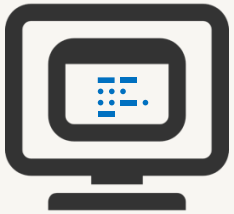
CSE446: Blockchain & Cryptocurrencies

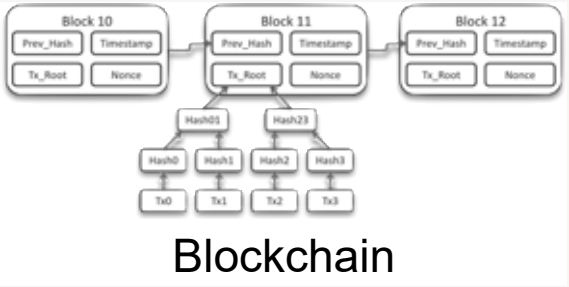
Lecture – 13: Ethereum - 2

Ethereum components

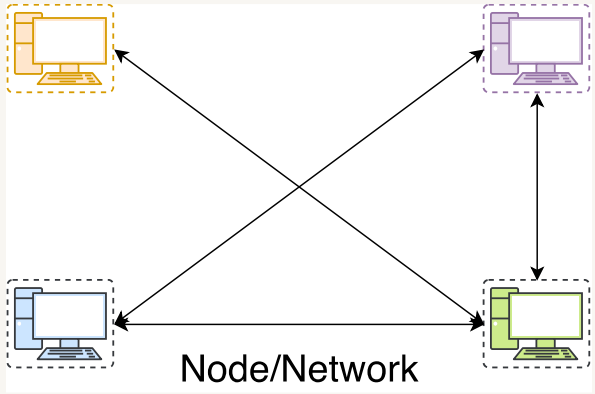
Users & code representation

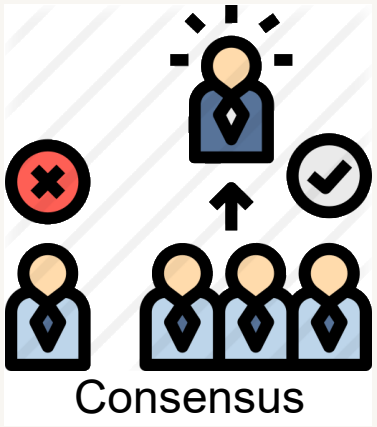


EVM



Blockchain





Consensus

Ethereum accounts

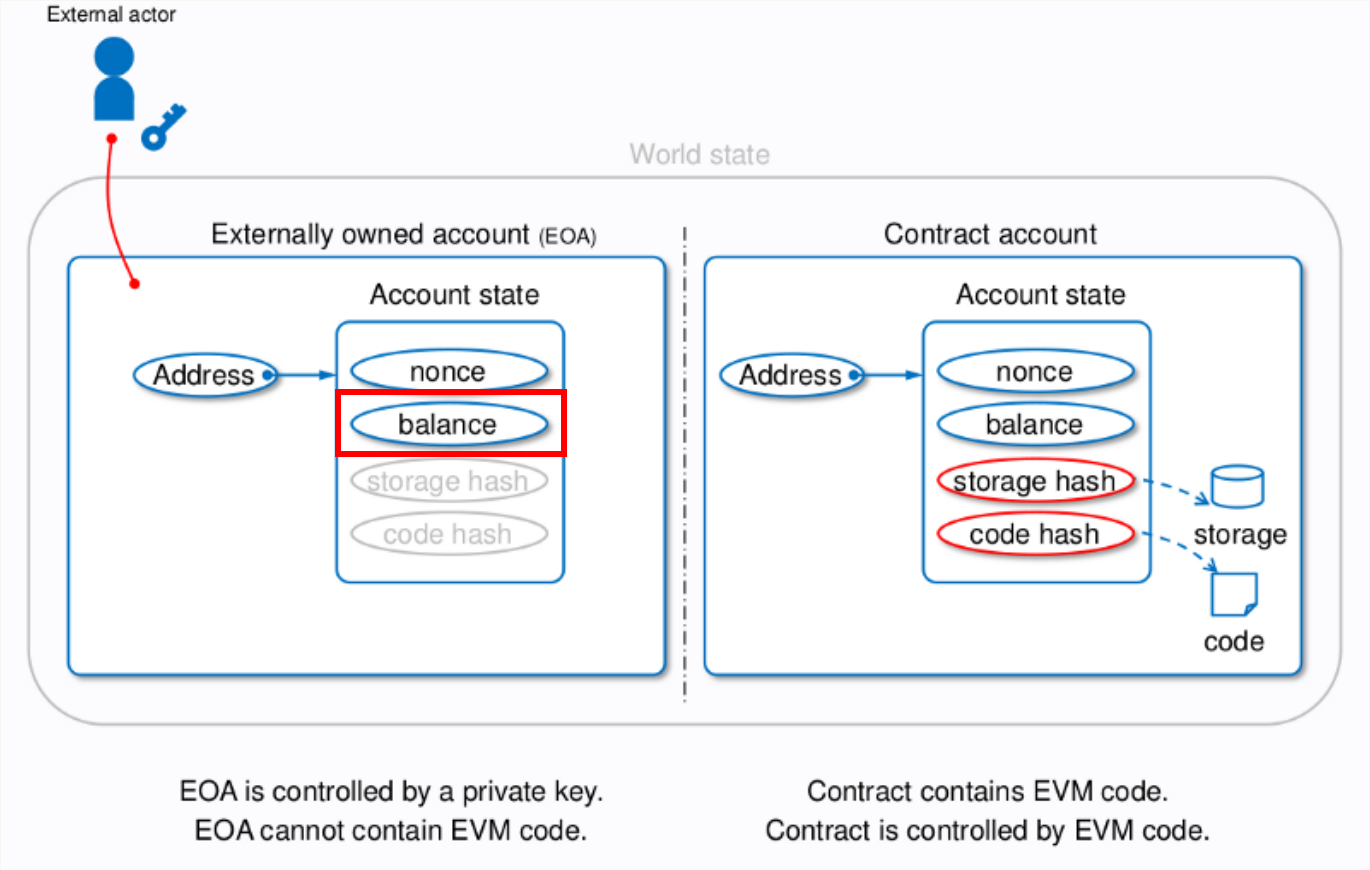
- Compared to Bitcoin, Ethereum uses an account-based ledger
  - Remember Bitcoin uses a transaction-based ledger
- Each distinct address represents a separate, unique account
- Accounts that are controlled by private keys and owned externally are EOA accounts
- EOA accounts do not have any code stored on the blockchain
  - This type can be seen as the default wallet of a user
- It can sign transactions, issue smart-contract functions calls and send Ether from one account to another
- The origin of any transaction is always an account controlled by a private key

Ethereum accounts

- Smart-contract accounts are controlled by their code
- Smart-contracts are treated as account entities with their own, unique address
- Such contract accounts don’t have any private key associated with it
- They can receive and send Ether just like an EOA
- Contracts can send messages to other accounts, both externally controlled accounts and smart contract accounts
- They can be invoked only by a transaction (similar to calling a function of a programming language program)
  - Either by an EOA or by another contract
- They have a persistent internal storage to write and read data from

Ethereum accounts

- Every EOA has its balance stored in the blockchain
- This is unlike Bitcoin where no separate balance database is maintained

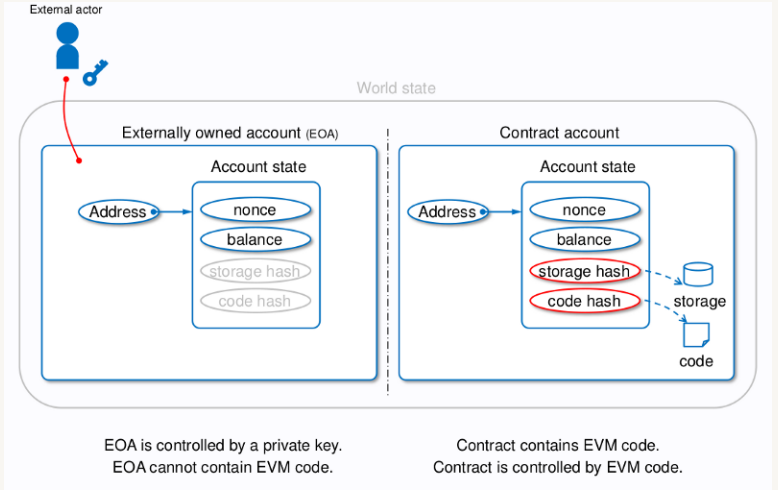


Here, world state represents the blockchain

https://cdn-images-1.medium.com/max/800/1*YC5PFXSJIZPw6zQWOuE7WQ.png

Ethereum accounts

- nonce: # of transactions sent / # of contracts created
- balance: # Wei owned (1 ether = 10^{18} Wei)
- storageRoot: Hash of the root node of a *Merkle Patricia* tree (MPT) of storage contents of the account
  - MPT will be discussed later
  - The tree is empty by default
- codeHash: Hash of the EVM (Ethereum Virtual Machine) code of this account



https://cdn-images-1.medium.com/max/800/1*YC5PFXSJlZPw6zQWOuE7WQ.png

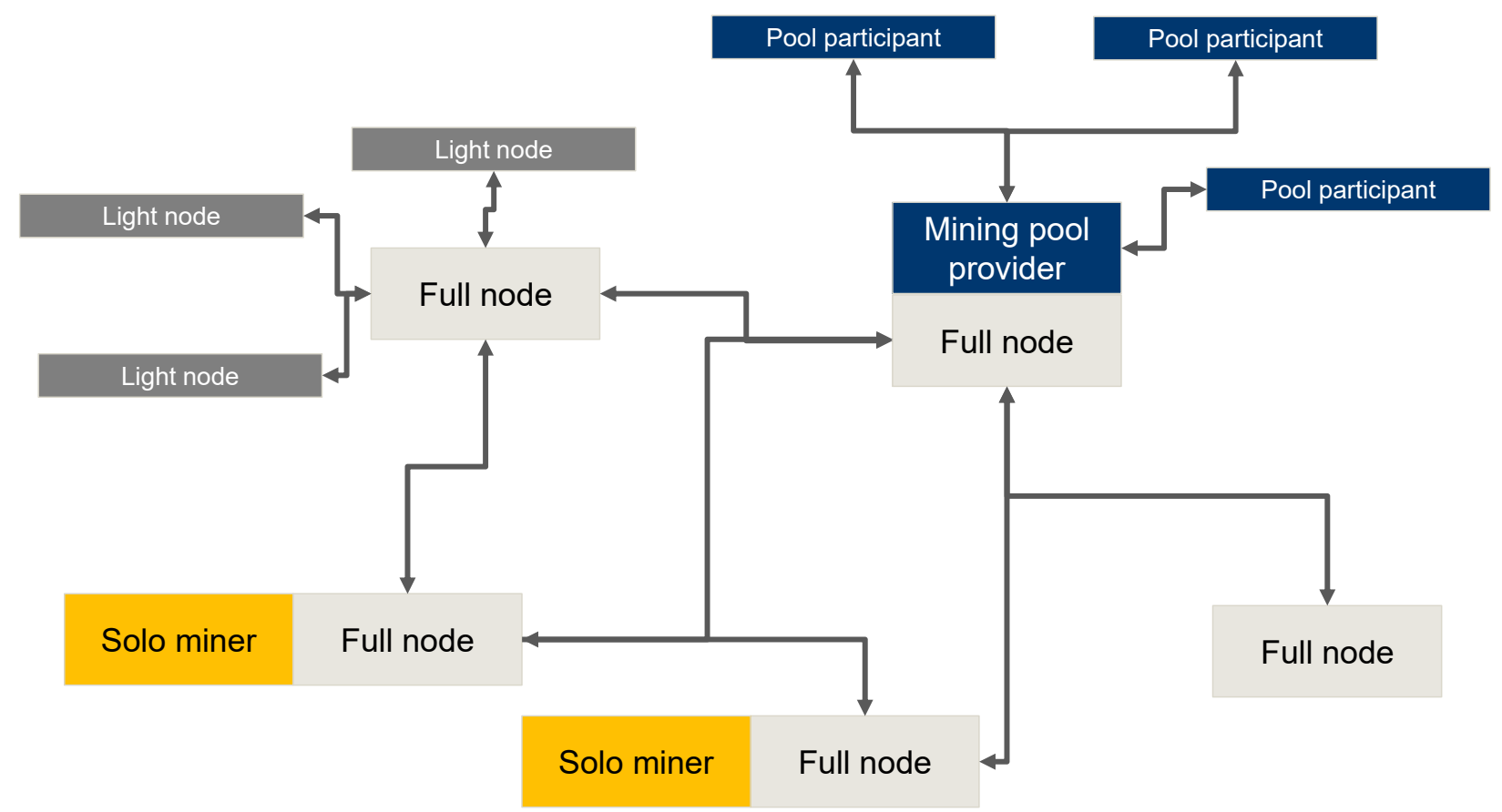
Ether denomination

Babbage, Lovelace and others represent influential CS/blockchain people

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

http://ethdocs.org/en/latest/ether.html

Ethereum network



Ethereum network node types

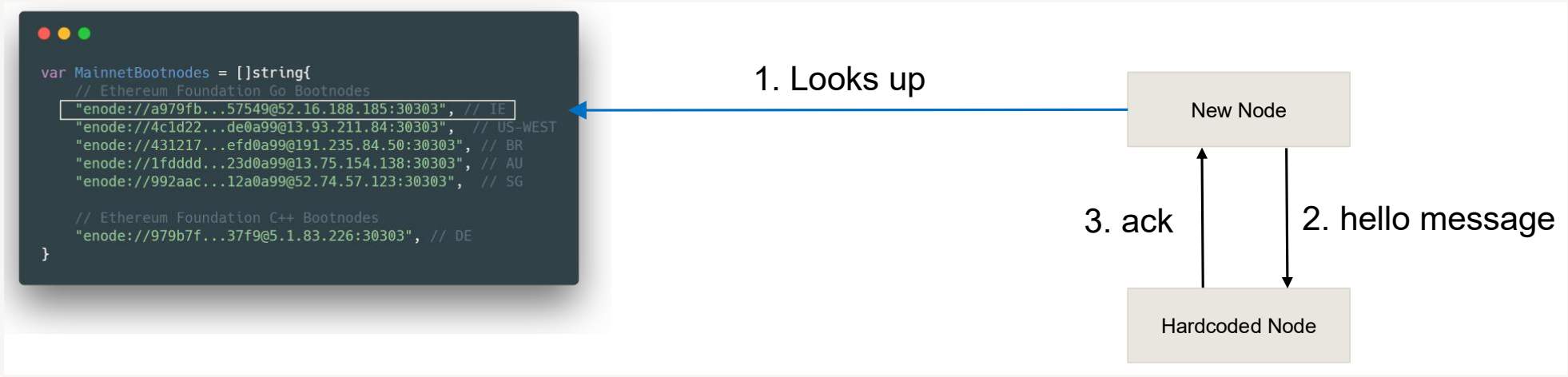
- Full nodes are the foundation of the Ethereum network
- Each full node holds a copy of the entire blockchain and syncs it with other nodes
- Transactions must be sent to a full node which distributes it among the network participants
- A light node is a client that is connected to a full node for the sake of not having to sync and download the entire blockchain
- For most private people, light nodes are the most comfortable way of interacting with the Ethereum blockchain
- One of the most common light nodes is https://myetherwallet.com (always triple check the domain)

Ethereum implementation

- To connect to the Ethereum network, you will need to download an Ethereum implementation
- Not all Ethereum nodes are using the same code base
- Since the specification for an Ethereum node is open source, basically anyone could create a different implementation
- The two major Ethereum implementations are: Geth and OpenEthereum

Node discovery

- Both Geth and OpenEthereum have a maintained list of default peers hardcoded into their source code
- Otherwise, it would be possible that no nodes are found, and the sync will always fail
- The Geth client comes with 6 hardcoded peers
- Once a node is selected, a hello message is sent to make an initial connection with the node



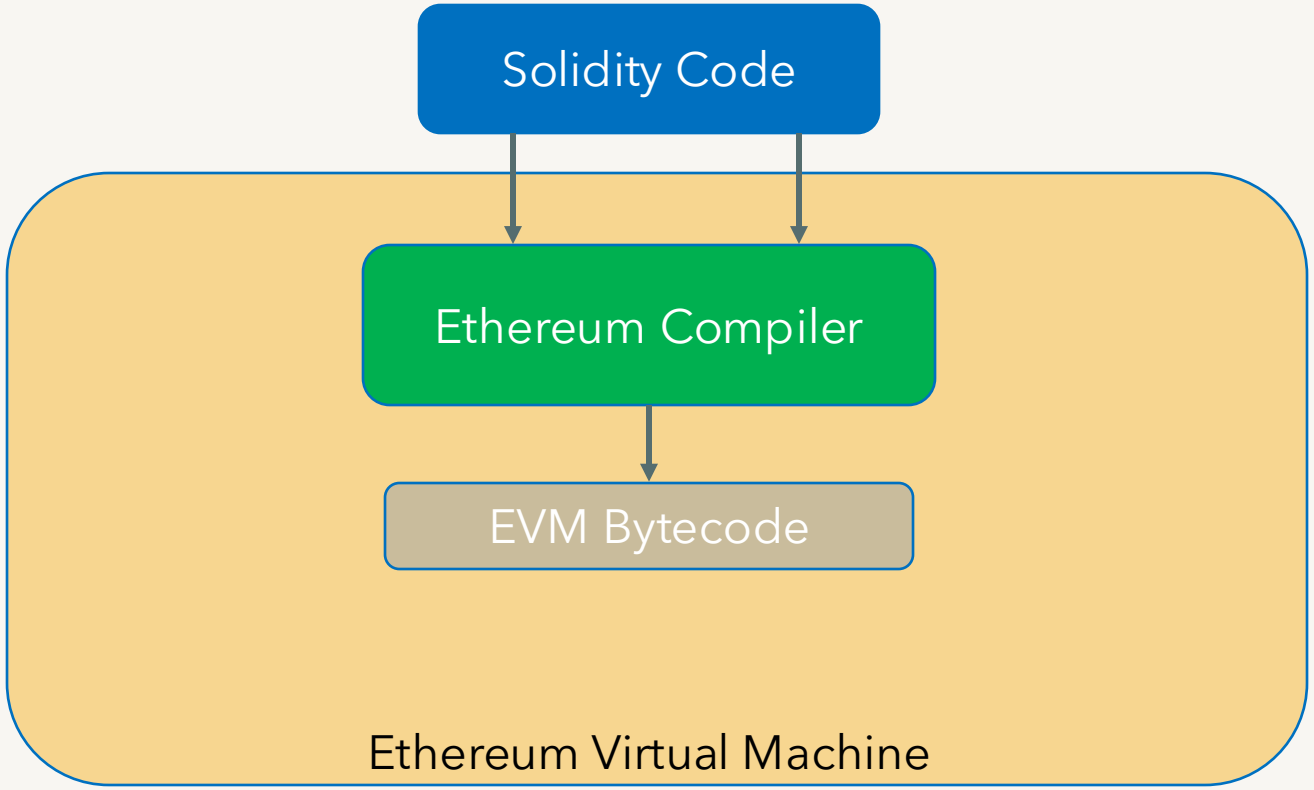
Ethereum Virtual Machine (EVM)

- The Ethereum Virtual Machine (EVM) is a simple but powerful, Turing complete 256bit Virtual Machine that allows anyone to execute arbitrary EVM Byte Code
- Turing completeness means you can write programs (contracts) that can (for the most part) solve any reasonable computational problem or write sophisticated logic
- It allows anyone to execute arbitrary code in a trust-less environment in which the outcome of an execution can be guaranteed and is fully deterministic

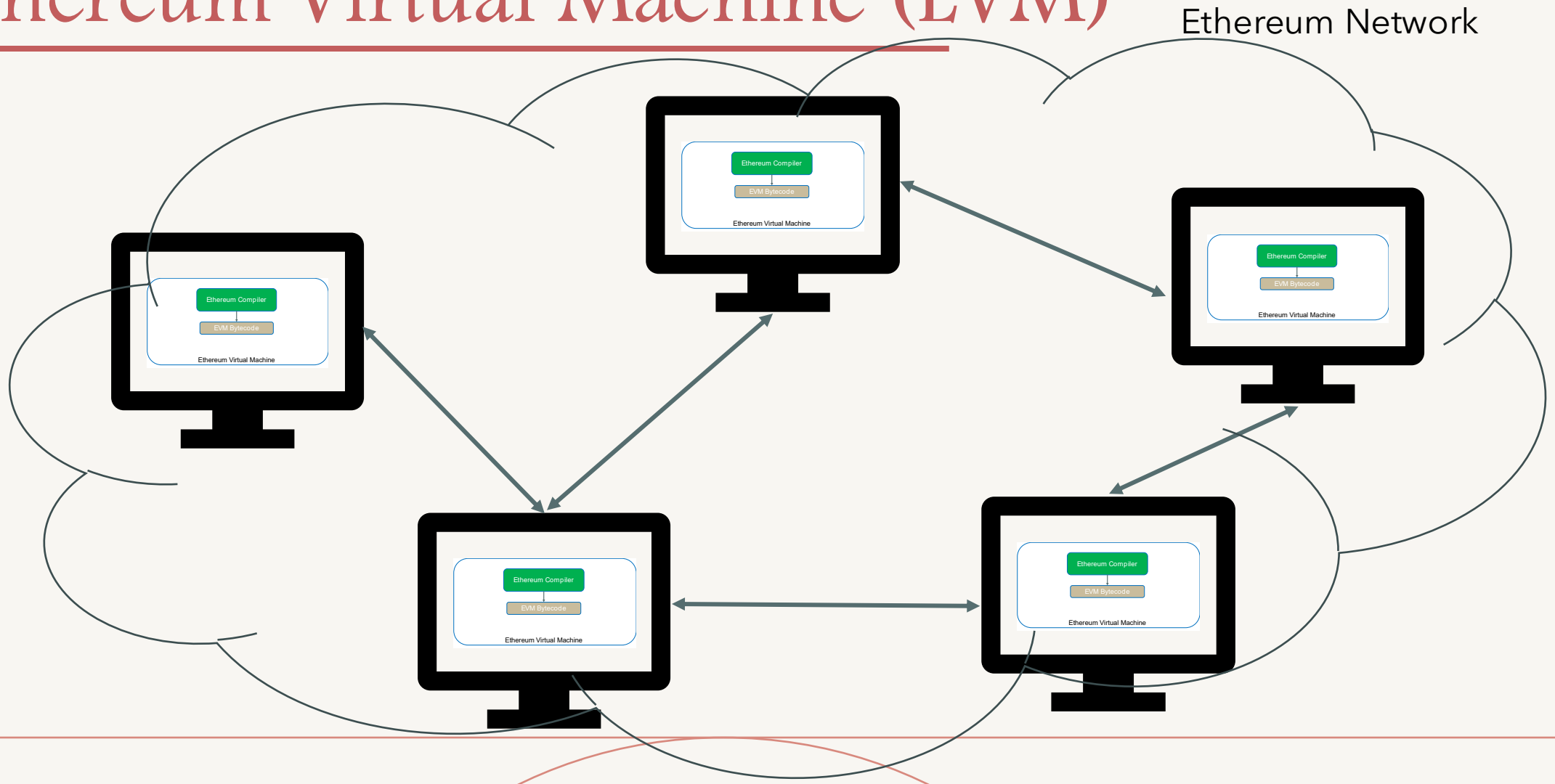
Ethereum Virtual Machine (EVM)

- Smart contract code is usually written in a high level programming language such as Solidity
- This code gets compiled to something called the EVM bytecode which gets deployed to the Ethereum blockchain
- This is very similar to a programming language like Java where the code gets converted to JVM Byte code
- The Ethereum runtime environment only understands and can execute the bytecode

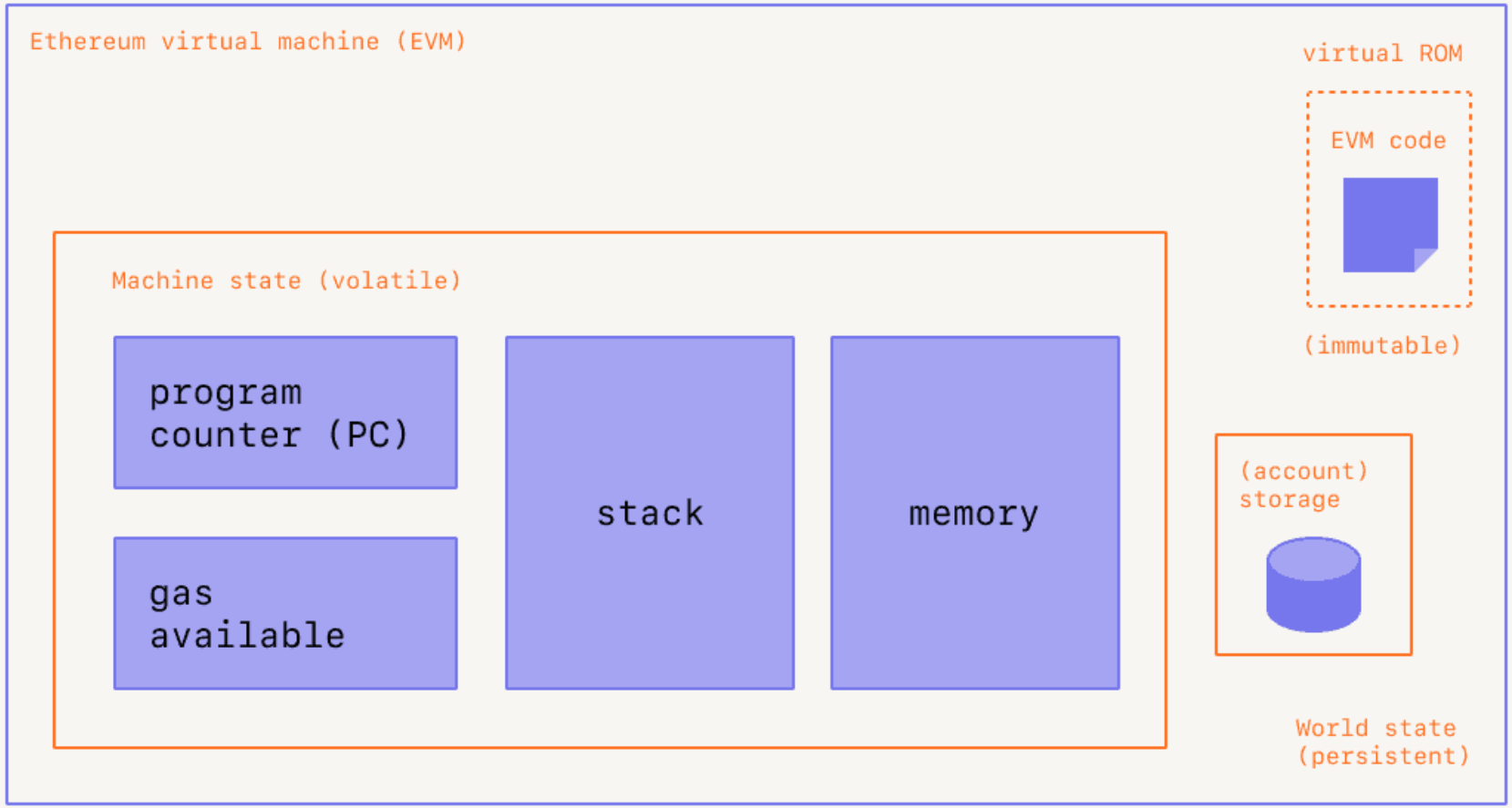
Ethereum Virtual Machine (EVM)



Ethereum Virtual Machine (EVM)

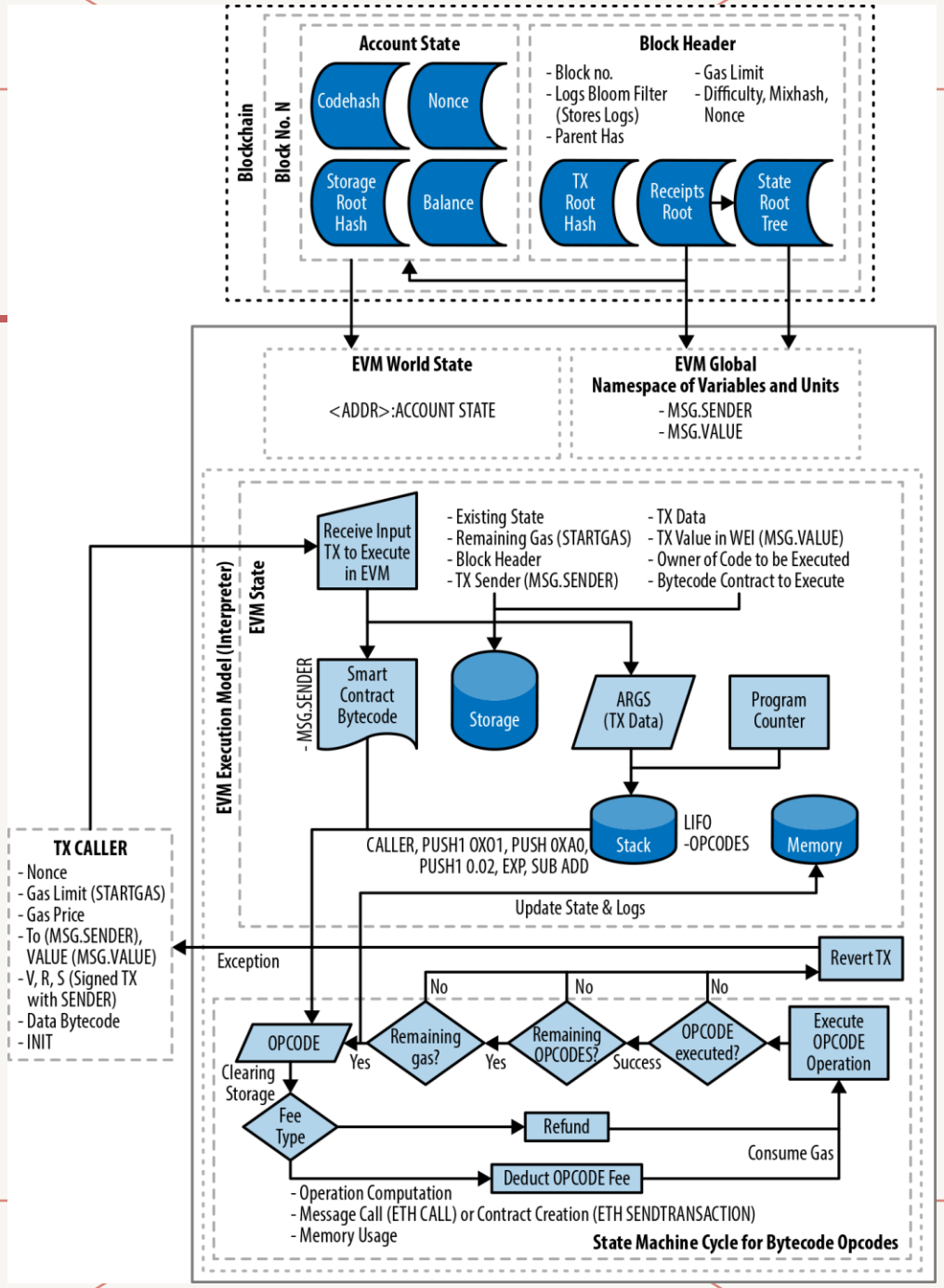


Ethereum Virtual Machine (EVM)



https://ethereum.org/static/e8aca8381c7b3b40c44bf8882d4ab930/302a4/evm.png

EVM



https://cypherpunks-core.github.io/ethereumbook/13evm.html

EVM Bytecode

- EVM bytecode is the instruction sets for the EVM
- The EVM instruction set offers most of the operations you might expect, including
  - Arithmetic and bitwise logic operations
  - Stack, memory, and storage access
  - Control flow operations
  - Logging, calling, and other operators
  - Access to account information (e.g., address and balance) and block information (e.g., block number and current gas price) (Blockchain state information)

https://cypherpunks-core.github.io/ethereumbook/13evm.html

Ethereum blockchain

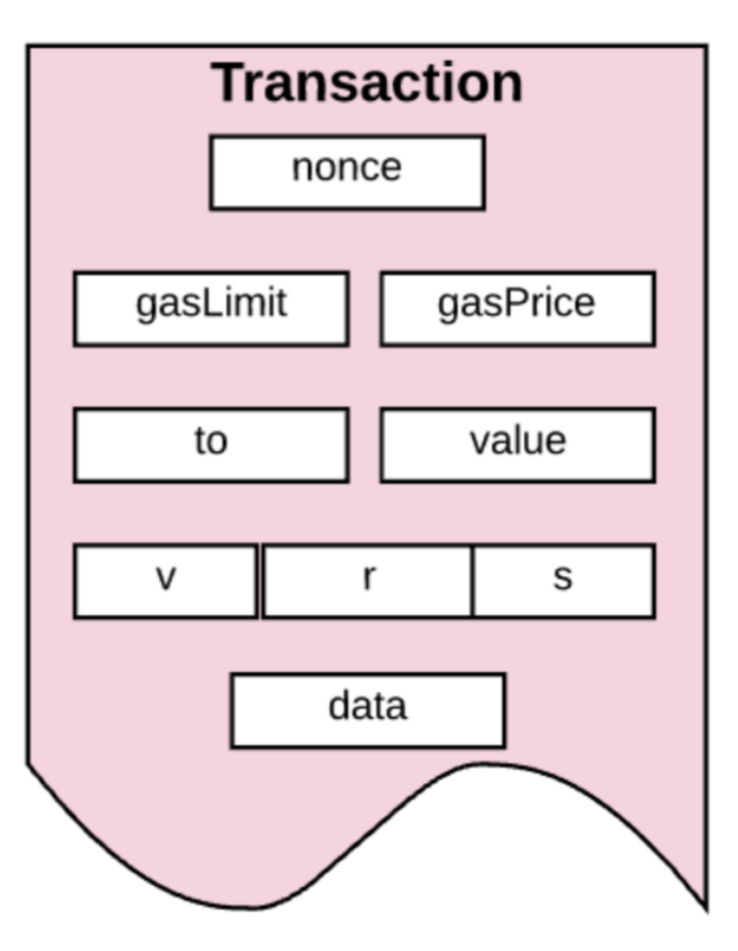
- Like Bitcoin, there are three major sub-components in the Ethereum blockchain
  - Transactions
  - Blocks
  - Blockchain

Ethereum transactions

- On Ethereum there are mainly two transactions
  - External transactions
  - Internal transactions
- An external transaction
  - is generated by a user, serialised and sent to the Ethereum network, put on the blockchain (just like Bitcoin)
  - Includes payment (regular) transactions and contract creation & contract call (execution) transactions

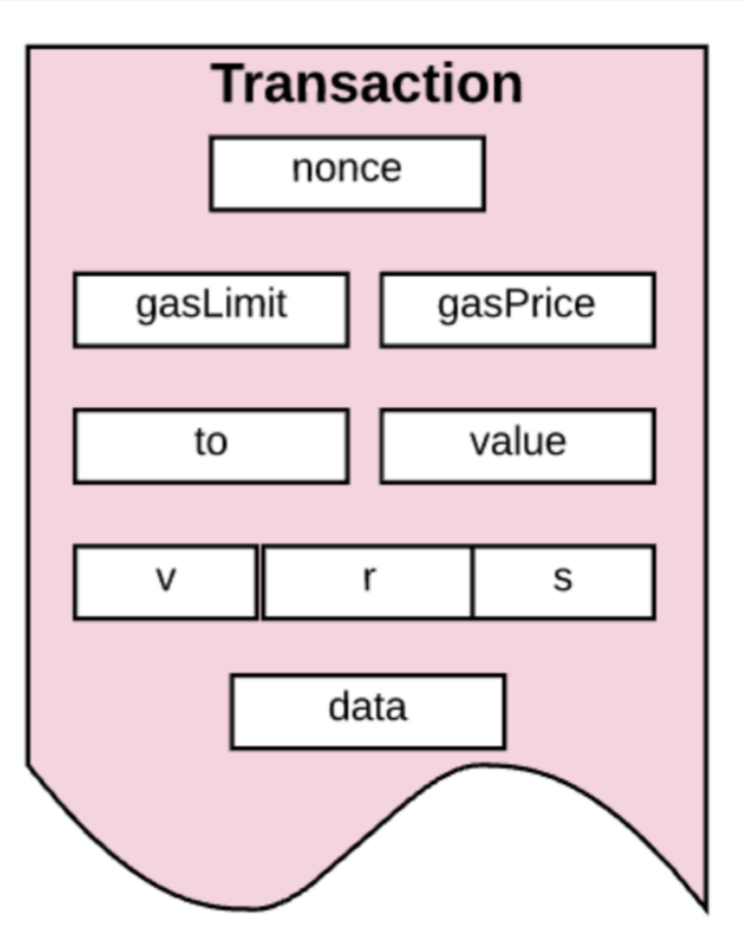
Ethereum external transactions: structure

- nonce: A count of the number of transactions sent by the sender (EOA) number, used to prevent message replay
- gasPrice: The price of gas (in wei) the originator (sender) is willing to pay
- gasLimit: The maximum amount of gas the originator is willing to buy for this transaction



Ethereum transactions

- to: Recipient's/contract address
- value: Amount of Wei transferred from the sender to the receiver
- v,r,s: The three components of an ECDSA digital signature of the originating EOA
- data: optional field to include code to create a contract or data for calling a function of an existing contract



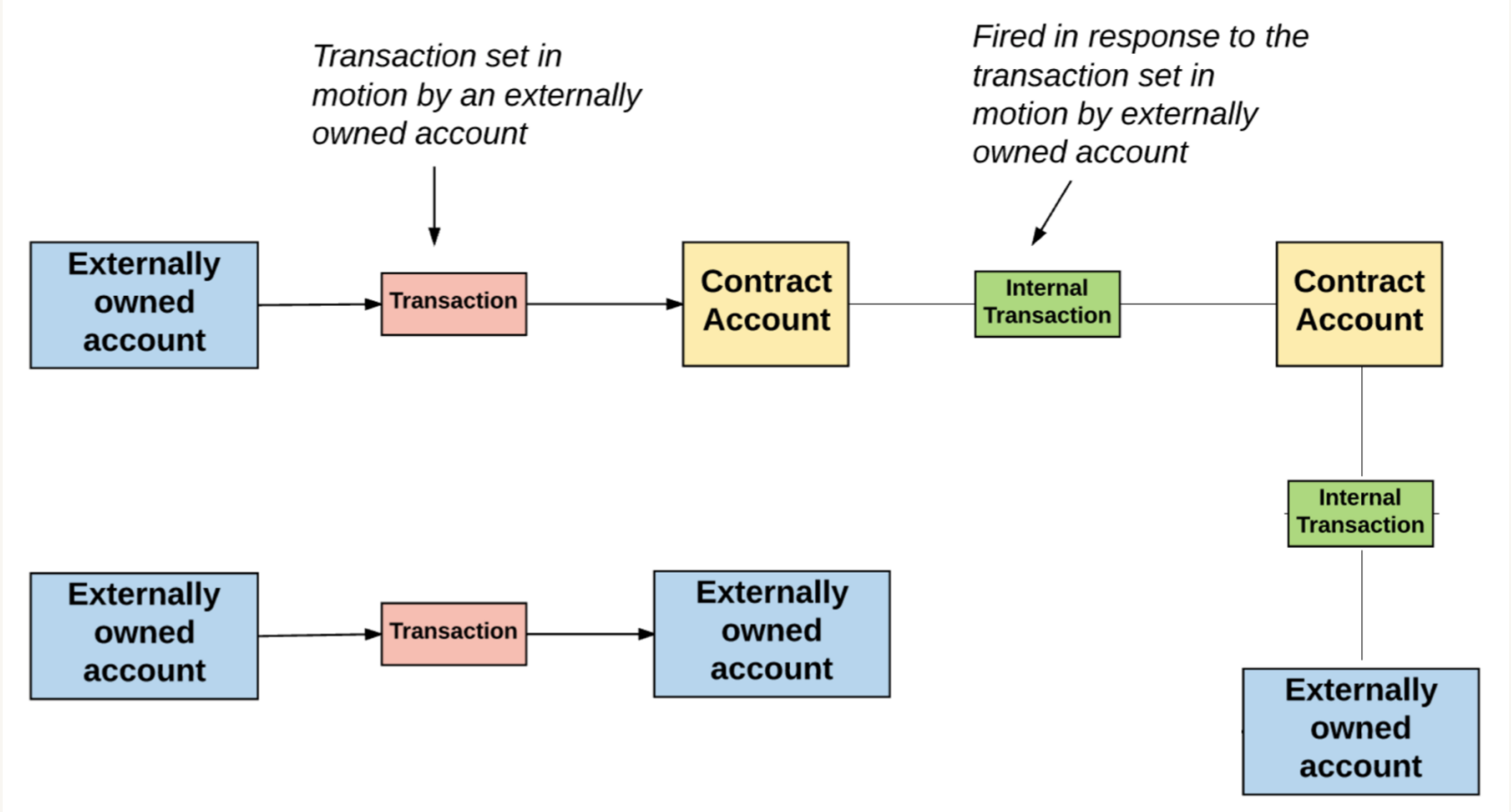
Ethereum external transactions

- Regular transactions
  - a transaction from one account to another (either to an EOA or to a contract account)
  - A 'to' address in this transaction denotes to receiver’s address
- Contract deployment transactions:
  - to deploy a smart-contract
  - a transaction without a 'to' address, where the data field is used for the contract code
- Execution of a contract:
  - a transaction that interacts with a deployed smart contract
  - In this case, 'to' address is the smart contract address
  - Data field contains the relevant value for the contract (function)

Ethereum transactions types

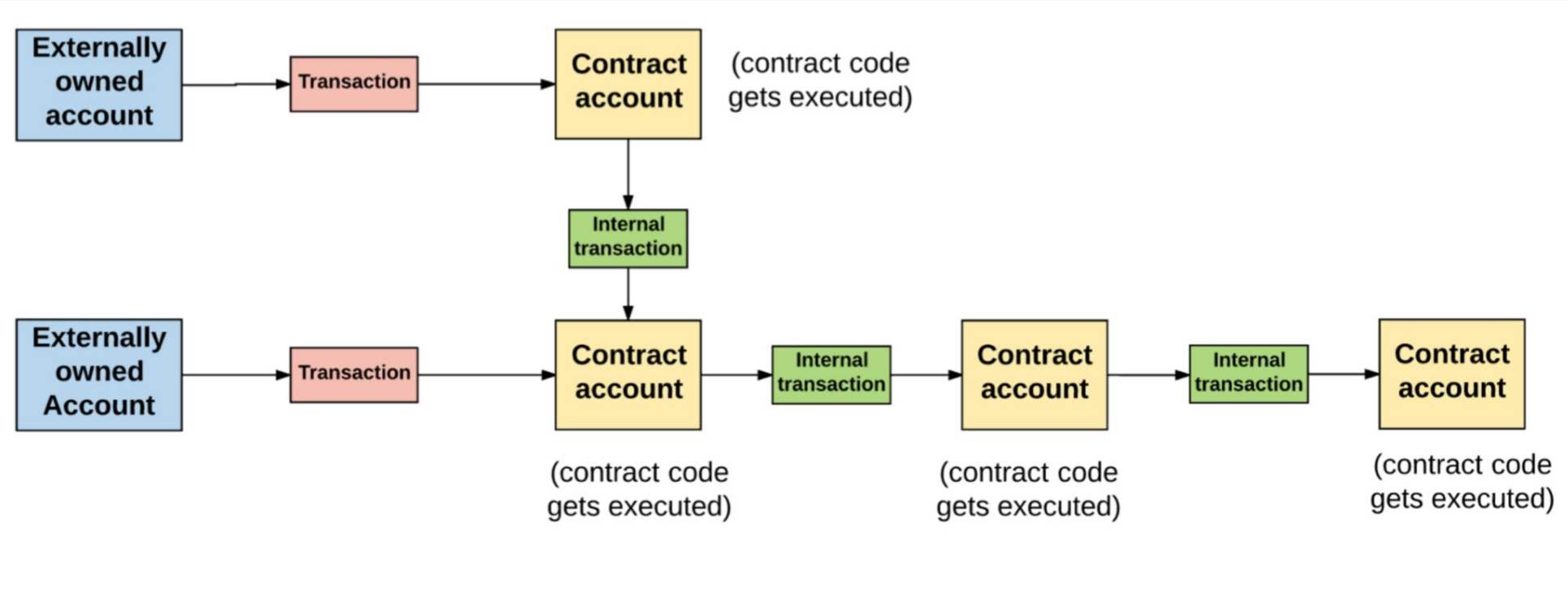
- Internal transactions
  - Contracts A can call a function in Contract B
  - These are not serialised and not put on the blockchain!
  - When these transactions transfer value (Eth) to another account, the account update is stored in the blockchain

Ethereum transactions types



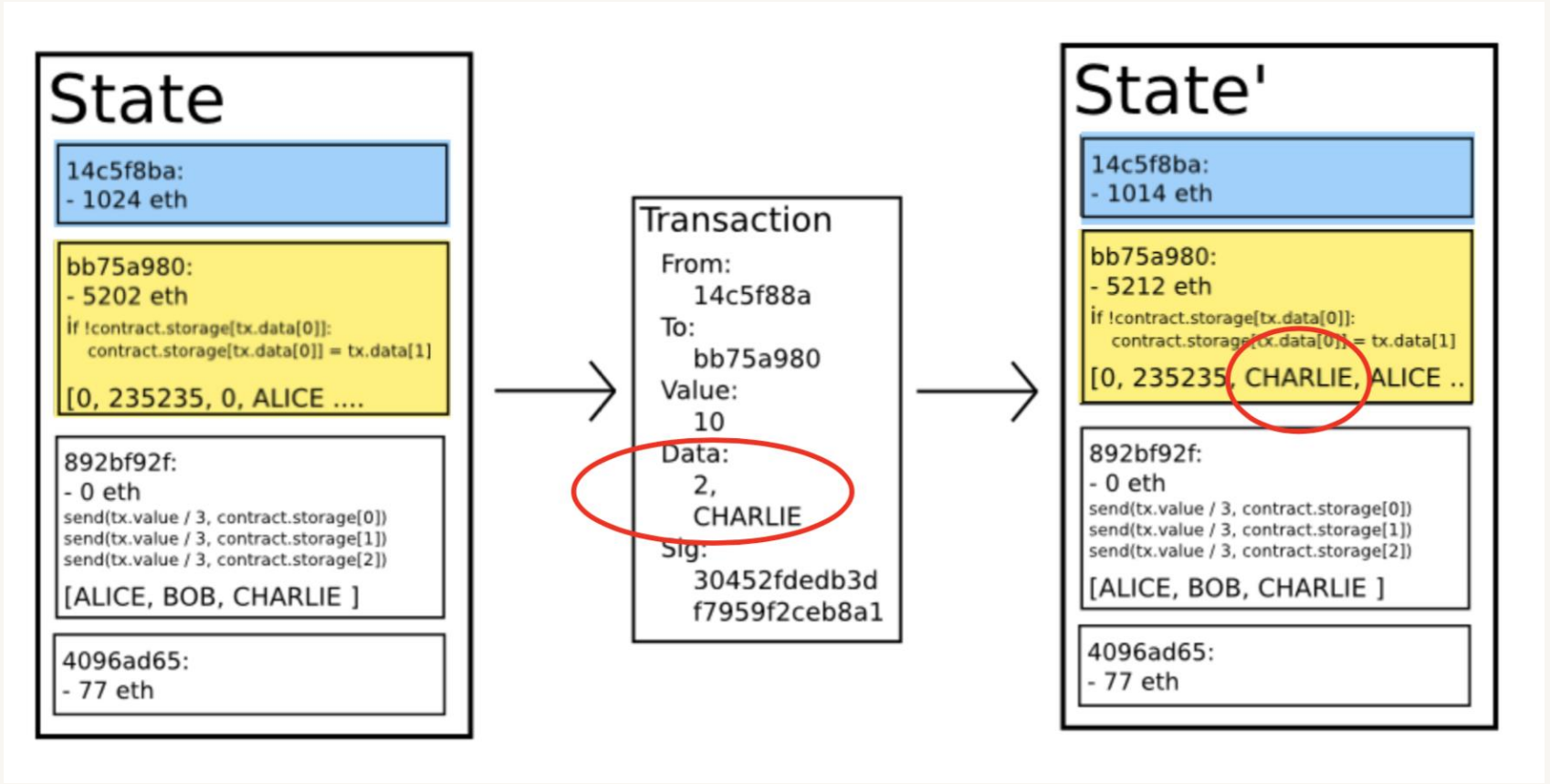
From Lecture Slides of Blockchains & Cryptocurrencies Course at Johns Hopkins University

Ethereum transactions types



From Lecture Slides of Blockchains & Cryptocurrencies Course at Johns Hopkins University

Ethereum transaction execution



From Lecture Slides of Blockchains & Cryptocurrencies Course at Johns Hopkins University

Ethereum DoS protection

- EVM code is “Turing complete", e.g., has loops and may not halt
- In order to ensure safety, a type of DoS protection, the concept of fee is used
- Ethereum utilises a “pay as you execute” model
  - Meaning users need to pay for both computation and storage of data
- This payment is collected by the respective miner

Ethereum gas and gas price

- Each transaction contains a “gas limit”, and a “gas price” which is the price the sender will pay per unit of gas, denominated in ETH (currency)
- Along with sufficient ETH to pay the fee
- But, how much do you need to pay?
- It depends on how much unit of work, i.e. code execution, is carried out or the amount of data stored in the network
  - The unit of work in Ethereum is defined with the concept of Gas
  - Ether buys GAS to fuel the EVM, like hour(s) of labour

Ethereum gas and gas price

- Each operation in Ethereum utilises some amount of gas
- For example:
  - If you add two numbers, 3 gas is used
  - If you multiply two numbers, 5 gas is used
  - One hash call requires 30 gas
- Ethereum also uses gas for contract creation
- By calculating total operations, we can estimate how much gas it might require
- It is difficult to know the exact gas required for contract execution
  - So, set a higher limit using the notion of gasLimit, that implies the maximum amount of gas you are willing to buy

Ethereum gas and gas price

- But, still we don't know how much Ether we have to pay to the miners
- That is determined by Gas Price, which is like the hourly labour rate: 500 taka/hour
- Fee for computation in Ether = gasLimit x gasPrice;
  - gasLimit (gas) and gasPrice are specified in the transaction
  - The calculated Fee is placed in Escrow
- Gas price is determined by the sender
- If gasPrice is too low, no miner will bother about processing a transaction

Ethereum gas and gas price

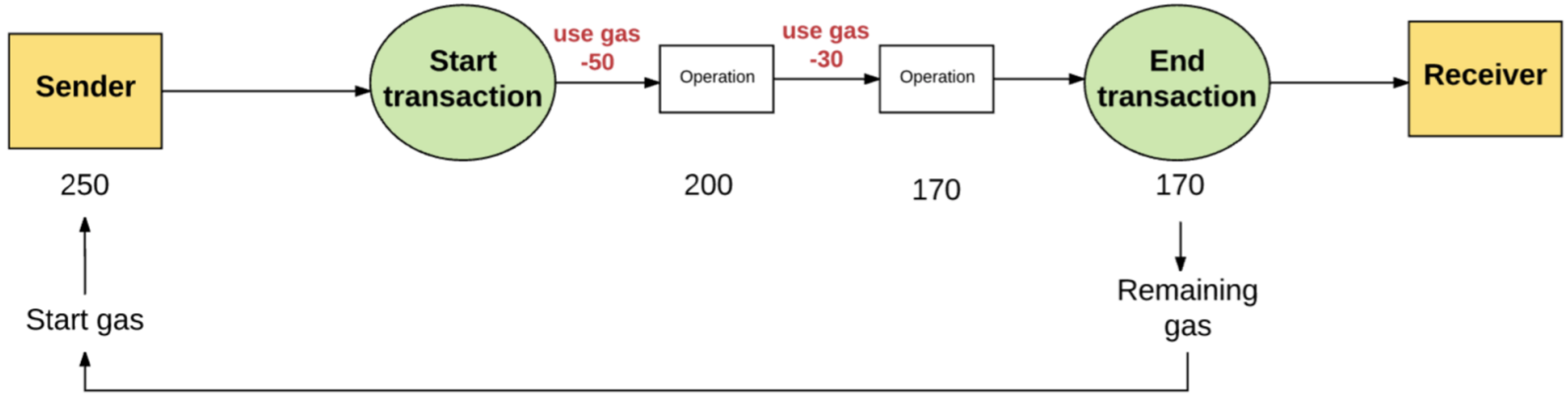
- If the fee is less than required, gas runs out before a transaction is completed
  - State is reverted back to the previous state
  - Fee is consumed from the escrowed Ether
- If execution completes
  - consumed gas x gasPrice is paid to the miner
  - Remaining gas (gasLimit – consumed gas) x gasPrice is returned back to the caller
- The higher the gas price you set, the sooner your transaction gets mined
  - Most miners tend to choose transactions with higher fees
- If price is relatively low, your transaction will eventually get included in the block but you might have to wait for a bit

Ethereum gas and gas price

- Gas limit: Max no. of computational steps the transaction is allowed.
- Gas Price: Max fee the sender is willing to pay per computation step.

Ethereum gas and gas price

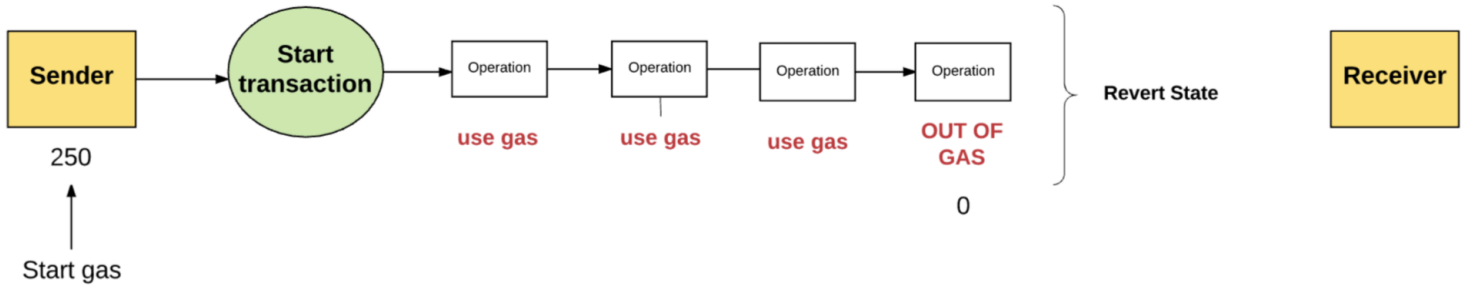
The sender is refunded for any unused gas at the end of the transaction.



From Lecture Slides of Blockchains & Cryptocurrencies Course at Johns Hopkins University

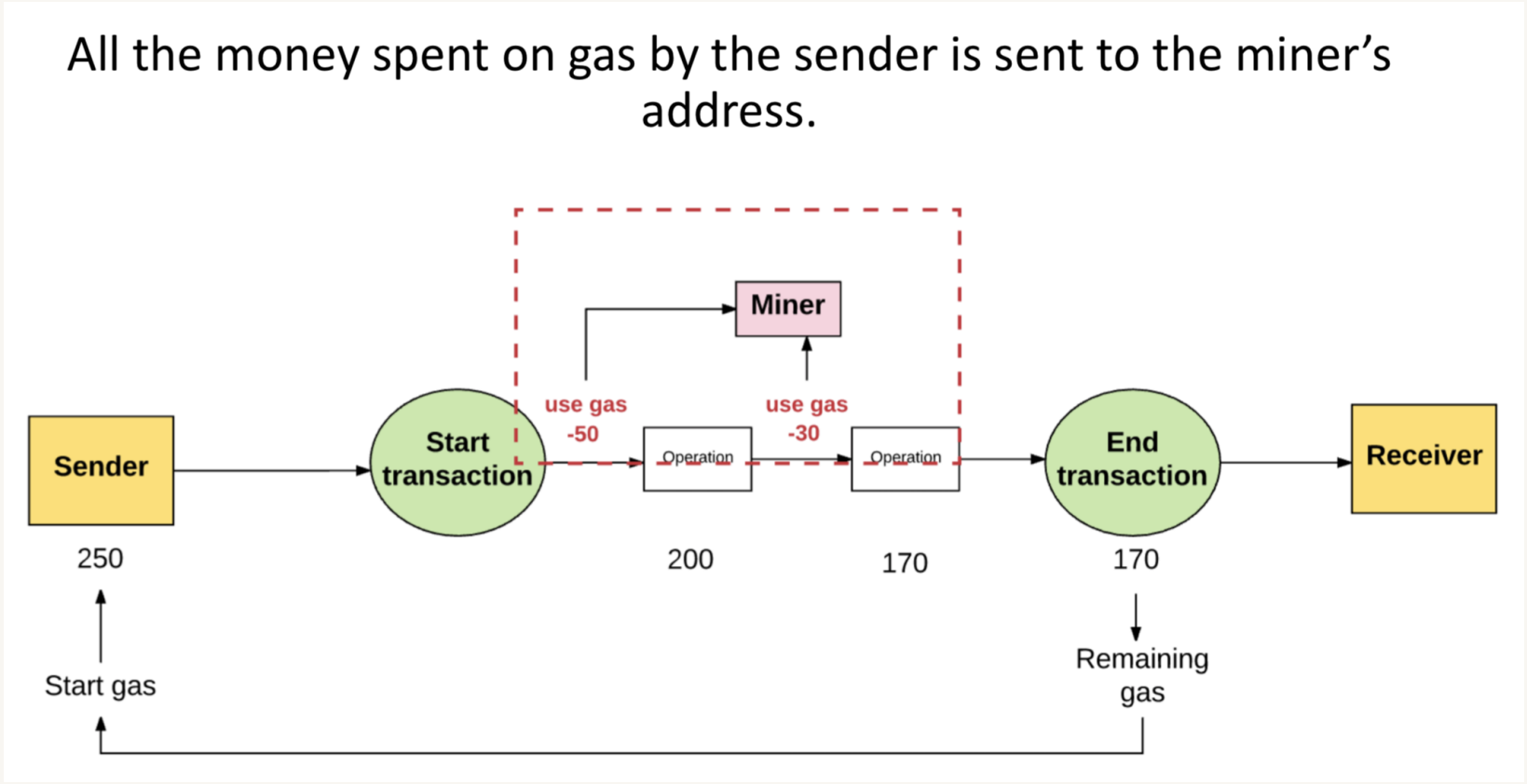
Ethereum gas and gas price

If sender does not provide the necessary gas to execute the transaction, the transaction runs “out of gas” and is considered invalid.



- The changes are reverted.
- None of the gas is refunded to the sender.

Ethereum gas and gas price



From Lecture Slides of Blockchains & Cryptocurrencies Course at Johns Hopkins University

Question?