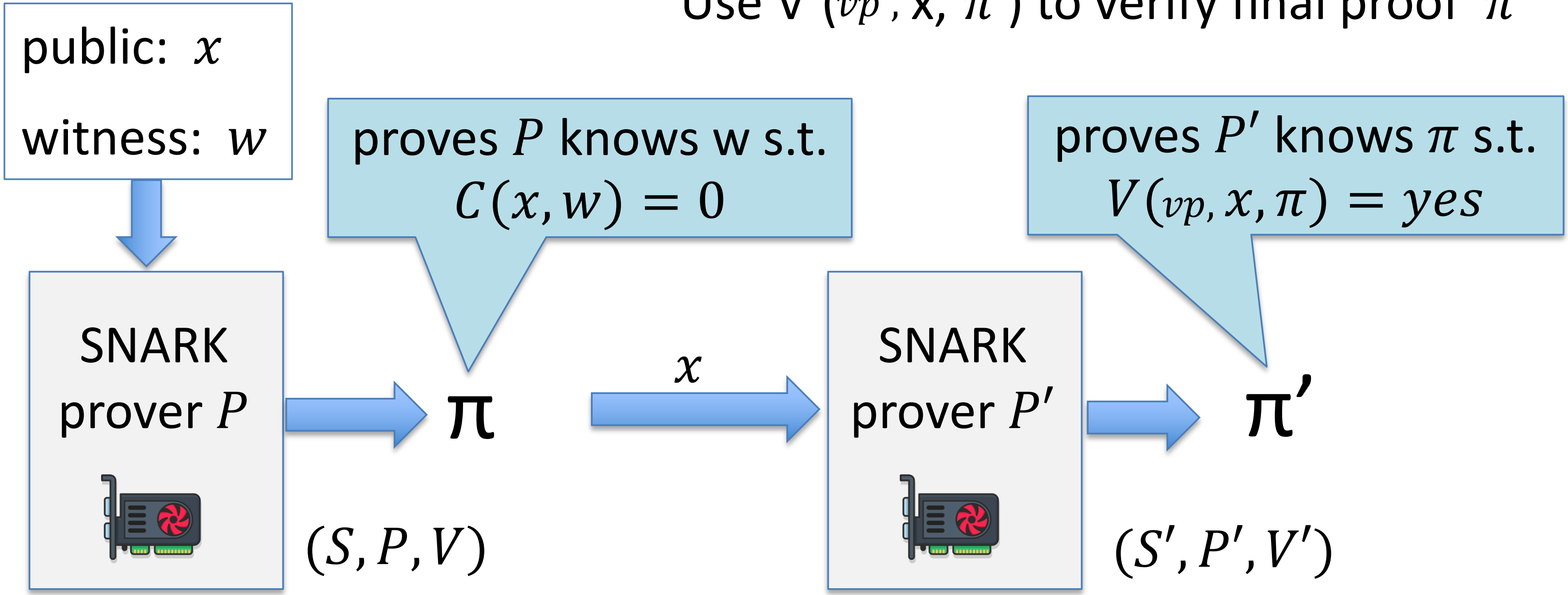
SNARK recursion

Layer 3 and beyond …

SNARK recursion

Two level recursion: **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'

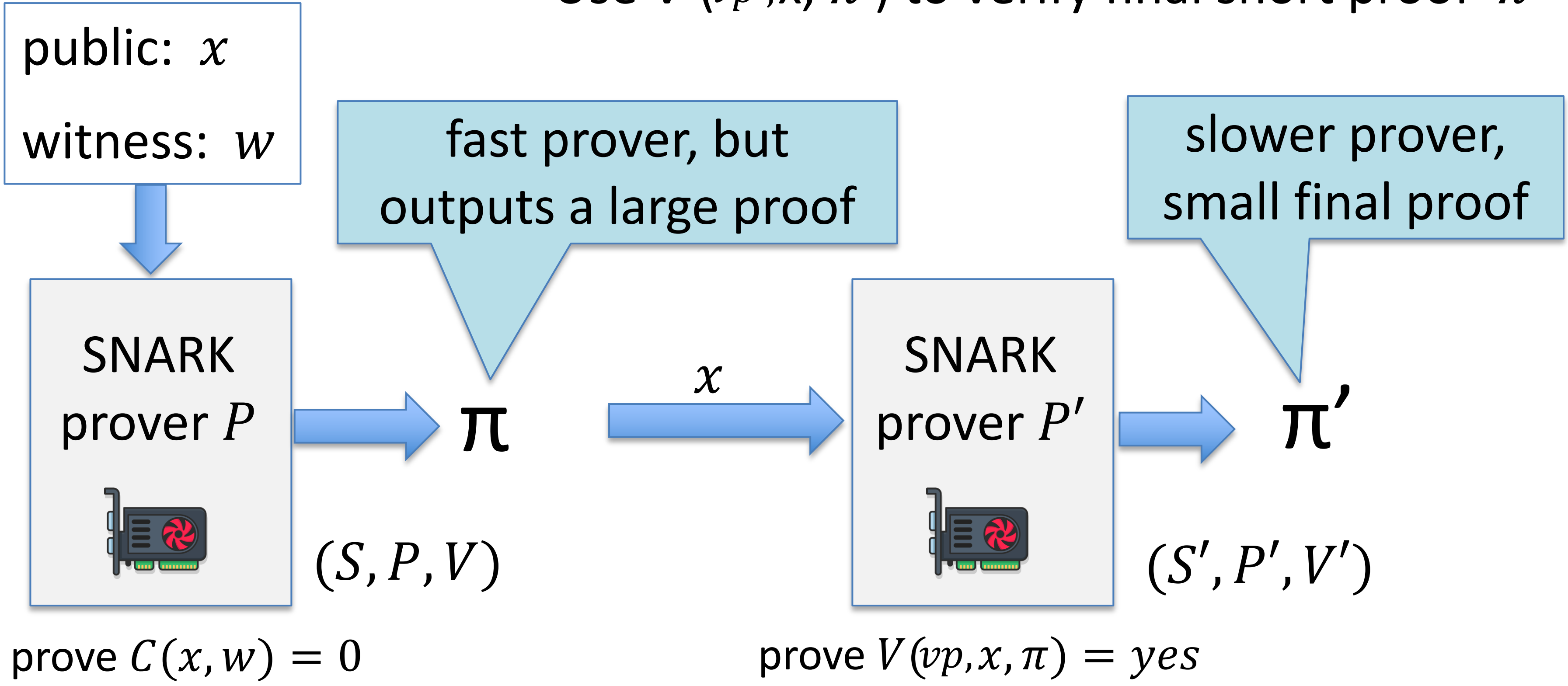
π’

(S, P, V)

(S', P', V')

Application 1: proof compression

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



public: x

witness: w

fast prover, but outputs a large proof

slower prover, small final proof

SNARK prover P

π

x

SNARK prover P'

π'

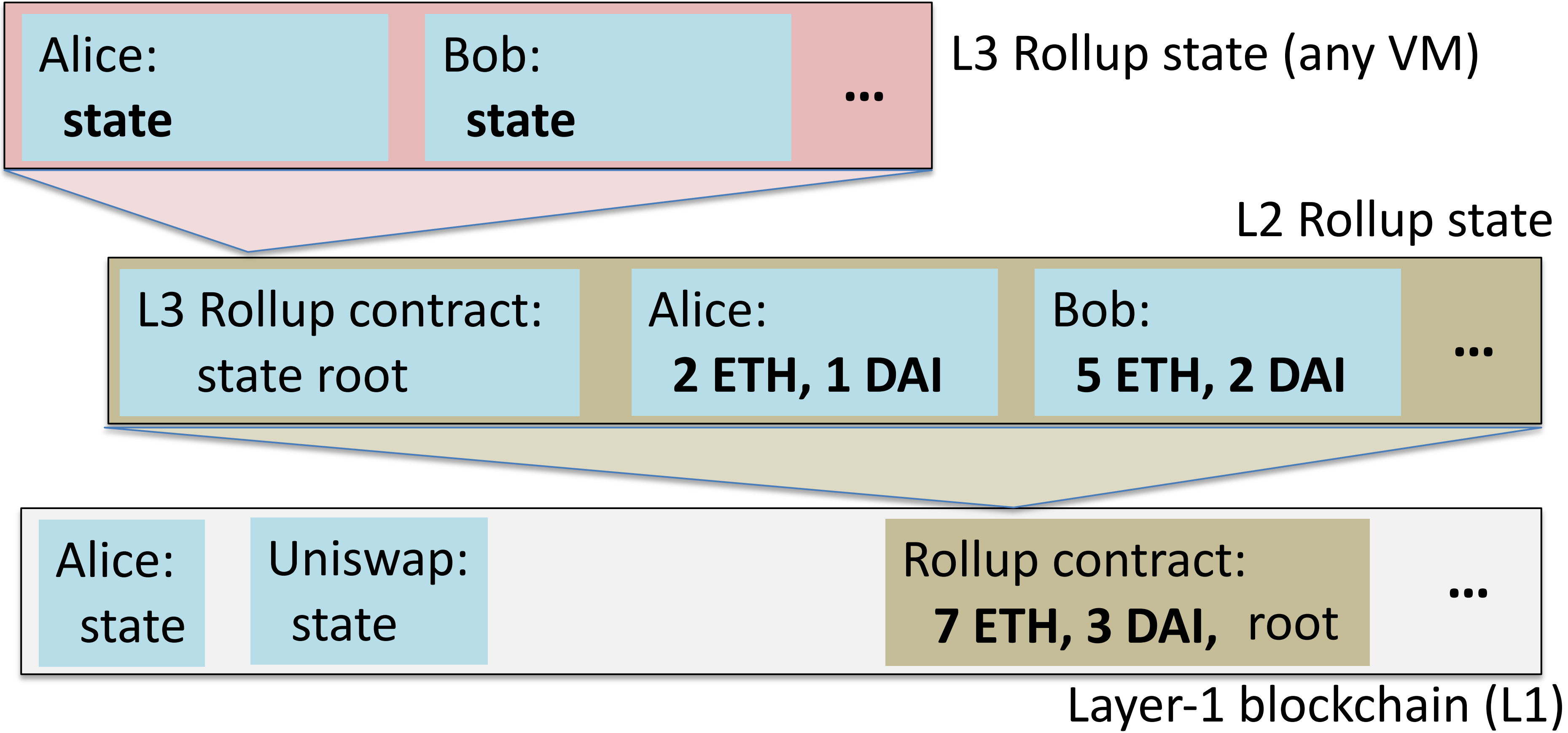
(S, P, V)

(S', P', V')

prove $C(x, w) = 0$

prove $V(vp, x, \pi) = yes$

Application 2: Layer three and beyond



Alice: **state**

Bob: **state**

…

L3 Rollup state (any VM)

L2 Rollup state

L3 Rollup contract: state root

Alice: **2 ETH, 1 DAI**

Bob: **5 ETH, 2 DAI**

…

Alice: state

Uniswap: state

Rollup contract: **7 ETH, 3 DAI,** root

…

Layer-1 blockchain (L1)

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.

$\Rightarrow$ Scaling factor 100×100

END OF LECTURE

Next lecture: more applications