Blockchain Technology and Applications

CS 731

Ethereum

Dr. Ir. Angshuman Karmakar

IIT Kanpur

Teaching assistants

- **Sumit Lahiri (**sumitl@cse.iitk.ac.in**)**
- **Chavan Sujeet** (sujeetc@cse.iitk.ac.in)
- **Indranil Thakur** (indra@cse.iitk.ac.in)

Ethereum

Motivation

- Bitcoins or blockchains started to gain popularity around 2013
- General public consensus was that blockchains were useful
- Revolutionary technology
- Not just currency
- Many other applications
  - Data storage
  - E-voting
  - Domain registration
  - Gambling
  - Etc.

Ethereum

Motivation

- Most blockchains were designed around one application
  - Bitcoin for currency
  - Namecoin for decentralized DNS
  - Primecoin for distributed prime number search
  - Etc.
- New application, add a new transaction type
  - Difficult to maintain
- Why not build a platform that can execute all transactions?

Ethereum

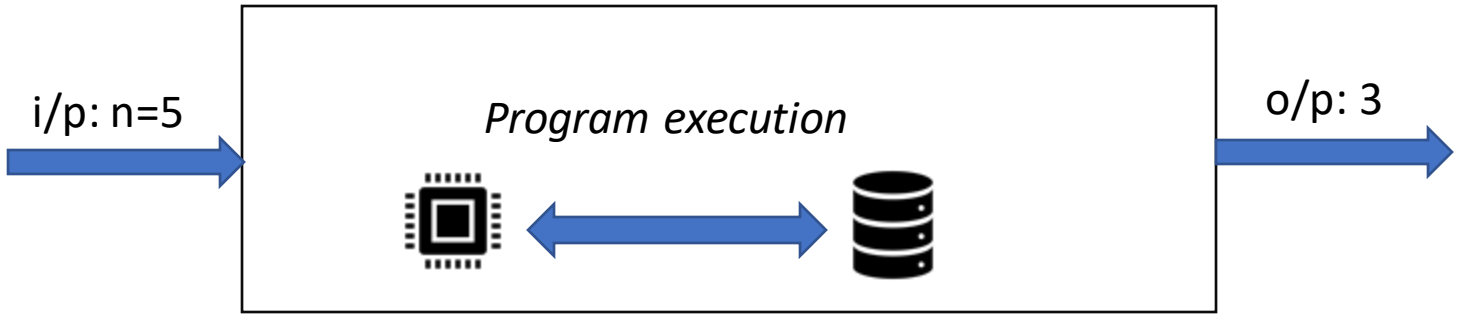
Motivation

- Main idea
  - Write codes that can execute arbitrary, complex programs
  - Programs take care of the transactions and protocols
- Bitcoin --> A decentralized global currency
- Ethereum --> A decentralized global computer
- Where this program runs?

Ethereum

Execution states : Simple example

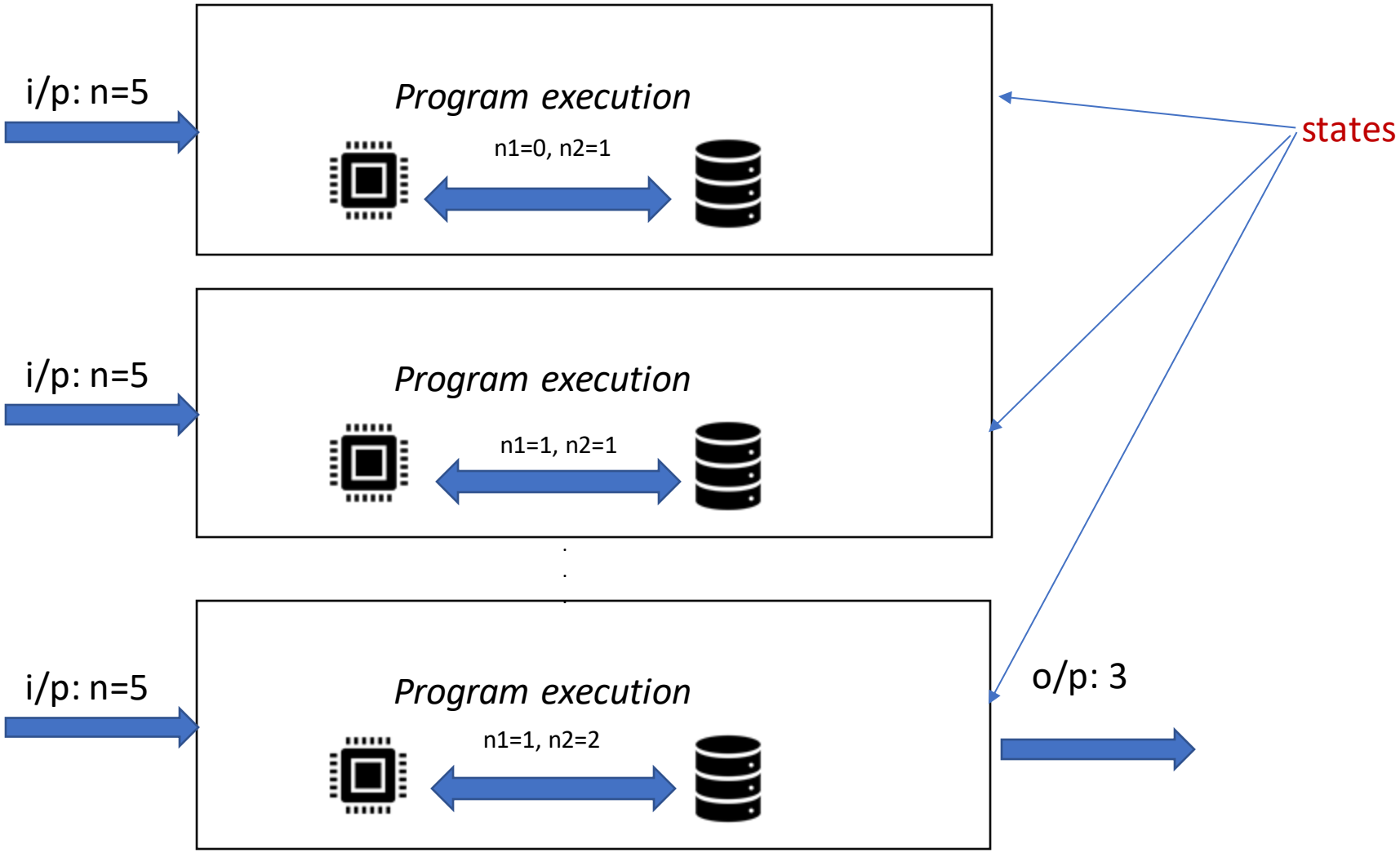
- *Find n-the Fibonacci number*



Ethereum

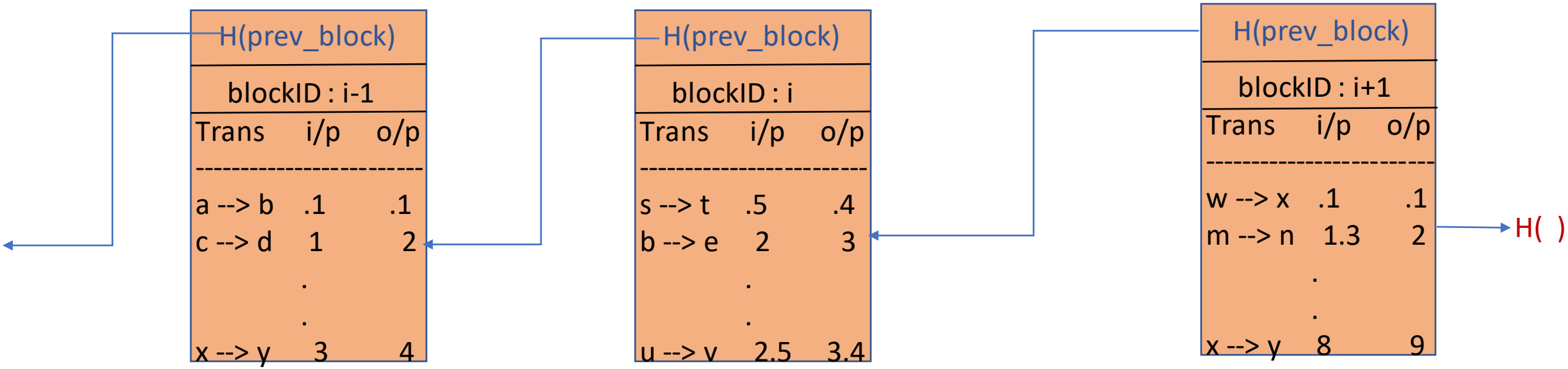
Execution states : Simple example

- *Find n-the Fibonacci number*



Ethereum

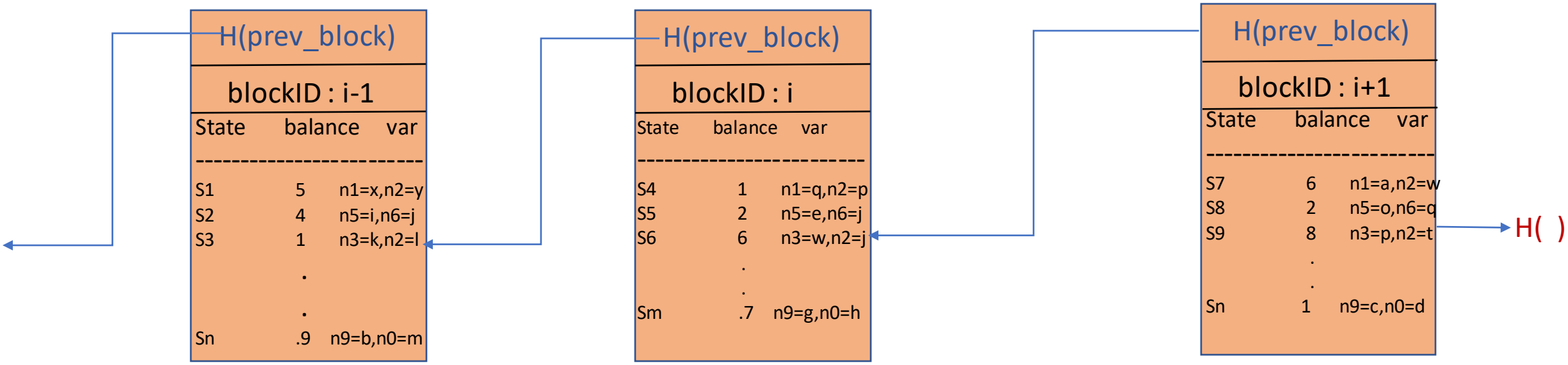
Motivation



- Bitcoin : a robust currency system with no double spending, authentication, etc.

Ethereum

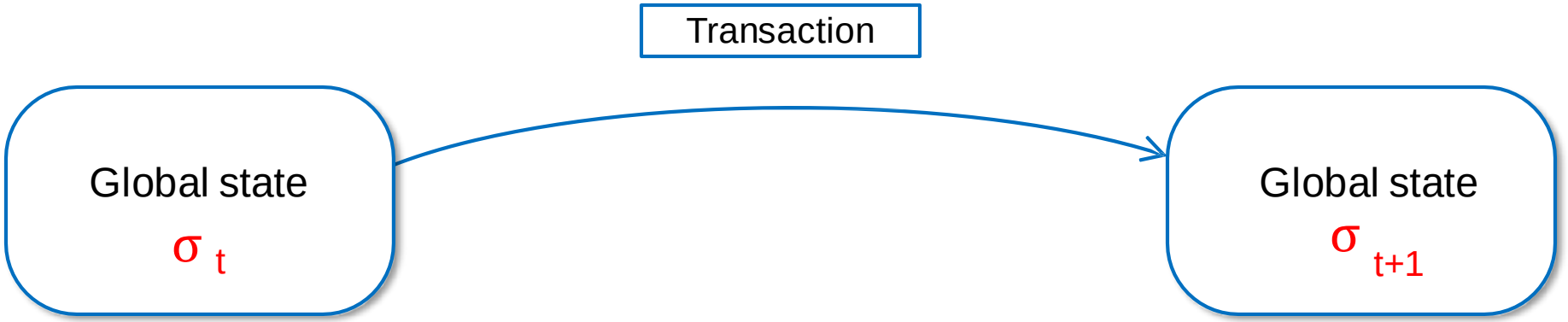
Global execution environment



- Ethereum : defines a virtual machine where execution states are maintained in the blockchain
- Distributed computing?

Ethereum

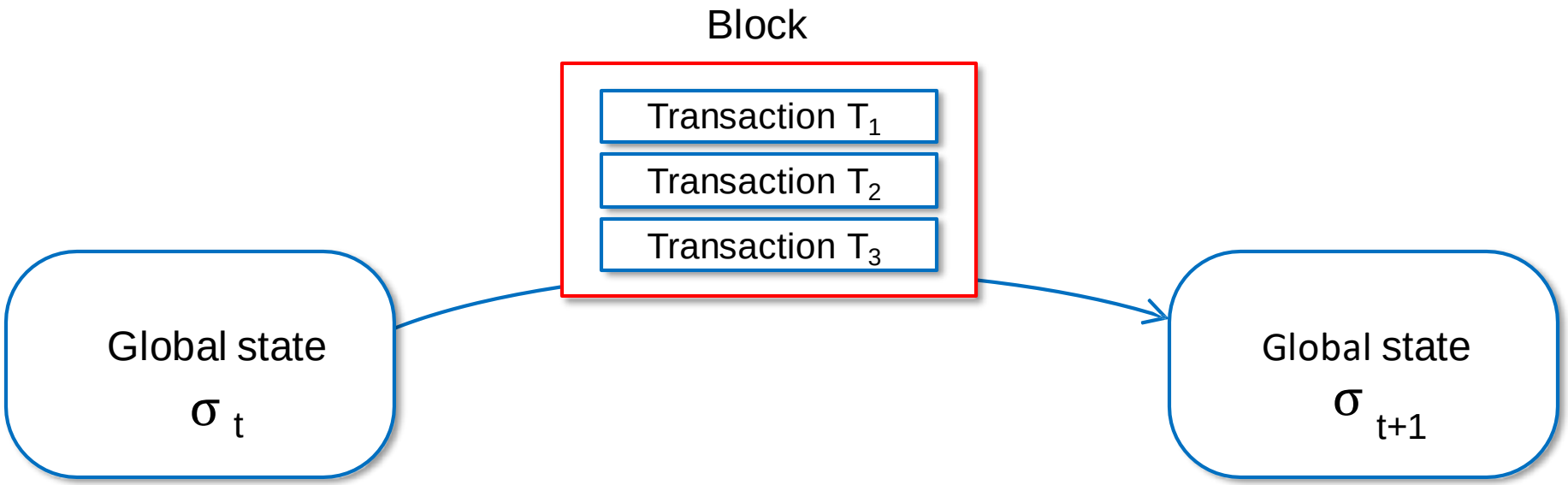
Transactions based state machine



- Ethereum can be viewed as a transaction-based state machine

Ethereum

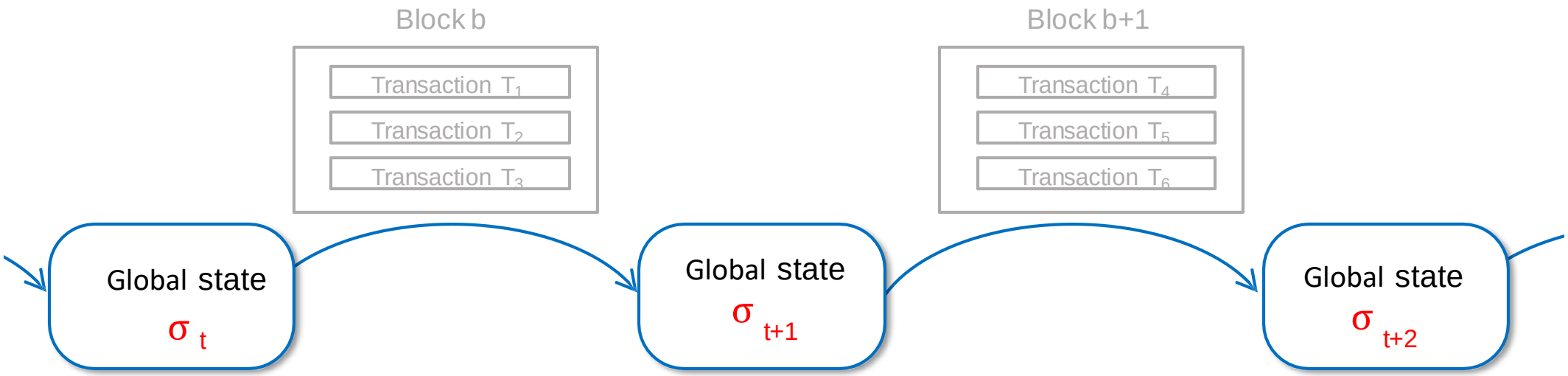
Transactions based state machine



- Multiple transactions are collated into blocks for efficiency

Ethereum

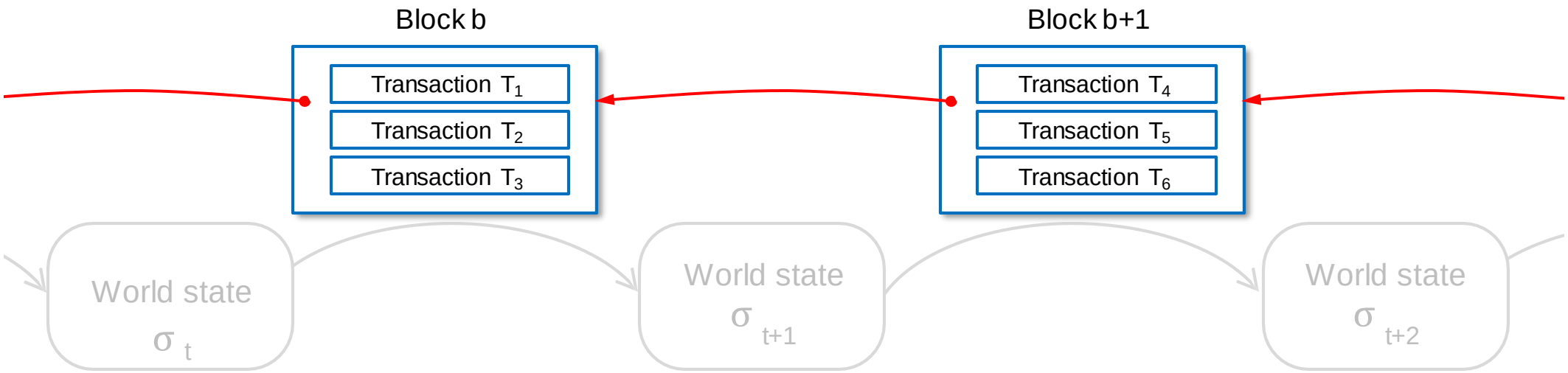
Transactions based state machine



- From the viewpoint of a *global computer* Ethereum can be seen as a state machine

Ethereum

As a blockchain



- From the viewpoint of the implementation, Ethereum can also be seen as a chain of blocks
- So, it is a blockchain

Ethereum

Use cases

- Decentralized Finance (DeFi)
  - No central authority
  - Always open
  - No censorship
  - Fairness and transparency
- Decentralized Autonomous organizations (DAO)
  - Owned and controlled by a group
  - Rules governed by smart contracts
  - Has treasuries, can only be accessed by consent of the group
  - E.g. : charitable organizations, ventures, freelancing networks, etc.
- Non-fungible tokens (NFT)
- And many more....

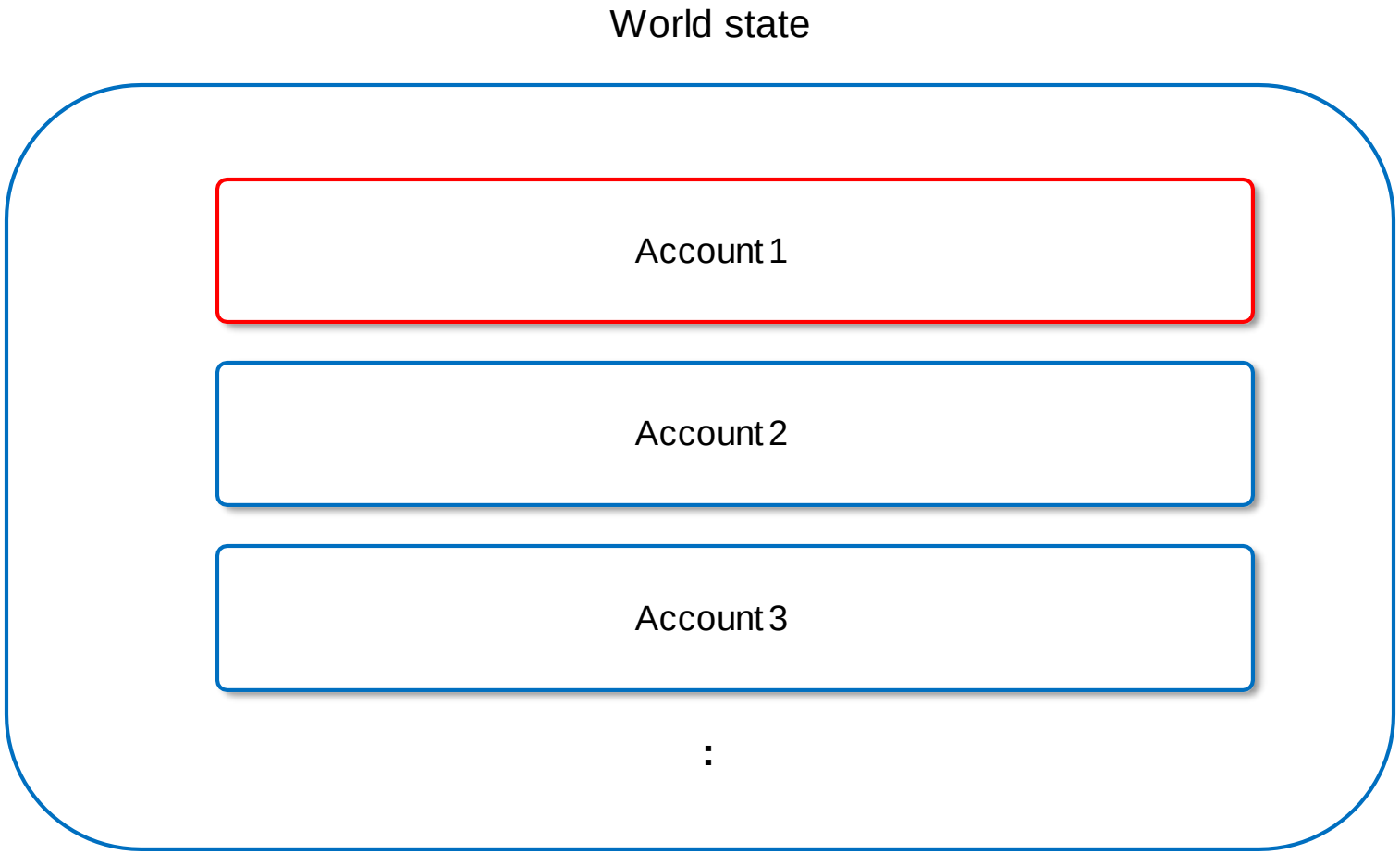
Mechanism

Ethereum

- Has its own programming language
  - Solidity
  - Turing complete
- One can program any protocol, application
  - Contracts
- Contracts have unique addresses
  - Can be invoked by a user or another contract
- There are two types of accounts in Ethereum
  - User accounts (externally owned account)
  - Contract accounts (contract account)

Ethereum

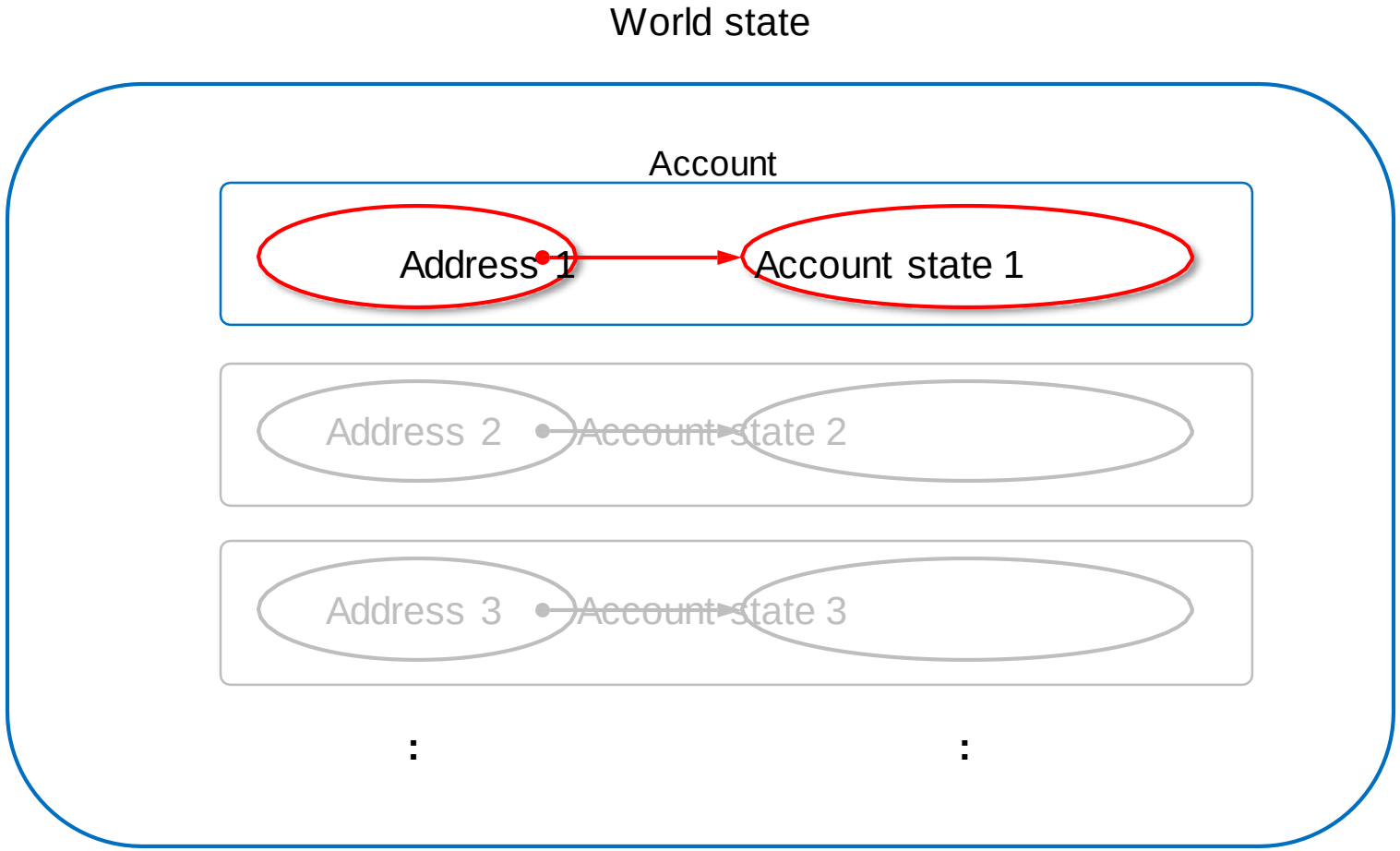
Accounts



- An account is a basic object in the world state

Ethereum

Accounts



- An account contains an address and account state
- Stack contents, variables, ether balance, etc.

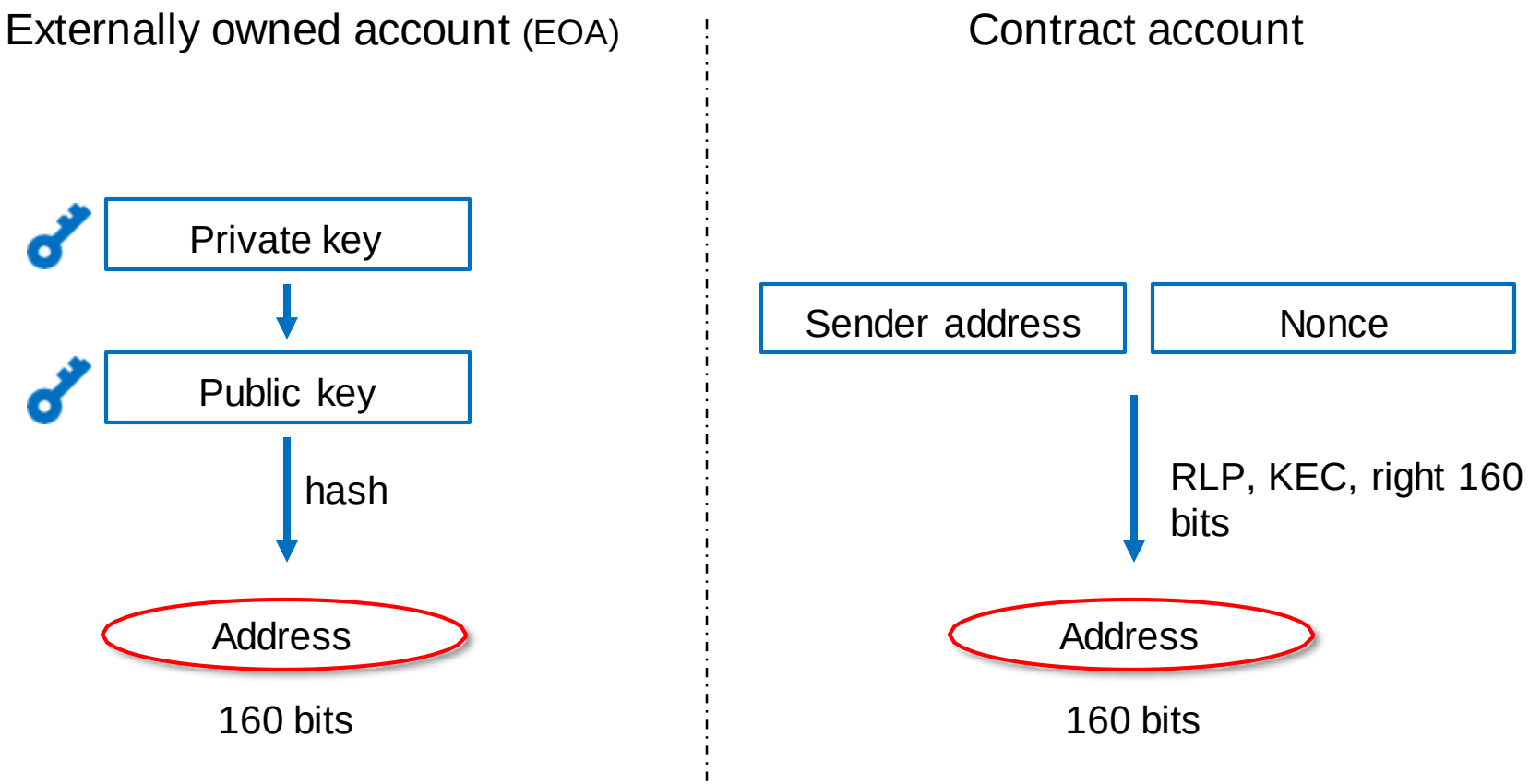
Ethereum

Accounts

- Externally owned account (EOA)
  - requires public-key
  - Created when one registers on the Ethereum network
- Contract account (CA)
  - Controlled by contracts/bots
  - Gets created when one contract is deployed
  - Has almost same privileges as EOA

Ethereum

Account addresses



EOA vs CA

	EOA	CA
Public/private key	Yes	No
Controlled by	Users/humans	Contract code
Gas (price)	Not required	Required
Code/Storage	No	Yes
Has address	Yes	Yes
Can hold ether	Yes	Yes

UTXO vs Account

- Bitcoin stores data about users' balances in *unspent transaction outputs* (UTXOs):
  - the entire state is defined by the UTXO set
- Transaction validity
  - Every referenced input must be valid and not yet spent
  - The transaction must have a signature matching the owner of the input
  - The total value of the inputs ≥ total value of the outputs
- A user's "balance" in the system is thus the total value of unspent coins
  - for which the user can prove ownership

UTXO vs Account

- In Ethereum, the state stores a list of accounts
  - each account has a balance, as well as Ethereum specific data (code and internal storage)
- a transaction is valid if the sending account has enough balance to pay for it,
  - the sending account is debited, and the receiving account is credited with the value
- If the receiving account has code
  - the code runs, and internal storage may also be changed,
  - the code may even create additional messages to other accounts which lead to further debits and credits

UTXO vs Account

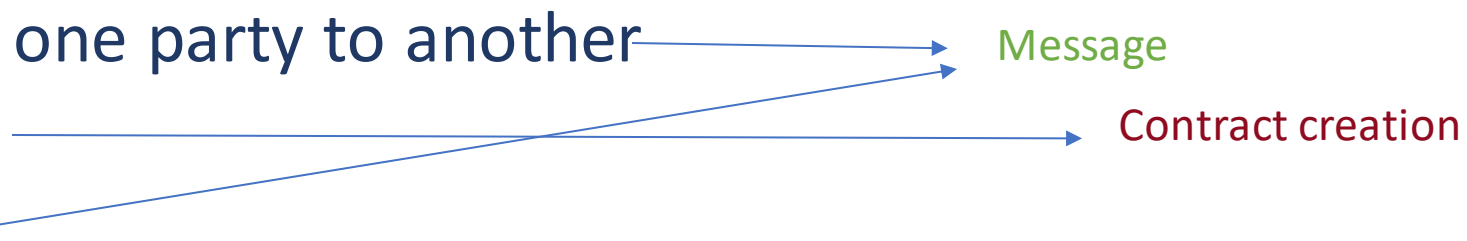
- Benefits of UTXO
  - Better privacy : gets new address after each transaction
  - Better scalable : only the owner needs to remember the Merkle proof of ownership
- Benefits of Accounts
  - Transactions are smaller 200-250 bytes (bitcoin) vs 100 bytes (Ethereum)
- Simple to code
  - Important once complex contracts are involved
- Replay attacks
  - Uses nonces to prevent replay attacks
  - Transaction is accepted only if the nonce is greater than last nonce

Transactions

Ethereum

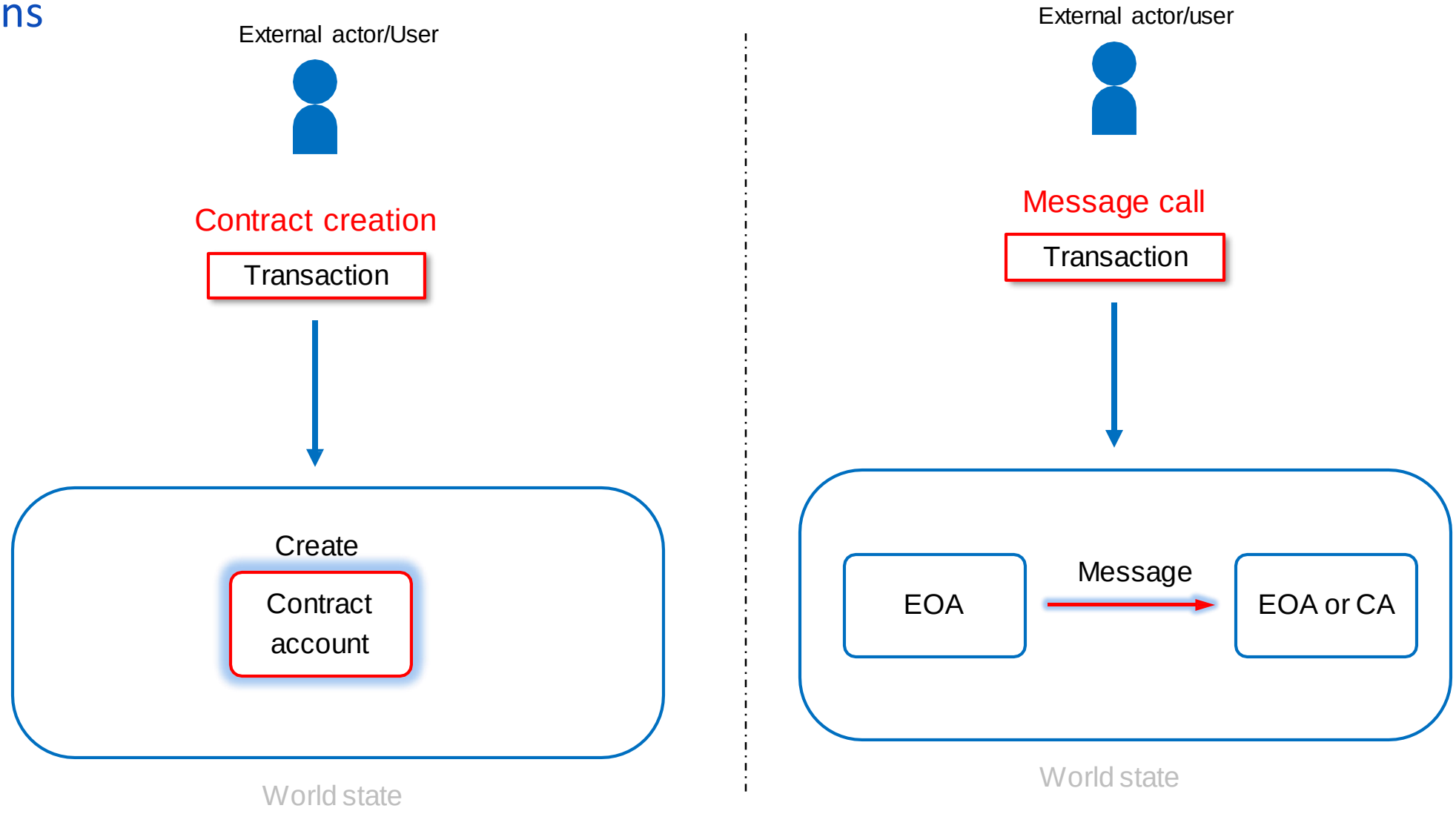
Transactions

- Transferring of Ether from one party to another
- Deploying smart contracts
- Invoking smart contracts



Ethereum

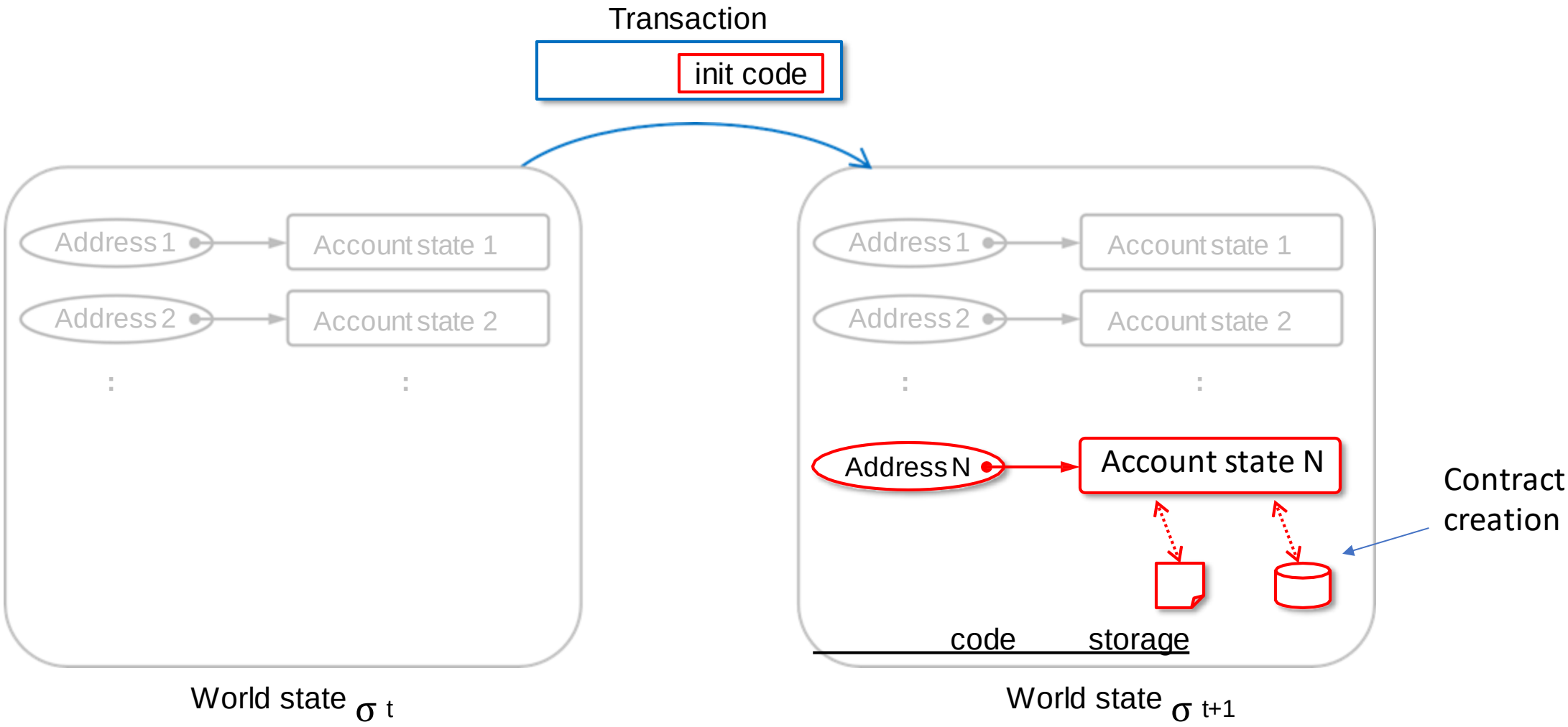
Transactions



- Only messages can invoke contracts
- EOA or CA can also pass messages to EOA or CA

Ethereum

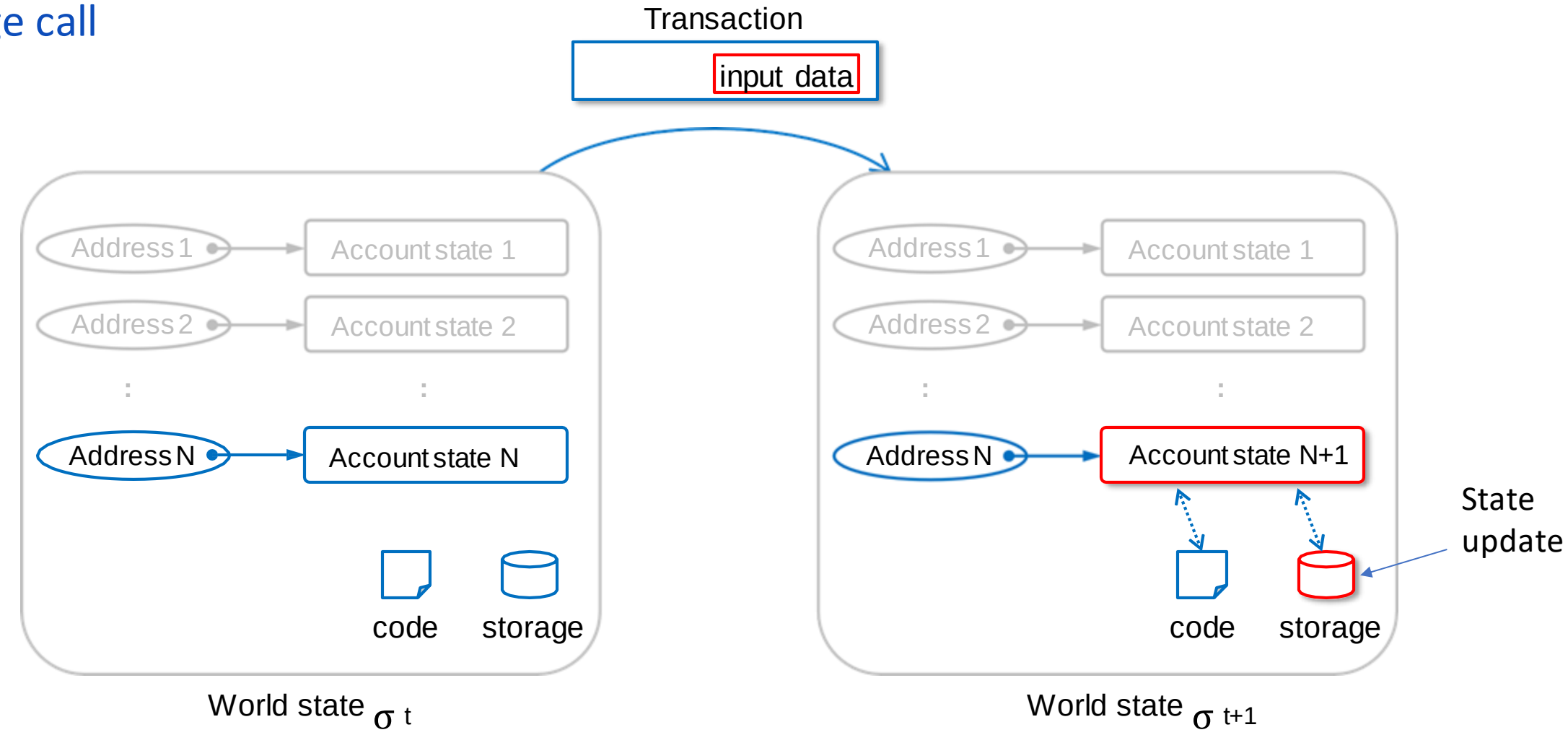
Contract creation



- The contract account gets its code, storage and possibly ether upon creation

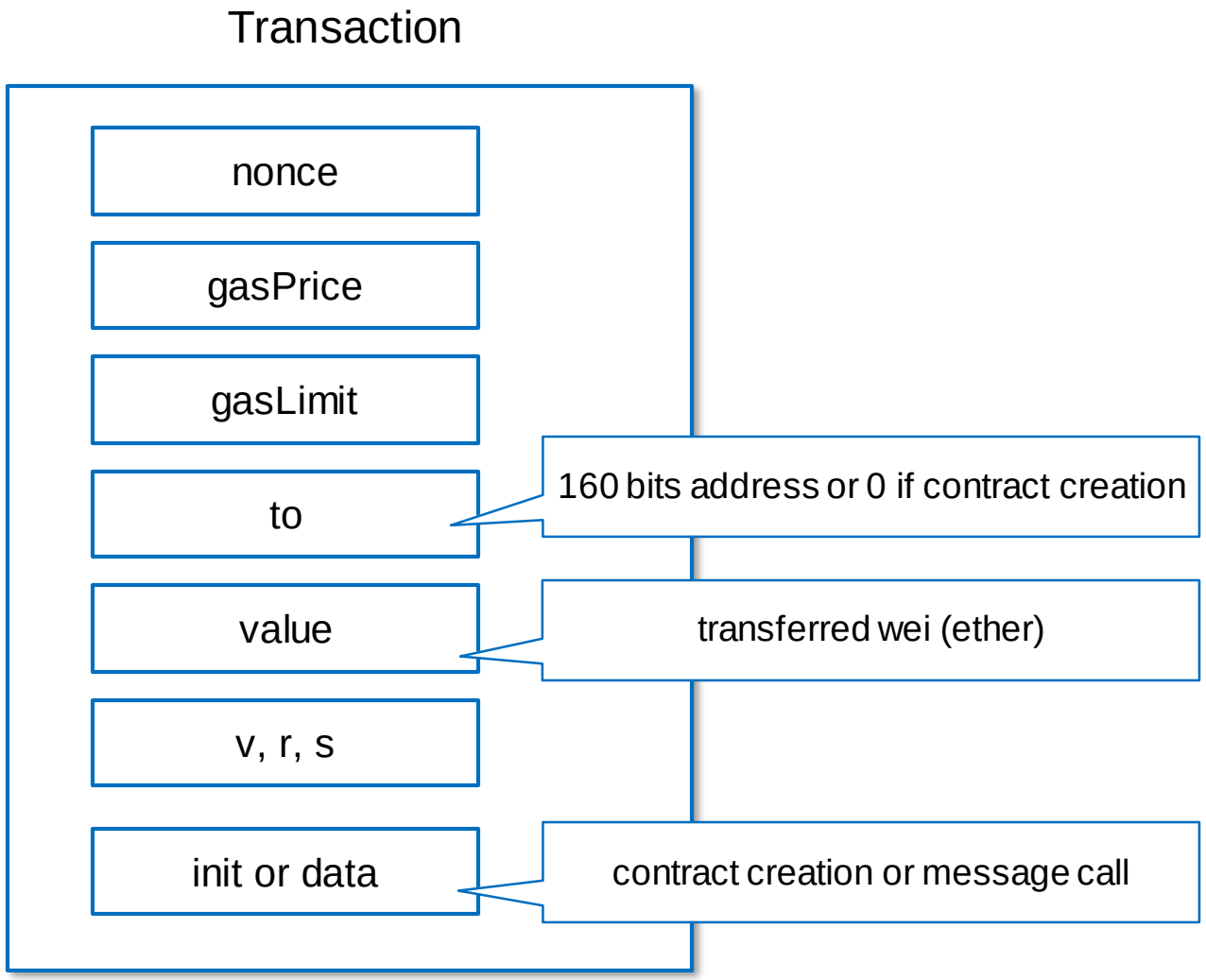
Ethereum

Message call



Ethereum

Fields of transaction



Program Execution

Ethereum

Program execution

- All codes run on Ethereum virtual machines
- Solidity compiler produces bytecode
  - This bytecode runs on EVM
- EVM has a stack and Memory Model
  - 32 byte instruction word size
  - No registers
  - Access to program *stack* : to store memory addresses for program counter, loop/jump, variable
  - An expandable temporary *memory*
  - Permanent *memory* which is written into the blockchain as program states
  - NO non-determinism allowed (e.g., no random() like function calls)

Ethereum

Program execution

- When an Ethereum block is “mined”,
  - The smart-contract deploys and the function call within that block gets executed on the mining node
  - State changes, memory access or transactions happen in the miner node
- This new block gets propagated to all the other nodes
  - Each miner node tries to independently verify the block,
  - Ensures that the state changes are consistent with their execution
- it will fail if the smart-contract acts non-deterministically.
  - If the other nodes cannot come to a consensus about the state of blockchain after the new block and its contracts get executed , the network could literally halt.

Ethereum

Program execution

- EVM smart-contracts cannot access data outside its *memory* and *storage*
  - *i.e.* the smart-contract can't read/write on the node's own hard-drive
- Cannot query outside resources
- Do not have access to many library functions like for parsing JSON structur-es or doing floating-point arithmetic,
  - Cost-prohibitive or stores too much data on the blockchain

Ethereum

Program execution

- When a smart-contract is executed that ultimately changes the state a *gas-cost* is incurred
  - Proportional to the amount of computation/memory required for the smart contract to execute
  - *Micropayment for microcomputing* system where it is expected to pay certain amount of gas for certain amount of computation forever
- The price of gas stays constant (unit : gwei)
  - This automatically adjusts the fluctuation of market price of ether
  - When ether price goes up price of gas goes down

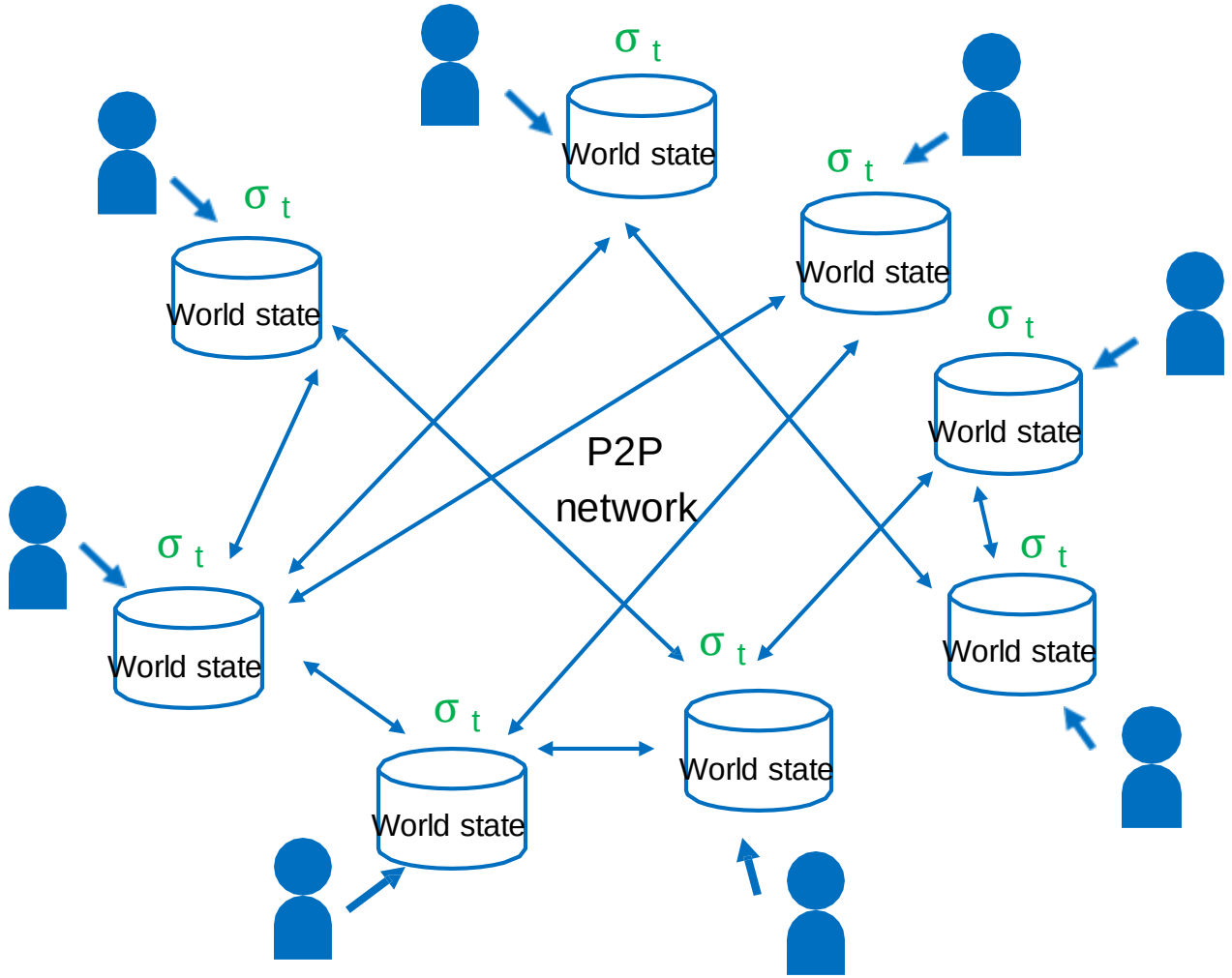
Ethereum

Program execution

- Gas limit specifies the minimum gas required to execute a contract
  - More gas limit more lucrative to miners
- User can specify gas price per unit it is willing to pay
  - The mining node decides if this is a good enough price for them
  - If yes, the smart contract function call gets included in their next block

Ethereum

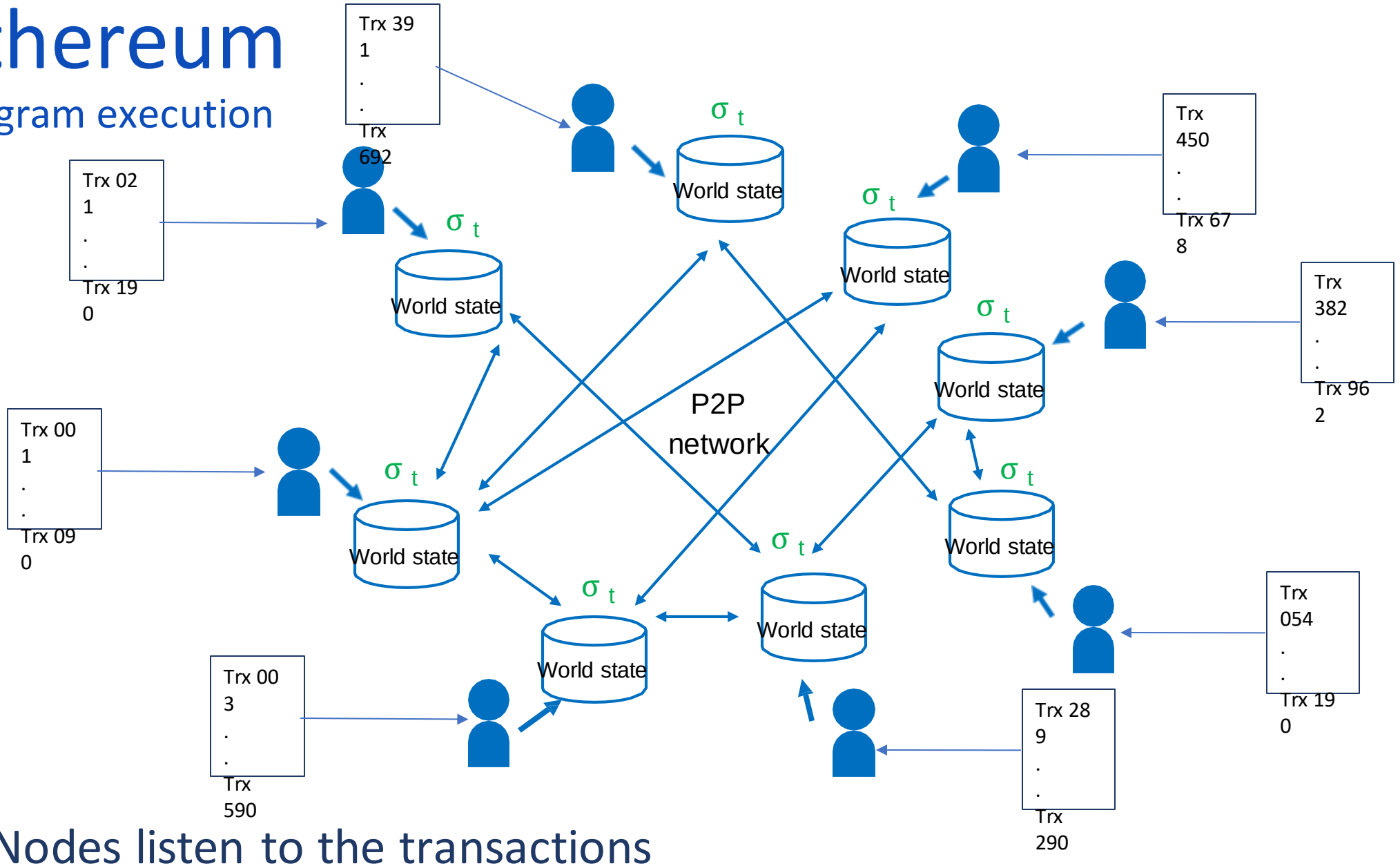
Program execution



- At time=t, all the nodes have the same world state

Ethereum

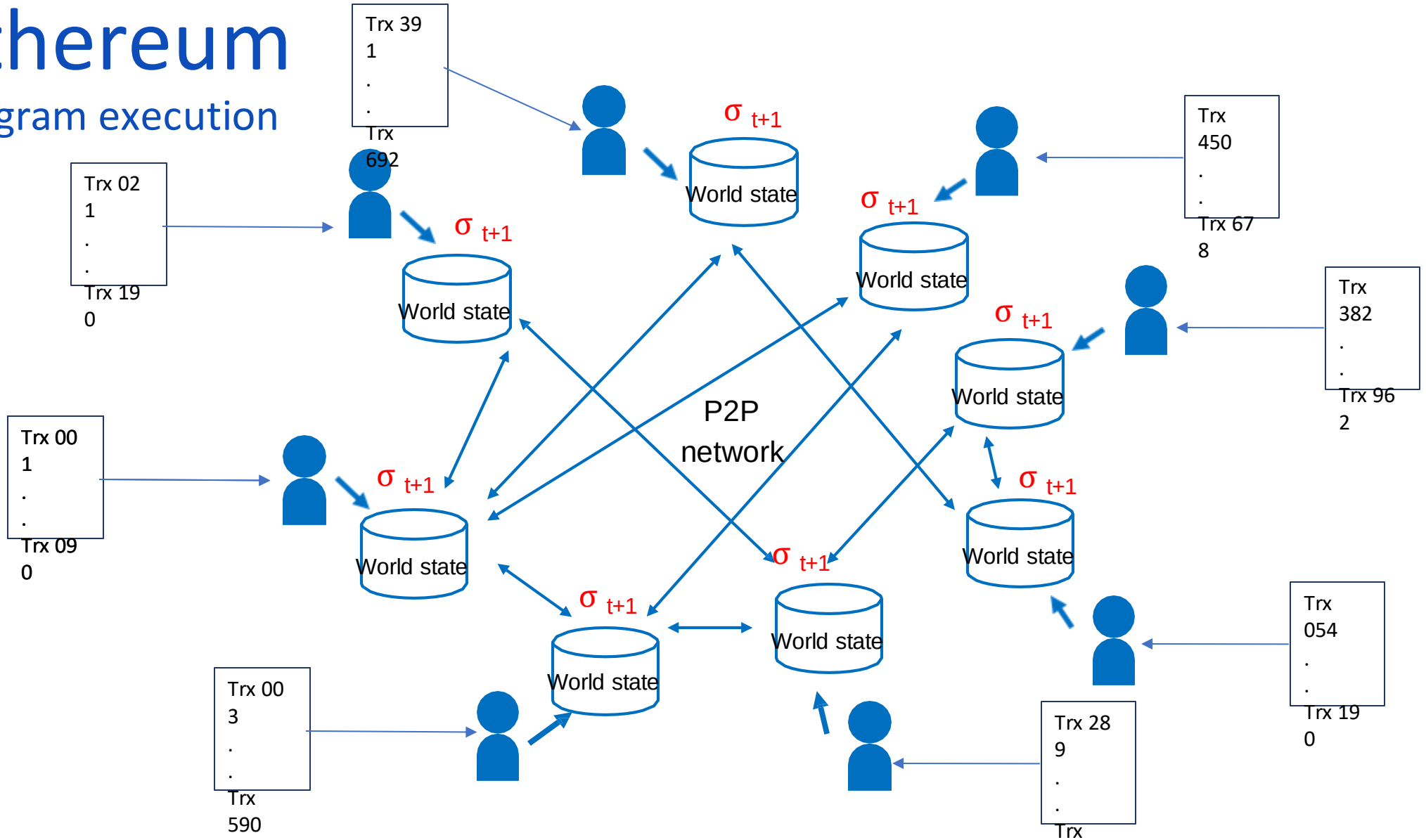
Program execution



- Nodes listen to the transactions
- And executes them

Ethereum

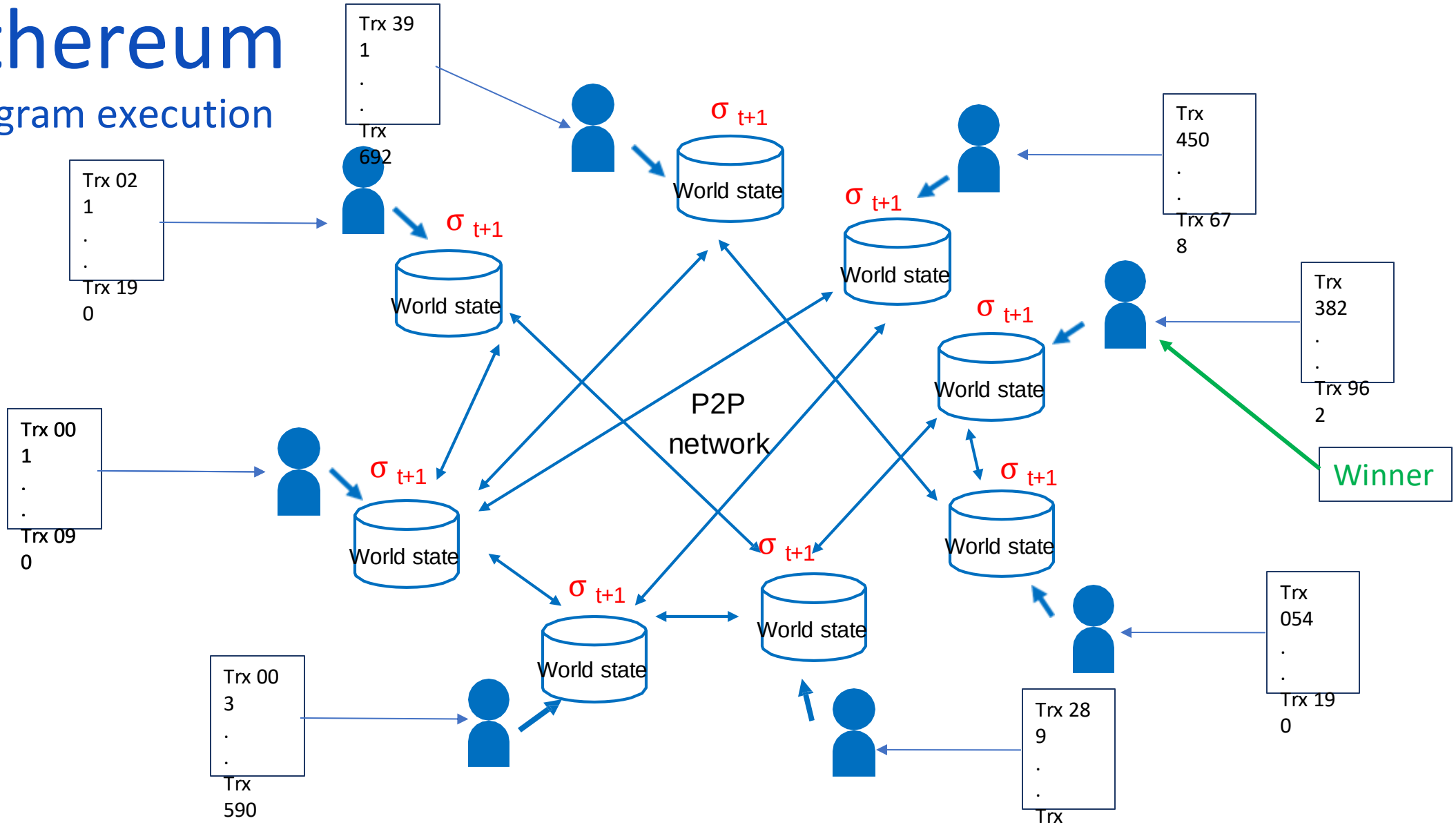
Program execution



- Everybody executes the transactions and updates their local states
- Mining process begins

Ethereum

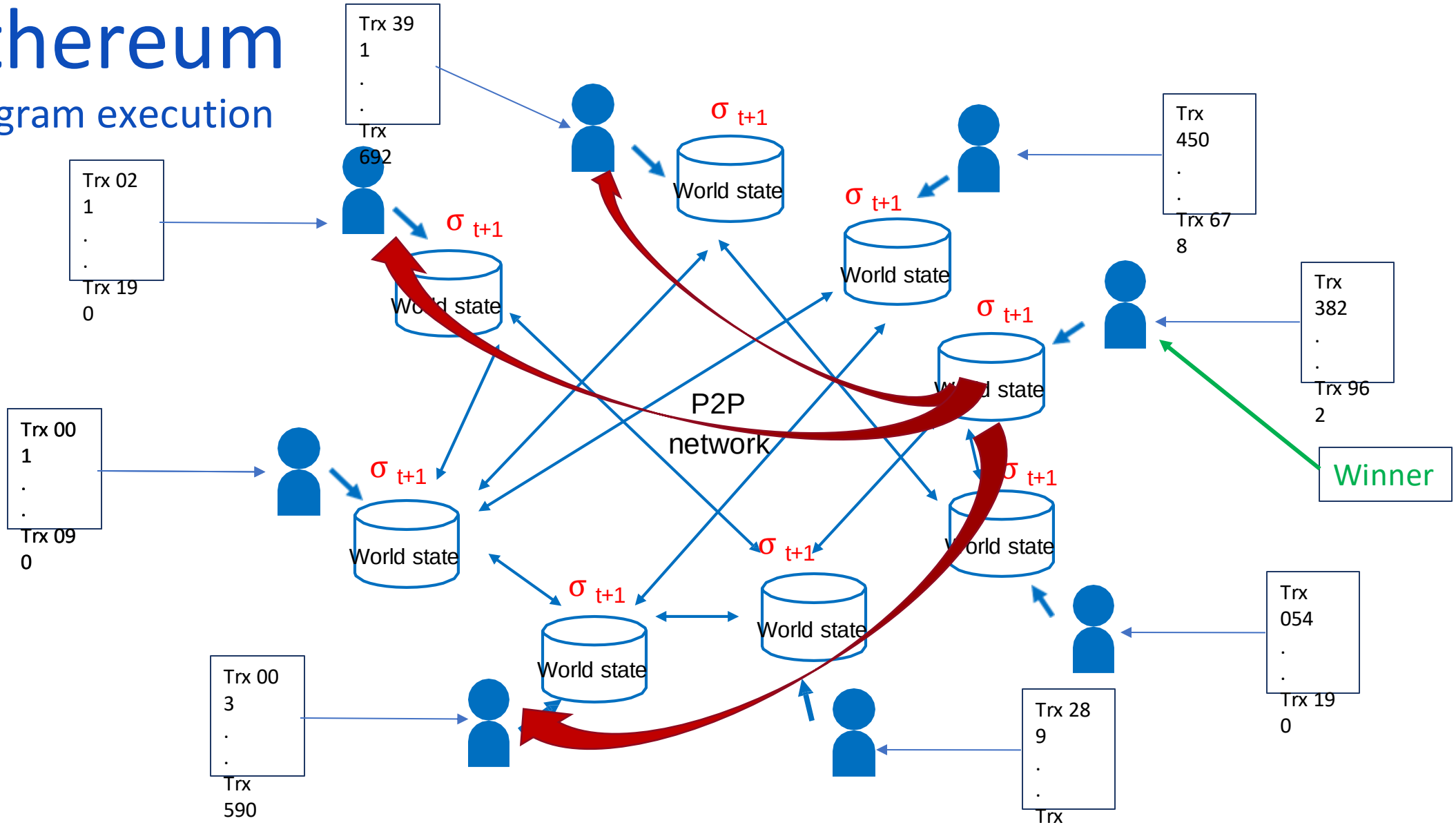
Program execution



- A winner emerges

Ethereum

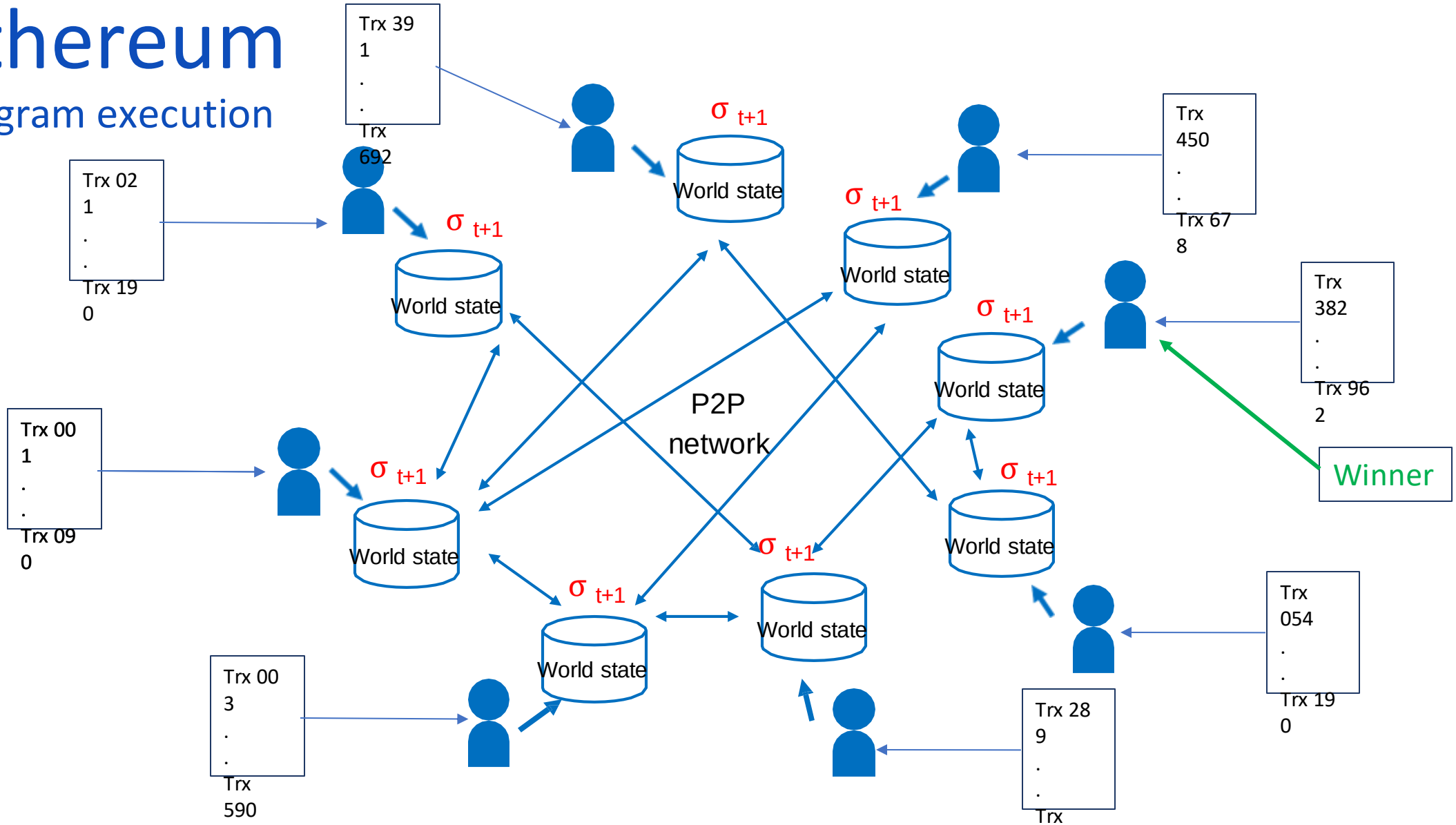
Program execution



- Winner's block is propagated to all the other nodes

Ethereum

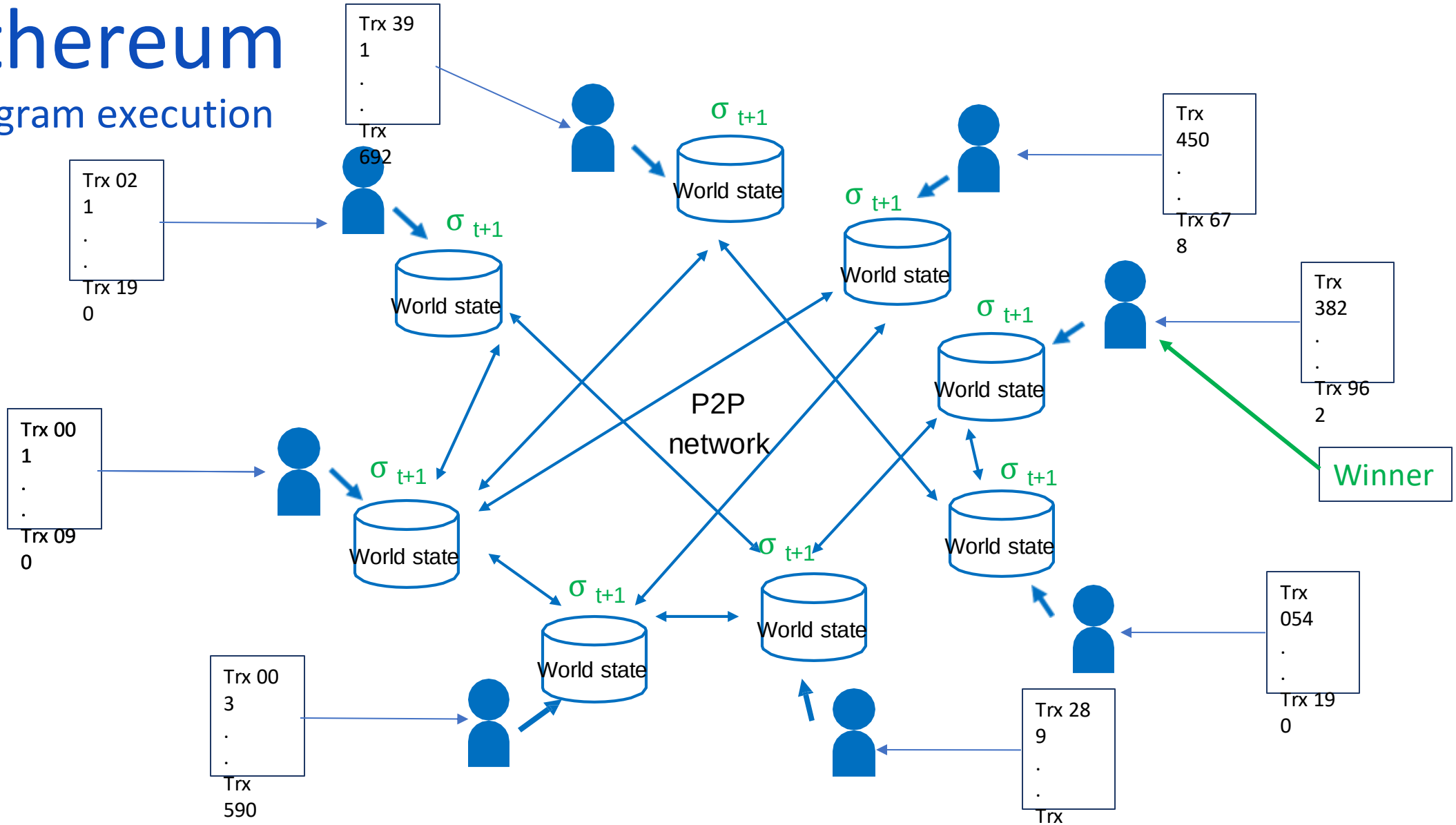
Program execution



- All the other nodes execute the transactions in the block

Ethereum

Program execution



- If it is consistent, then all the nodes update their states
- And the whole network reaches to a new global state

Ethereum

Program execution

- Smart contracts can read data from the Ethereum blockchain
- They can also access info on transactions in older blocks
- What about outside data?
  - Weather information, stock prices
- Oracle function call provides data of outside world in a reliable way
- Although real-world values are not deterministic
  - Oracles always answer each nodes query about what happened in reliable manner

Ethereum

Networks

- On the MainNet, data on the chain, account balances, transactions are public
- Anyone can create a node and begin verifying transactions
- Ethers on this network has a market value
- There are test networks to run a smart contract before real deployment
  - Ropstein, Kovan, Rinkeby, Gorli

The End !!!