

# Ethereum Basics

Blockchain-Based Systems Engineering

Chair of Software Engineering for Business Information Systems (sebis)  
Faculty of Informatics  
Technische Universität München  
[www.matthes.in.tum.de](http://www.matthes.in.tum.de)

## 1. Ecosystem

- Historical Overview
- Crowdsale Statistics
- Technical Papers
- Foundations
- Network Metrics

## 2. System Architecture

- Concept of a World Computer
- EVM
- Accounts
- Blockchain Properties
- Smart Contracts

## 3. Network Architecture

- Overview
- Node Types
- Clients
  - Geth
  - OpenEthereum





Vitalik Buterin at a Techcrunch conference

- In **November 2013**, Vitalik Buterin started working on the **first version** of the Ethereum **white paper**
- Buterin made the **first public announcement** of Ethereum on **24<sup>th</sup> of January 2014** at the Bitcoin conference in Miami
- On the **7<sup>th</sup> of July 2014**, Buterin **announced the start** of the **public crowd sale**
- The **sale lasted 42 days** until 2<sup>nd</sup> of September
  - For the first 14 days the price was 1 BTC for 2000 ETH
  - After that period the price went up to 1 BTC for 1337 ETH
- In total, **~60 million Ether** were sold in exchange for **31.591 Bitcoins**
  - Worth around 18.5 million USD at that time
  - Used by the Ethereum foundation


**"On that day I realized what horrors centralized services can bring..."**

Vitalik Buterin wrote on his website's bio that he was driven to create decentralized money because his World of Warcraft character was cruelly nerfed by Blizzard. He said "I saw everything to do with either government regulation or corporate control as just being plain evil. And I assumed that people in those institutions were kind of like Mr. Burns, sitting behind their desks saying, 'Excellent. How can I screw a thousand people over this time.'"



**VITALIK BUTERIN**

Zug, Switzerland

 Visit my website

#Copy-paste bio for conferences, articles, propaganda materials, etc:

<http://vitalik.ca/files/quickbio.txt>

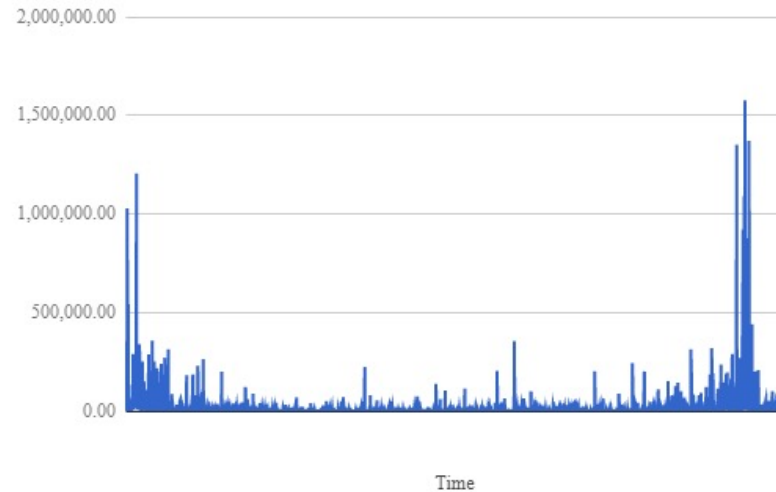
#More interesting bio

I was born in 1994 in Russia and moved to Canada in 2000, where I went to school. I happily played World of Warcraft during 2007-2010, but one day Blizzard removed the damage component from my beloved warlock's Siphon Life spell. I cried myself to sleep, and on that day I realized what horrors centralized services can bring. I soon decided to quit.

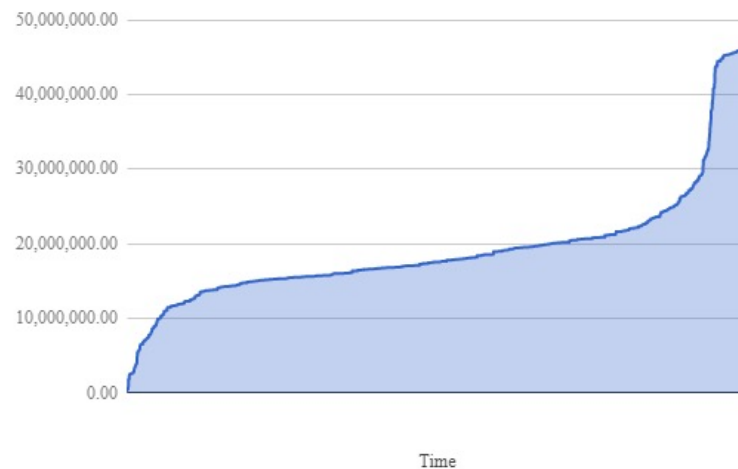
# Crowdsale Statistics

## First 14 Day Period

ETH sold



Cumulative ETH sold



- Around **48 million ETH** were **sold during the first price period** of 14 days
- **Most ETH** were **sold** at the **beginning** and the **end** of the period
- **Biggest single purchase** during the first period was **500 BTC** which equals **1.000.000 ETH**
- **Smallest purchase** was **0.01 BTC**
- **43.6%** bought **2000 or more ETH**
- **0.8%** bought **200.000 or more ETH**
- **11,901,464.23948 ETH** to the **development team** (<https://etherscan.io/address/0x5abfec25f74cd88437631a7731906932776356f9>)

## White Paper

Rootul Patel edited this page 2 days ago · 124 revisions

Edit

New Page

## A Next-Generation Smart Contract and Decentralized Application Platform

Satoshi Nakamoto's development of Bitcoin in 2009 has often been hailed as a radical development in money and currency, being the first example of a digital asset which simultaneously has no backing or "intrinsic value" and no centralized issuer or controller. However, another - arguably more important - part of the Bitcoin experiment is the underlying blockchain technology as a tool of distributed consensus, and attention is rapidly starting to shift to this other aspect of Bitcoin. Commonly cited alternative applications of blockchain technology include using on-blockchain digital assets to represent custom currencies and financial instruments ("colored coins"), the ownership of an underlying physical device ("smart property"), non-fungible assets such as domain names ("Namecoin"), as well as more complex applications involving having digital assets being directly controlled by a piece of code implementing arbitrary rules ("smart contracts") or even blockchain-based "decentralized autonomous organizations" (DAOs). What Ethereum intends to provide is a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code.

► Pages 179

### Basics

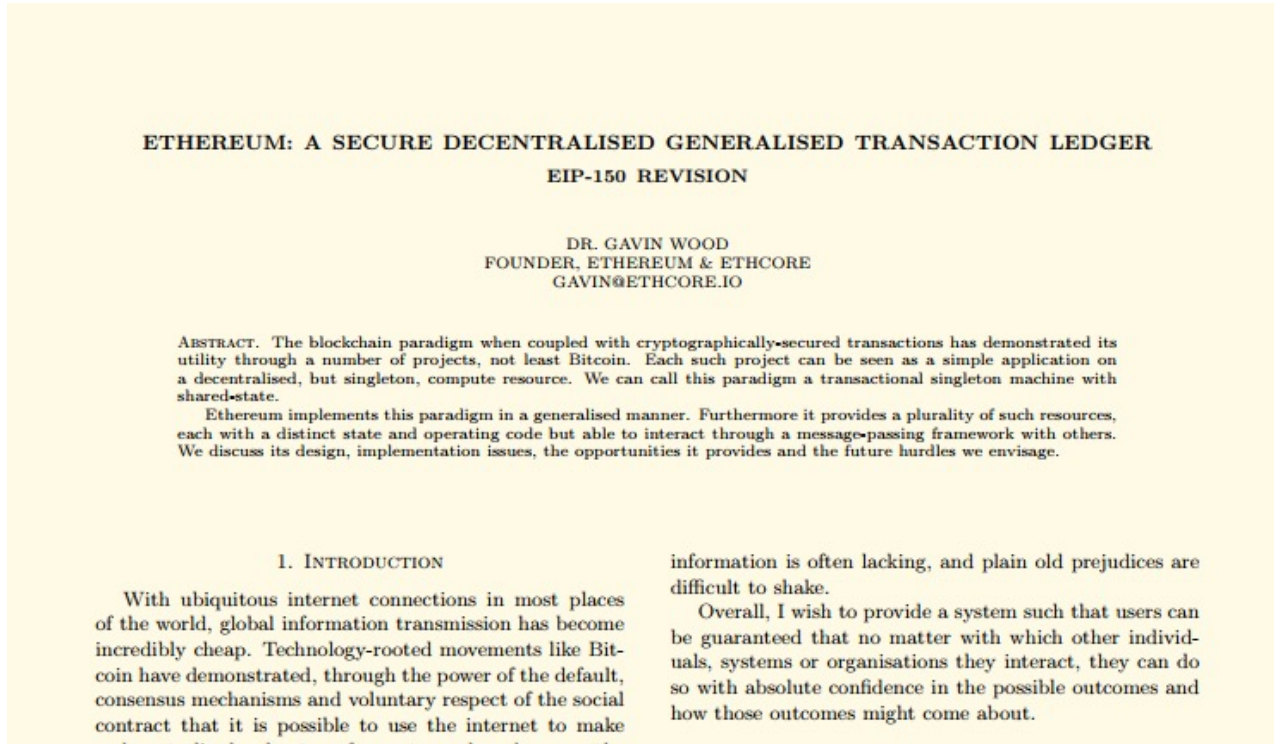
- [Home](#)
- [Wiki for \(old\) website](#) (still a good introduction)
- [Ethereum Introduction](#)
- [Ethereum Whitepaper](#)
- [Design Rationale](#)
- [EVM intro: Ethereum Yellow Paper, Beige Paper and Py-EVM.](#)
- [Getting Ether](#)
- [Uses: DAOs and dapps](#)
- [Releases](#)
- [FAQs](#)

[Ethereum Virtual Machine \(EVM\)](#)

[Ethereum Clients](#)

- **First draft** was written by Vitalik Buterin himself (**2013**)
- Contains **high level descriptions of Ethereum's core functionalities**
- **Living document** and **regularly updated** by Ethereum core developers (not only Buterin!)
- **Extensive summary** of the Ethereum platform and technology
- **Most current version** can be found the **public Git repository** of Ethereum:  
<https://github.com/ethereum/wiki/wiki/White-Paper>





Ethereum yellow paper: <https://ethereum.github.io/yellowpaper/paper.pdf>

- **Published in April 2014 by Dr. Gavin Wood**
- **Dr. Gavin Wood is still listed as the only author**
- **Defines the technical specification of Ethereum**
- **Very detailed**, contains mathematical function definitions and byte code mappings
- **Required to implement** a full node
- **Only updated** when **errors** are found or the **specification changes**

*“The Ethereum Foundation’s **mission is to promote and support Ethereum platform and base layer research, development and education** to bring decentralized protocols and tools to the world that empower developers **to produce next generation decentralized applications** (dapps), and together build a more globally accessible, more free and more trustworthy Internet.”*



- **Founded in June 2014 in Zug, Switzerland**
- **Non-profit** organization
- **Foundation council** consists of **Vitalik Buterin** and **Patrick Storchenegger** who is responsible for all legal affairs
- **Owns** (or had owned) at least **31.591 Bitcoins** funding capital due to the crowdsale



# Like Bitcoin, Ethereum is in Productive Use

## Transactions per Day



Ethereum Daily Transactions Chart

Source: Etherscan.io

Click and drag in the plot area to zoom in

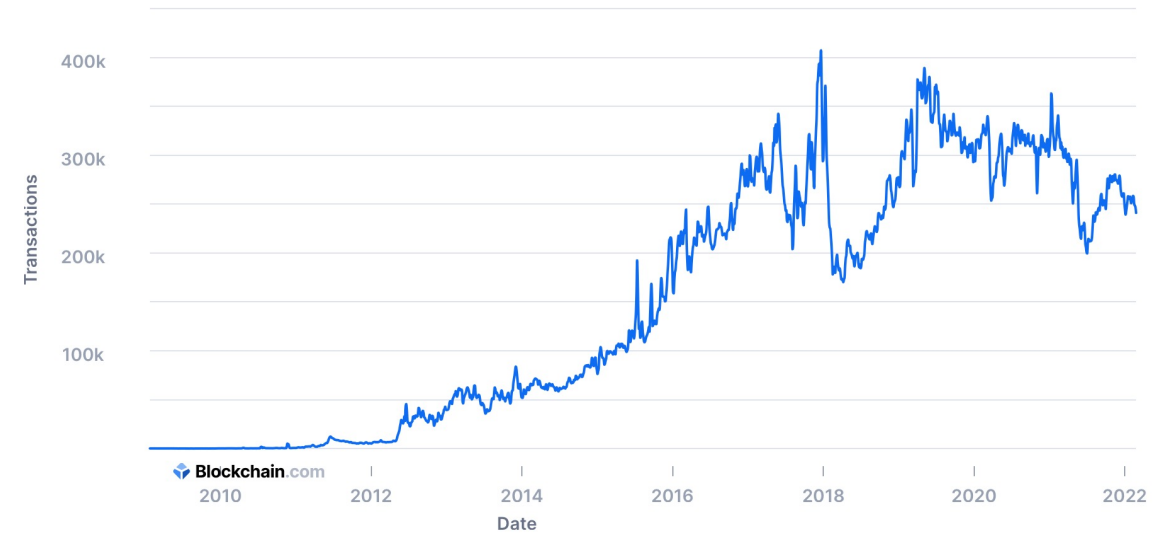


- **1.716.600** peak of transactions per day
- Currently almost **6x** the number of **BTC transactions** each day



Confirmed Transactions Per Day

The total number of confirmed transactions per day.



- **439.549** peak of transactions per day<sup>1</sup>

<sup>1</sup>The given graph shows 7-day average.

# Like Bitcoin, Ethereum is in Productive Use

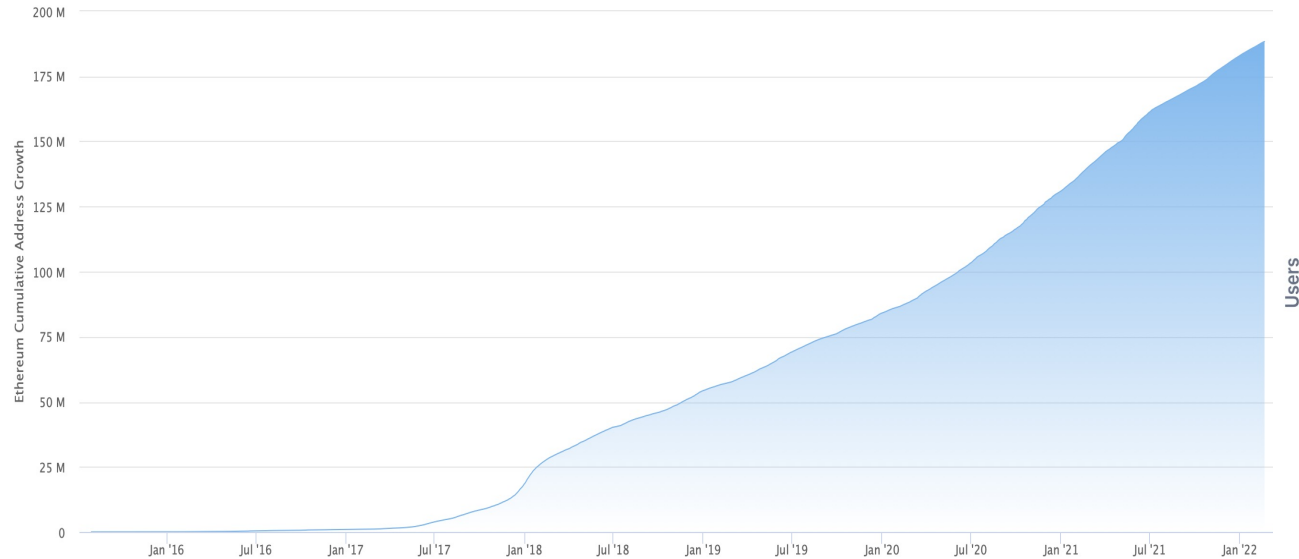
## Active Wallets



Ethereum Unique Addresses Chart

Source: Etherscan.io

Click and drag in the plot area to zoom in



- Currently around **188 million unique wallets** with at least one incoming or outgoing transaction



## Blockchain.com Wallets

The total number of unique Blockchain.com wallets created.



- Currently around **81 million unique wallets** with at least one incoming or outgoing transaction

# Like Bitcoin, Ethereum is in Productive Use

Also Uses Proof of Work



Ethereum Network Hash Rate Chart

Source: Etherscan.io

Click and drag in the plot area to zoom in



- As of **Jan 2022**, the network **hash rate** is at around **~1.010.019 GH/s**
- Mostly **GPUs** are used for hashing (**ASIC-resistant PoW**)
- Estimated **annual electricity consumption** at the current rate is **113.15 TW/h<sup>1</sup>**

<sup>1</sup>Remember: Bitcoins annual electricity consumption is 204.50 TW/h

## 1. Ecosystem

- Historical Overview
- Crowdsale Statistics
- Technical Papers
- Foundations
- Network Metrics

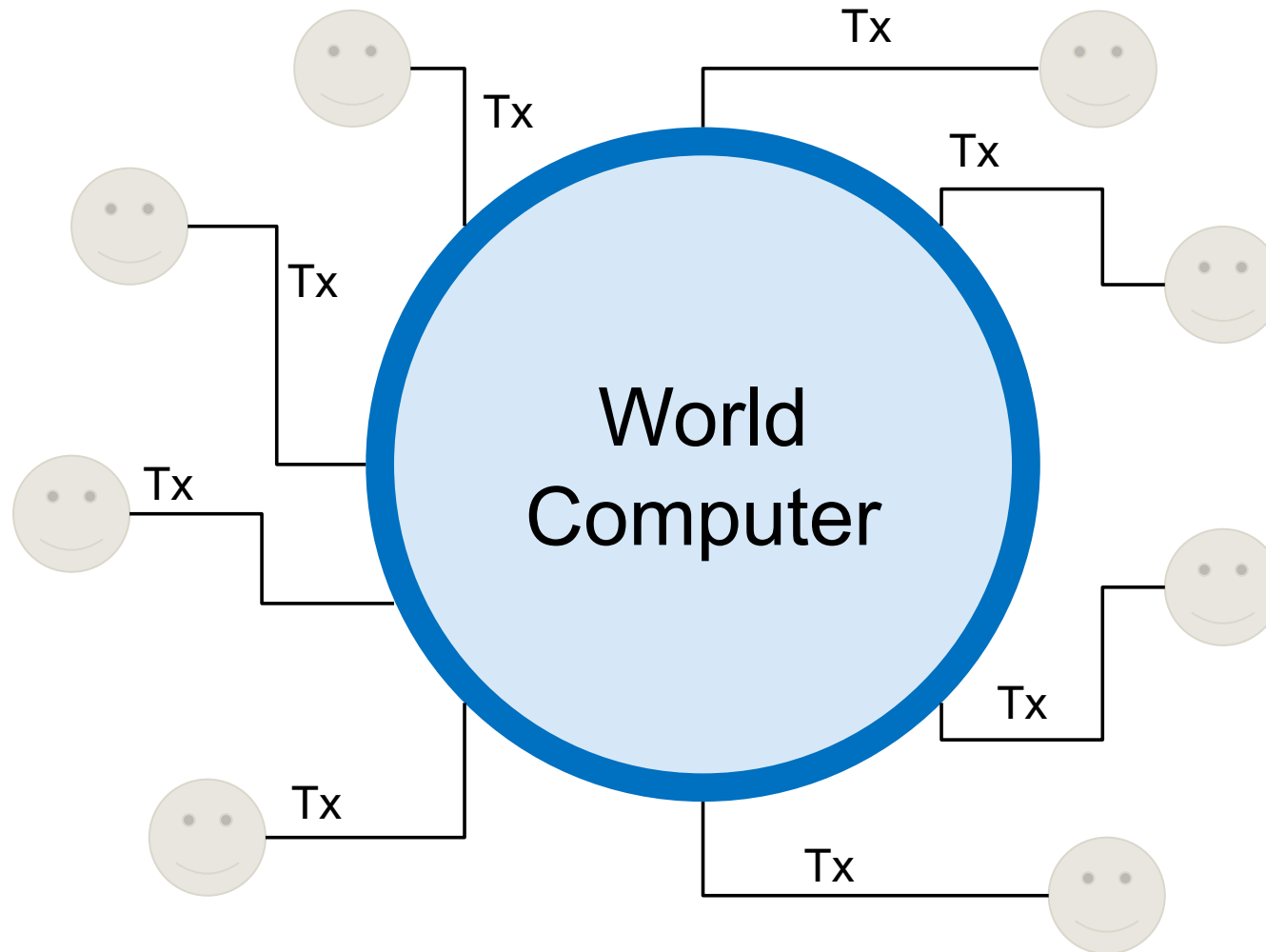
## 2. System Architecture

- Concept of a World Computer
- EVM
- Accounts
- Blockchain Properties
- Smart Contracts

## 3. Network Architecture

- Overview
- Node Types
- Clients
  - Geth
  - OpenEthereum



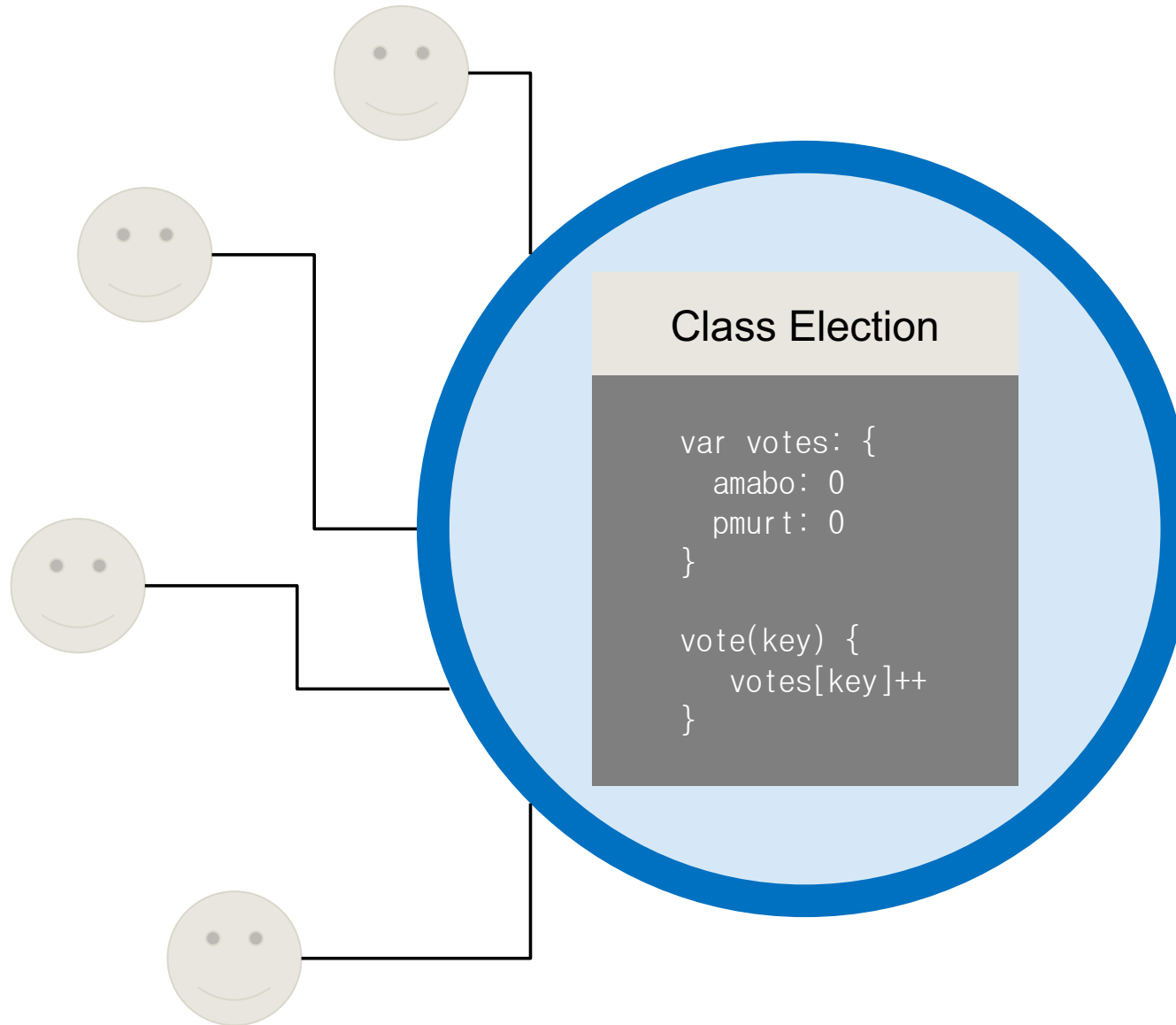


Tx = Transaction

### Properties

- **All participants are using the same computer**
- Users issue **transactions to call programs** on the computer
- **Everyone shares** the same **resources** and **storage**
- The **computer has no explicit, single owner**
- **Using** the computer's **resources costs money**

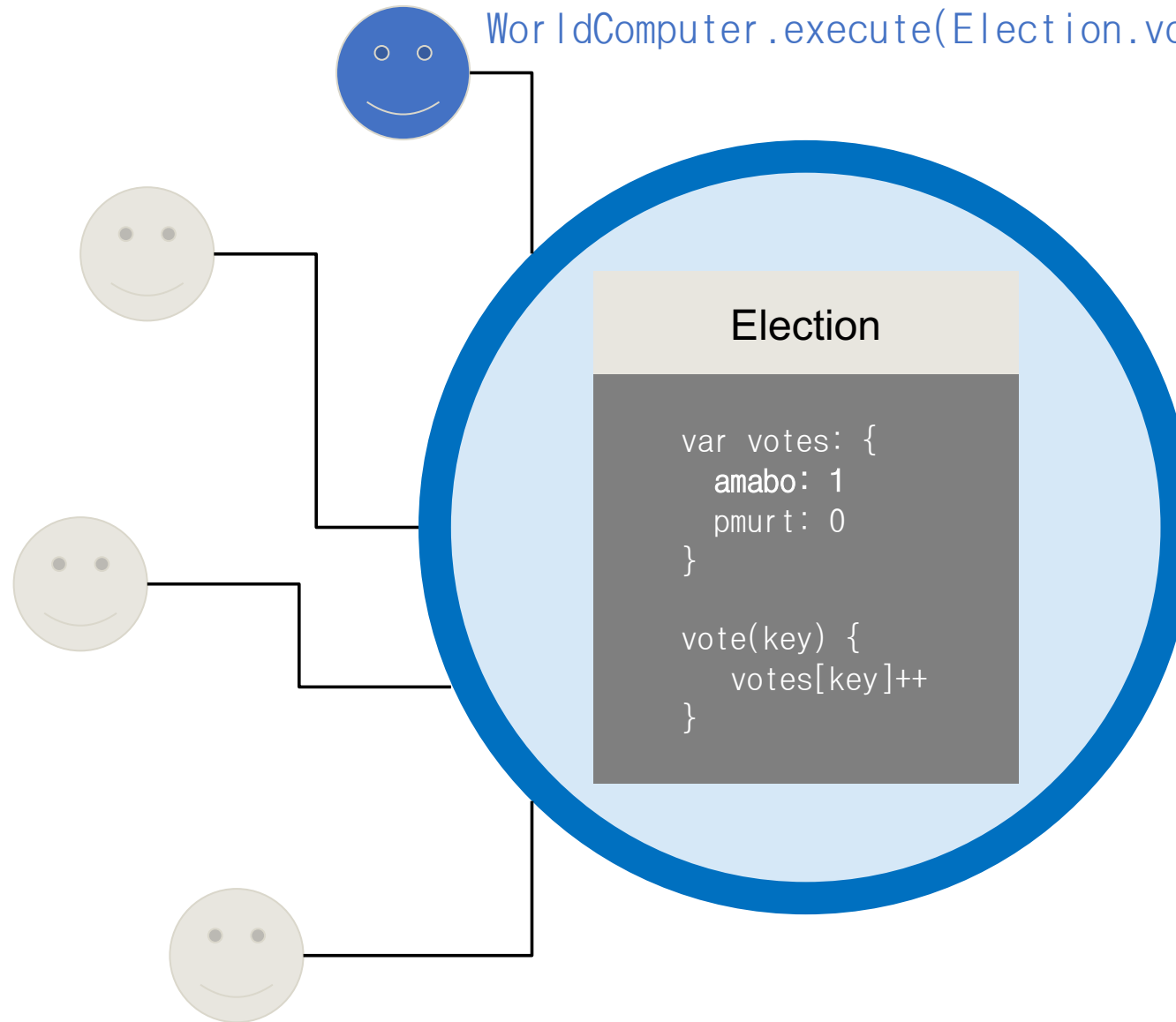
# Election Example Using a World Computer



## State of the world

**State 0 (initial)**  
(nothing happened yet)

# Election Example Using a World Computer (cont.)



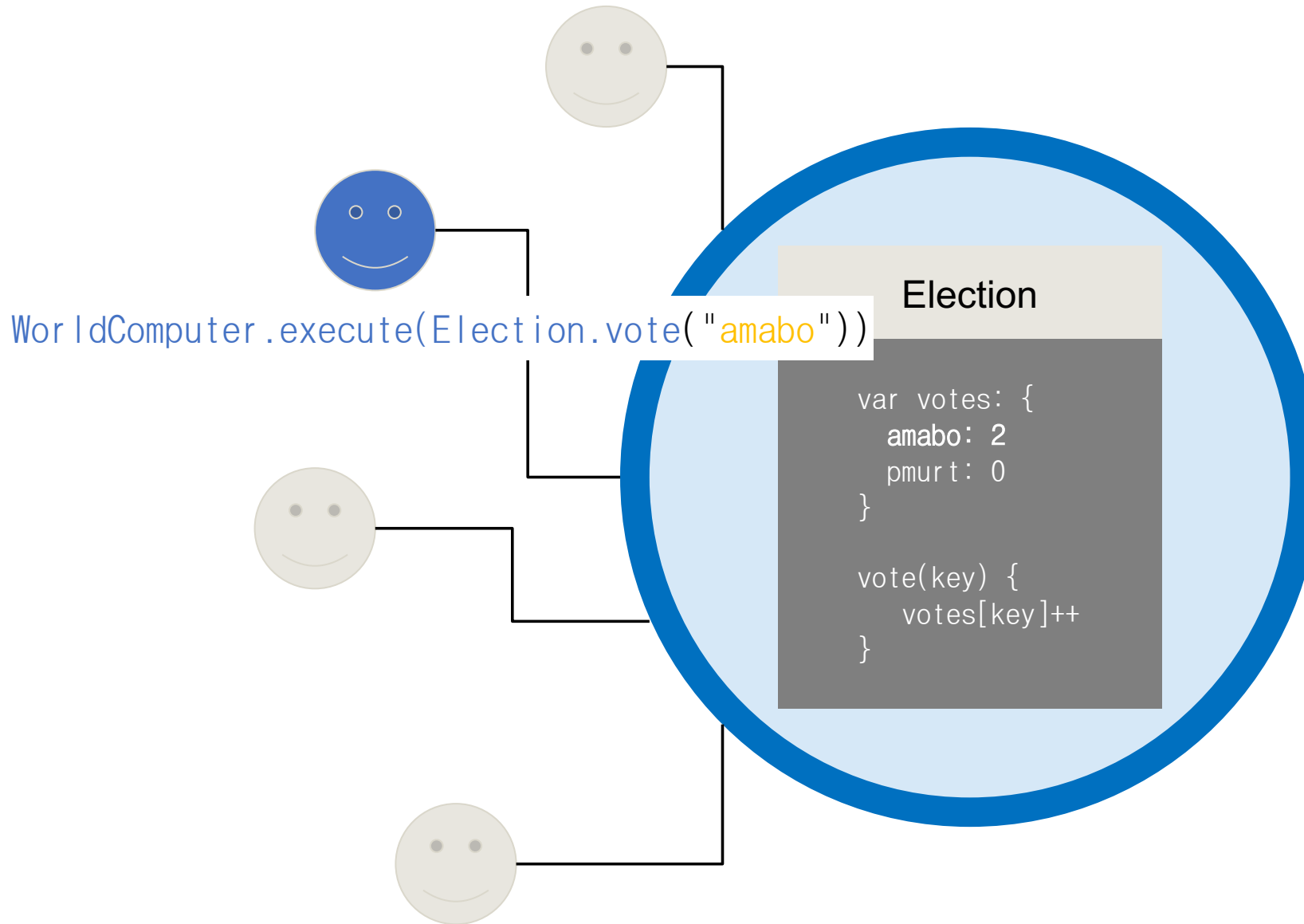
## State of the world

State 0 (initial)  
(nothing happened yet)

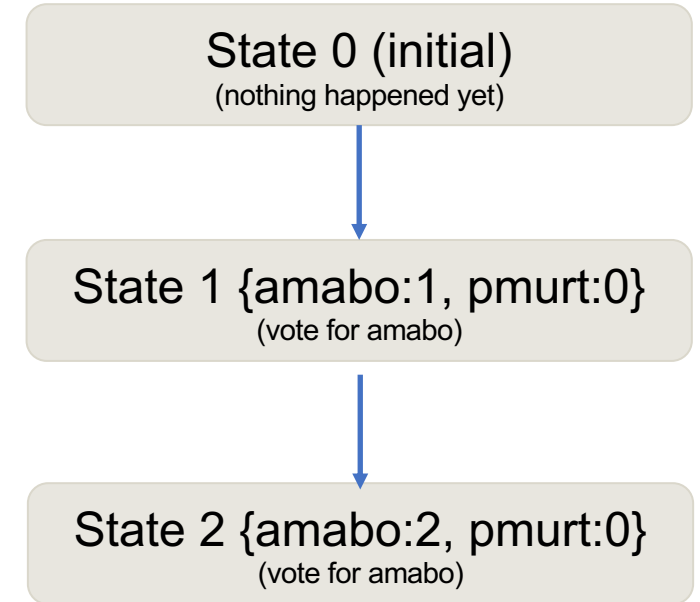


State 1 {amabo:1, pmurt:0}  
(vote for amabo)

# Election Example Using a World Computer (cont.)

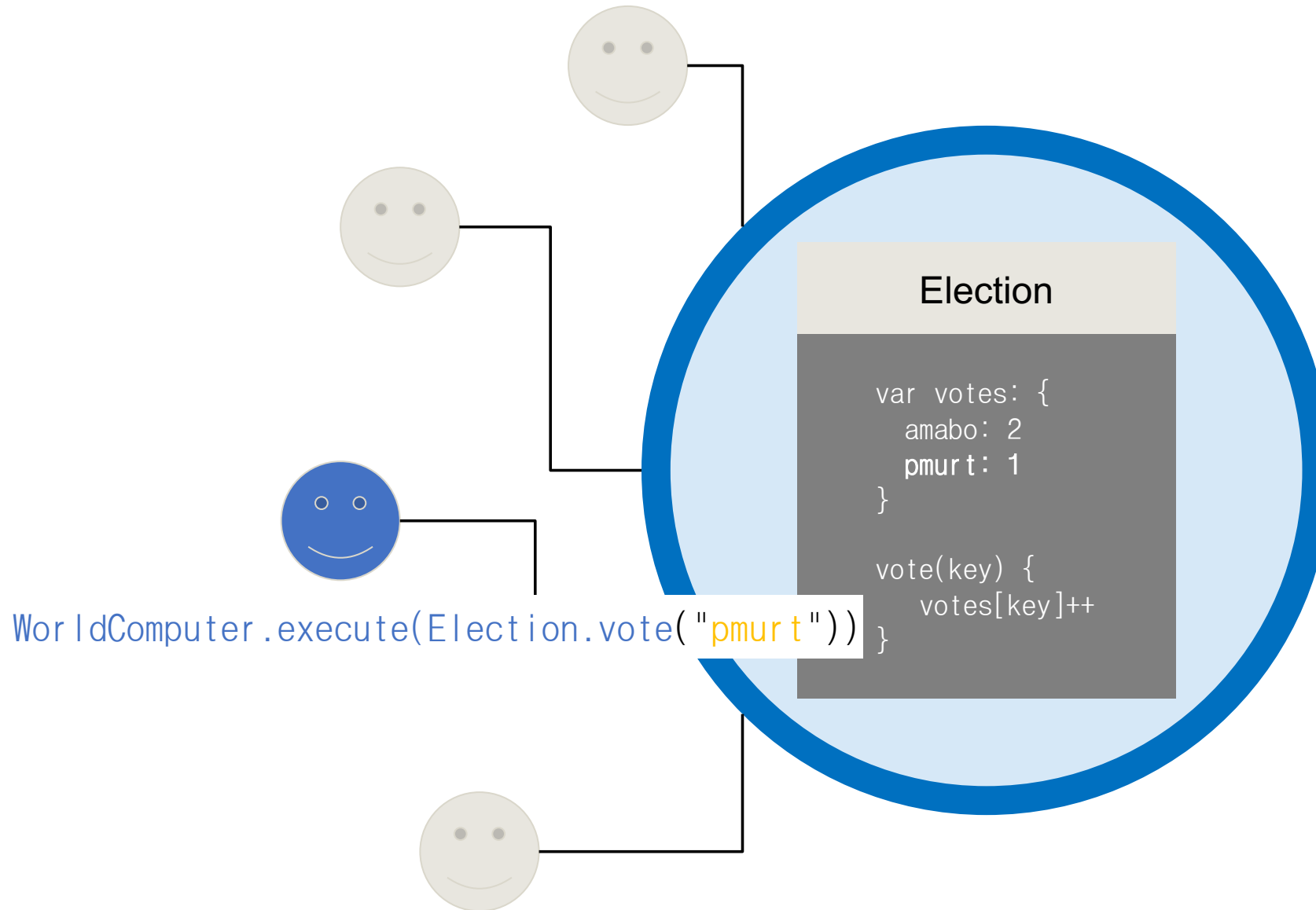


## State of the world

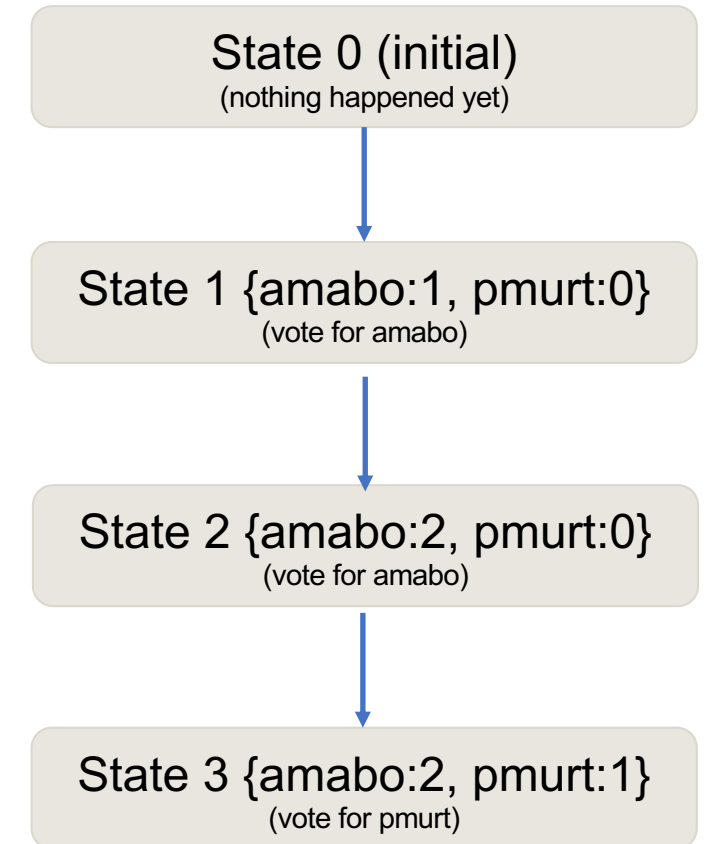




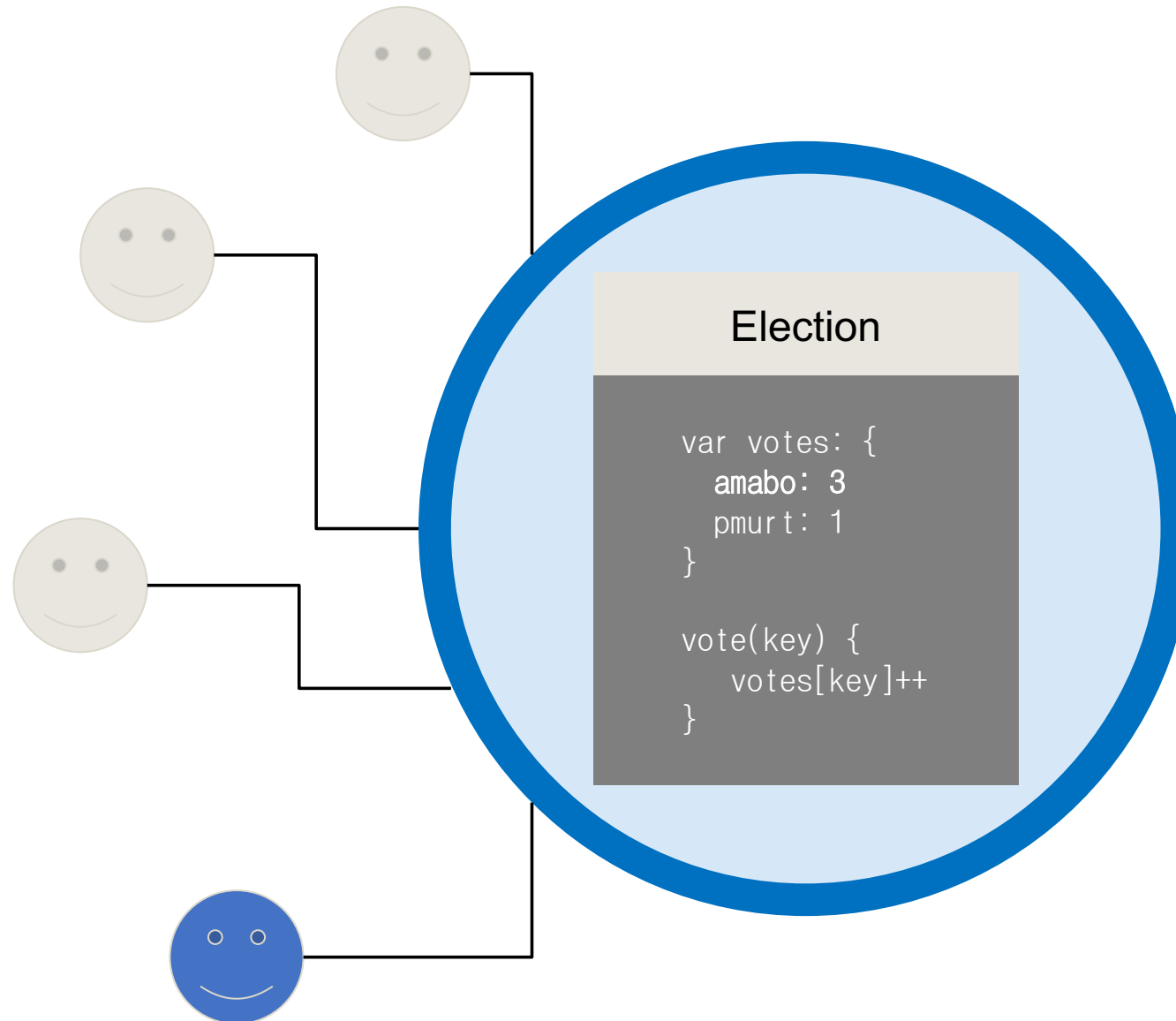
# Election Example Using a World Computer (cont.)



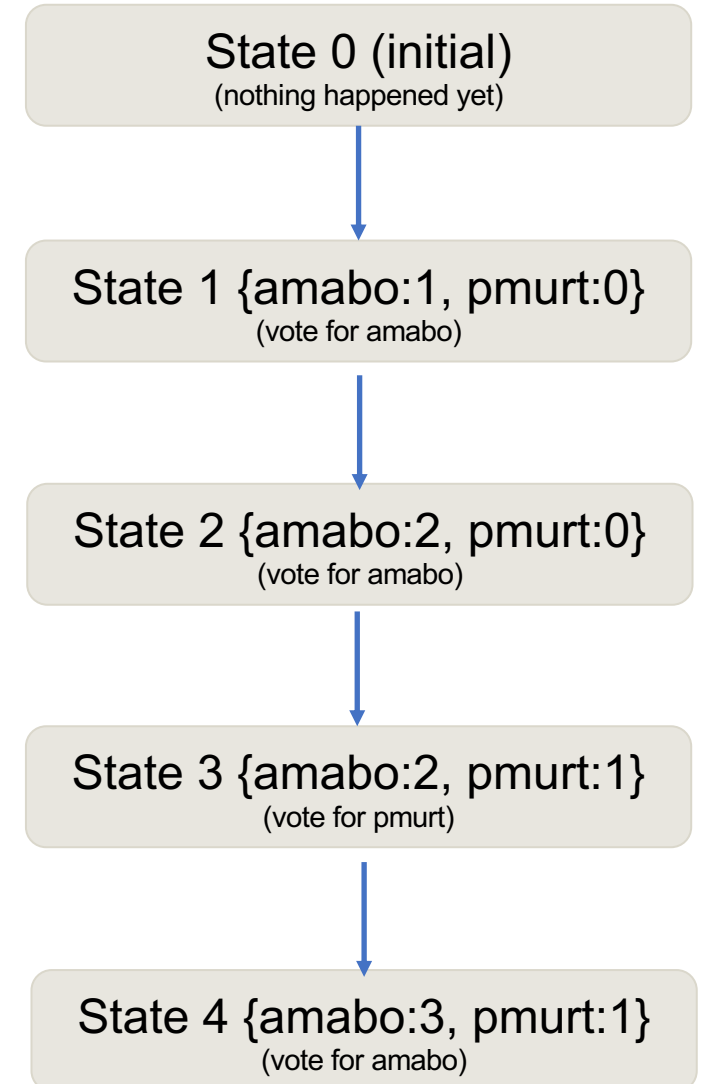
## State of the world



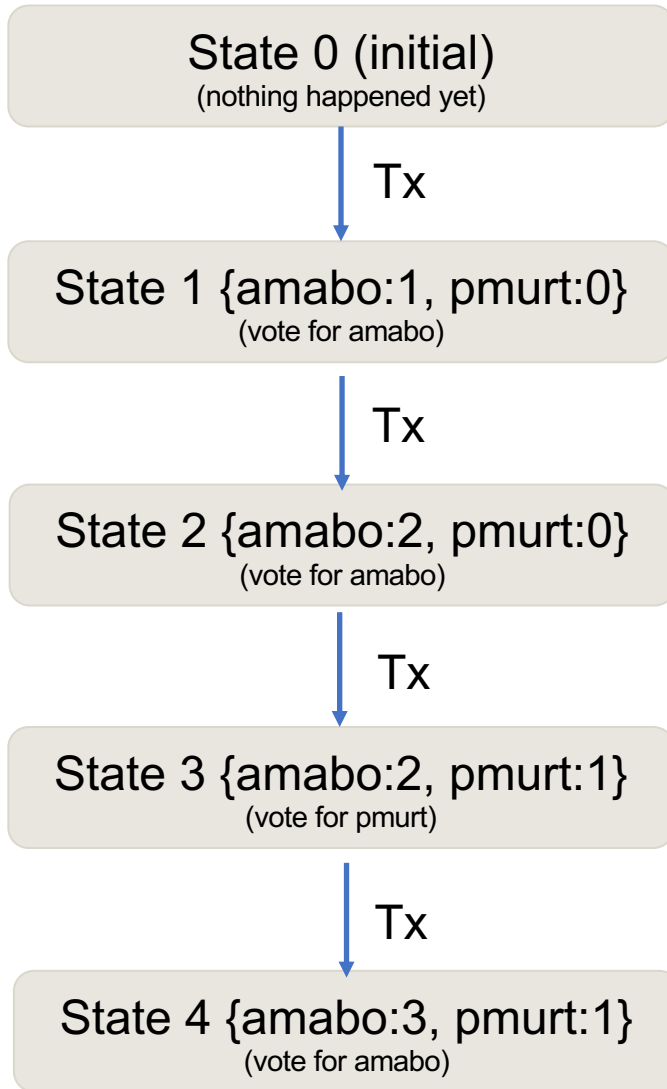
# Election Example Using a World Computer (cont.)



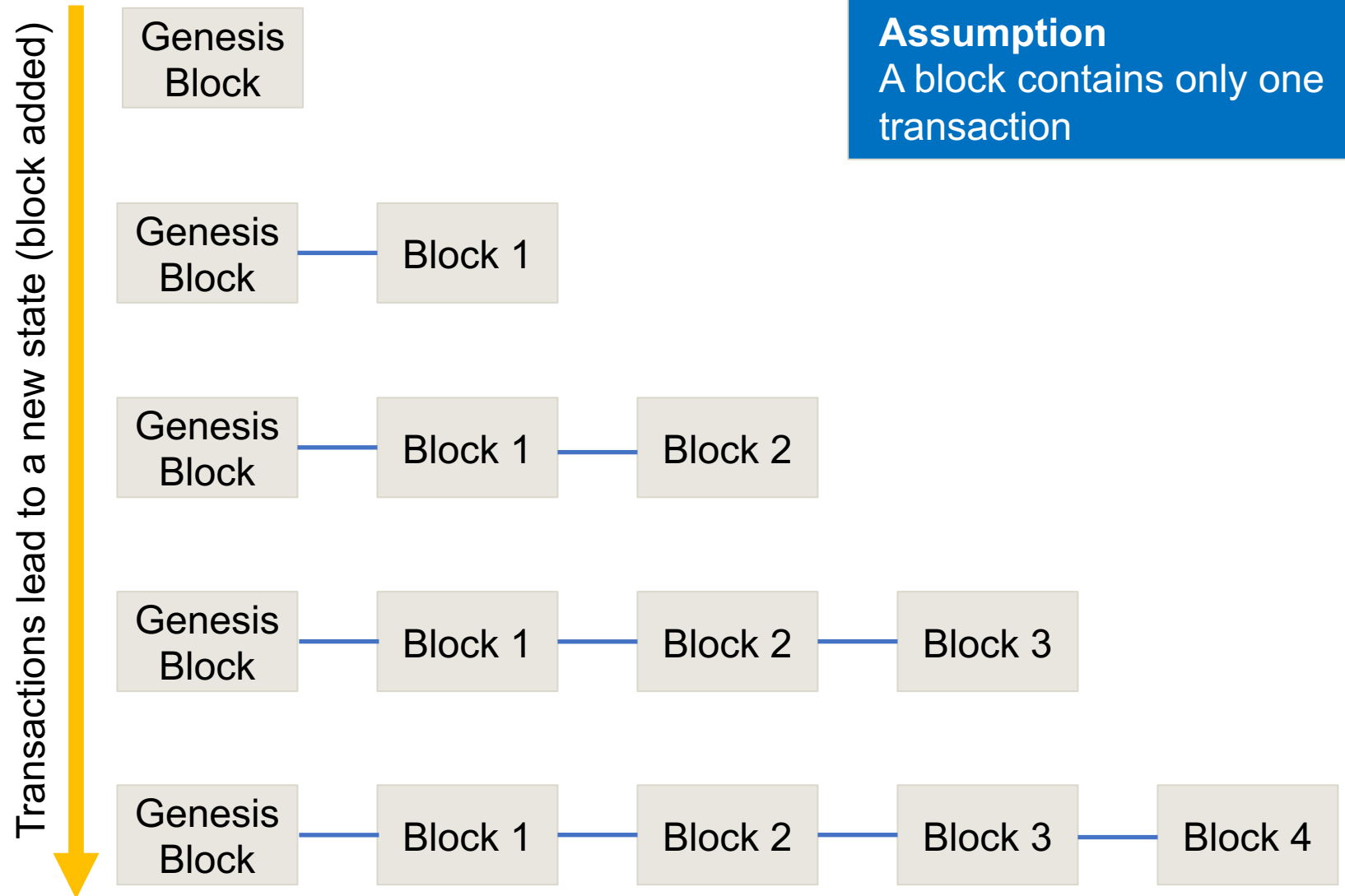
## State of the world



## State of the world



## Blockchain



**Assumption**  
A block contains only one transaction

# Brief Insight into the Ethereum Virtual Machine (EVM)

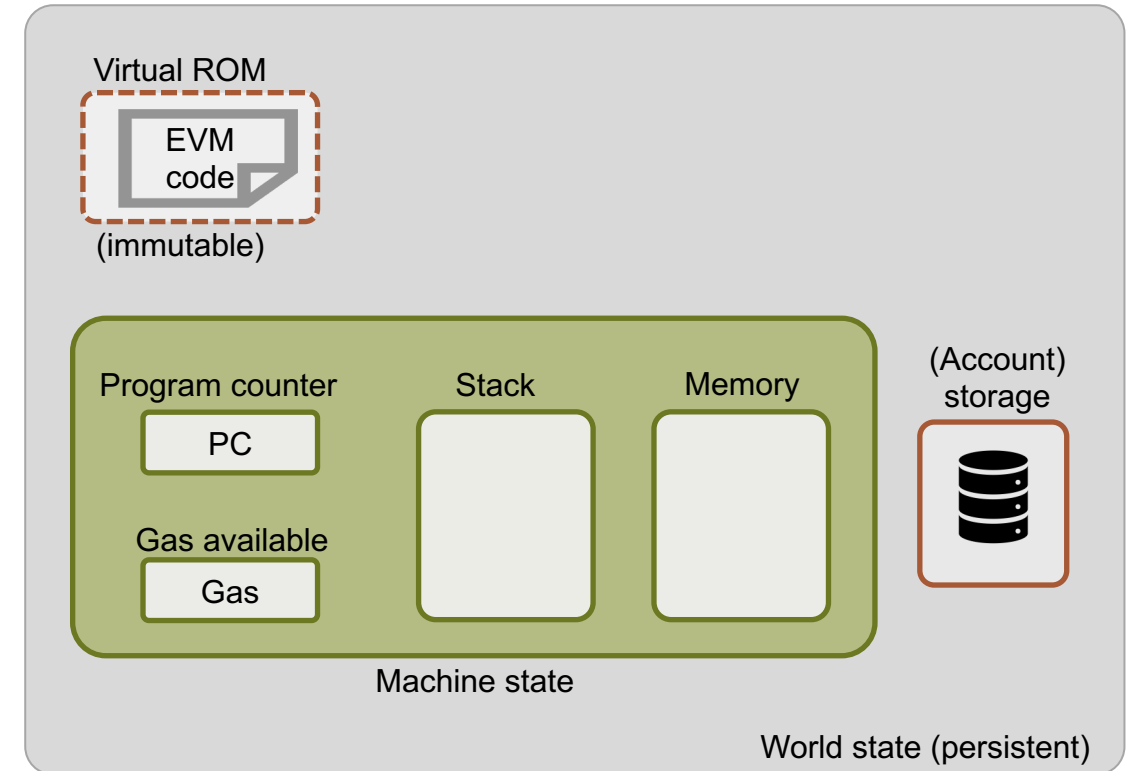
*What is a virtual machine in a blockchain?*

- Simply, virtual machines (VM) are mechanisms for creating instances of software that imitate real machines.

*What is the Ethereum Virtual Machine (EVM)?*

- EVM was the first virtual machine to be placed on a blockchain network, allowing programmers to conduct calculations on the blockchain for the first time. It's a stack-based architecture designed by Vitalik Buterin, which allows developers to create decentralized apps (dApps) on Ethereum. All Ethereum accounts and smart contracts are stored on this virtual machine.

## Ethereum Virtual Machine (EVM)





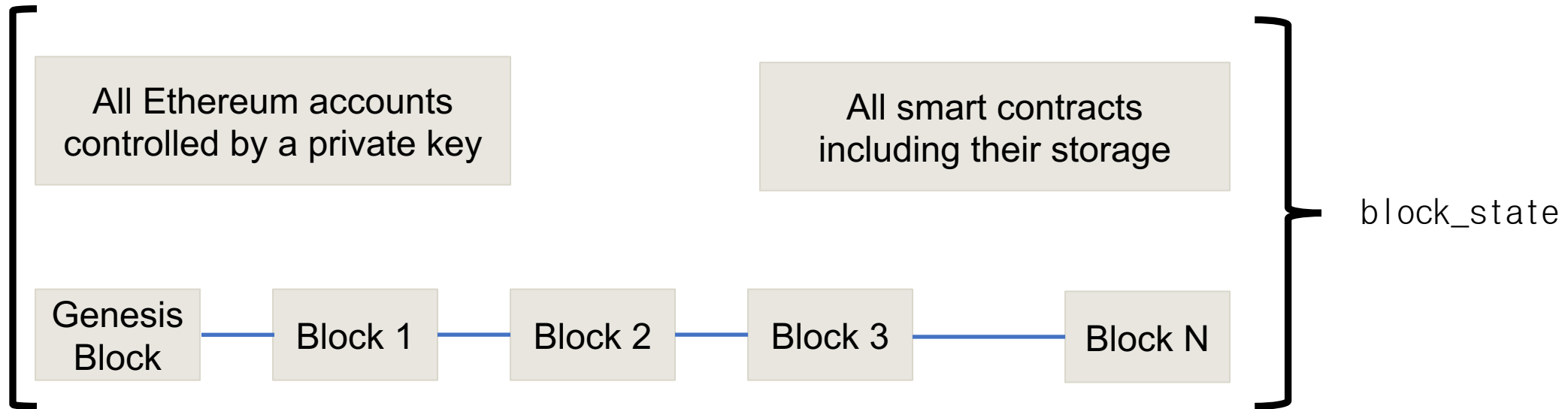
# Ethereum Virtual Machine (EVM)

## State Machine

The **EVM** specifies an **execution model for state changes** of the blockchain.

Formally, the **EVM** can be specified by the **following tuple**:  
(*block\_state*, *transaction*, *message*, *code*, *memory*, *stack*, *pc*, *gas*)

The *block\_state* represents the **global state** of the whole blockchain including **all accounts, contracts and storage**



Compared to Bitcoin, **Ethereum** uses an **account-based ledger**. Each **distinct address** represents a separate, **unique account**.

Ethereum supports two types of accounts:

## 1. Accounts that are controlled by private keys and owned externally

- This account type is called “Externally Owned Account” (EOA).
- Accounts that are controlled by a private key do not have any code stored on the blockchain. This type can be seen as the **default wallet of a user**<sup>1</sup>. 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**.

## 2. Smart contract<sup>2</sup> accounts which are controlled by their code

- Smart contracts are treated as **account** entities with their **own**, unique **address**.
- Contracts<sup>3</sup> **can send messages** to other accounts, both externally controlled and smart contracts.
- They **can't issue a transaction themselves**.
- They **have** a persistent **internal storage** for reading and writing.

<sup>1</sup>In this context, a user is an human being. Thus, EOAs are controlled by humans.

<sup>2</sup>Deeper overview on smart contracts can be found in the following slides.

<sup>3</sup>“Contract” is used as a short form of “smart contract”.

# Account Properties

On an abstract level, an Ethereum account is a 4-tuple containing the following data:  
(*nonce*, *balance*, *contract\_code*, *storage*)

## ***nonce***

An increasing number that is attached to any transaction to prevent replay attacks.

## ***balance***

The current account balance of the account in Ether.

## ***contract\_code***

The bytecode representation of the account. If no contract code is present, then the account is externally controlled.

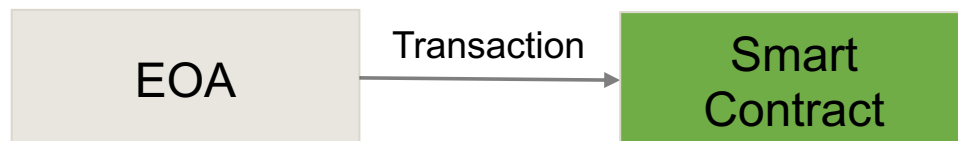
## ***storage***

The data storage used by the account and empty by default. Only contract accounts can have their own storage.

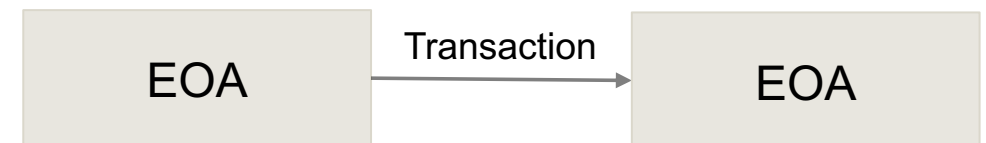
A **transaction** is a **signed data package** that is **always sent by a EOA** and contains the following data:

- The recipient of the transaction
- A signature identifying the sender
- The amount of ether to transfer from the sender to the recipient
- An optional data field – data field is used for function call arguments (e.g. function inputs)
- A *GASLIMIT*<sup>1</sup> value, representing the maximum amount of gas you are willing to consume on a transaction
- A *GASPRICE* value, representing the fee the sender pays per computational step
- There are two types of transactions: From EOA to EOA and from EOA to smart contract

### Type 1: EOA to Smart Contract



### Type 2: EOA to EOA



<sup>1</sup>Same terminology is also used for defining the maximum amount of gas a block uses.



## Message

A message is very similar to a transaction. Messages are only sent by contracts and exist only virtually, i.e. they are not mined into a block like transactions.

### A message contains:

- The sender of the message (implicit)
- The recipient of the message
- The amount of ether to transfer alongside the message
- An optional data field
- A *GASLIMIT* value

Whenever a **contract calls** a method on **another contract**, a virtual **message** is sent.  
Whenever an **EOA calls** a method on a contract, a **transaction** is sent.



### **code**

The code basically represents a smart contract as bytecode. For the EVM, a smart contract is a sequence of opcodes similar to assembly code.

### **Example:**

```
PUSH1 0x60  
PUSH1 0x40  
MSTORE  
PUSH1 0x04  
CALLDATASIZE  
LT  
PUSH2 0x00b6  
JUMPI  
PUSH4 0xffffffff
```

### **memory**

An infinitely expandable byte array that is non-persistent and used as temporal storage during execution.

### ***stack***

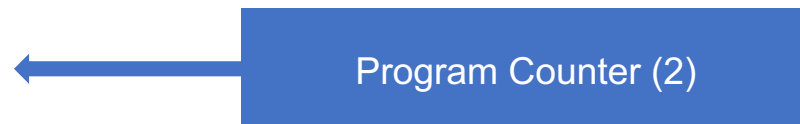
The stack is also used as a fast, non-persistent buffer to which 32 byte values can be pushed and popped during execution.

### ***pc***

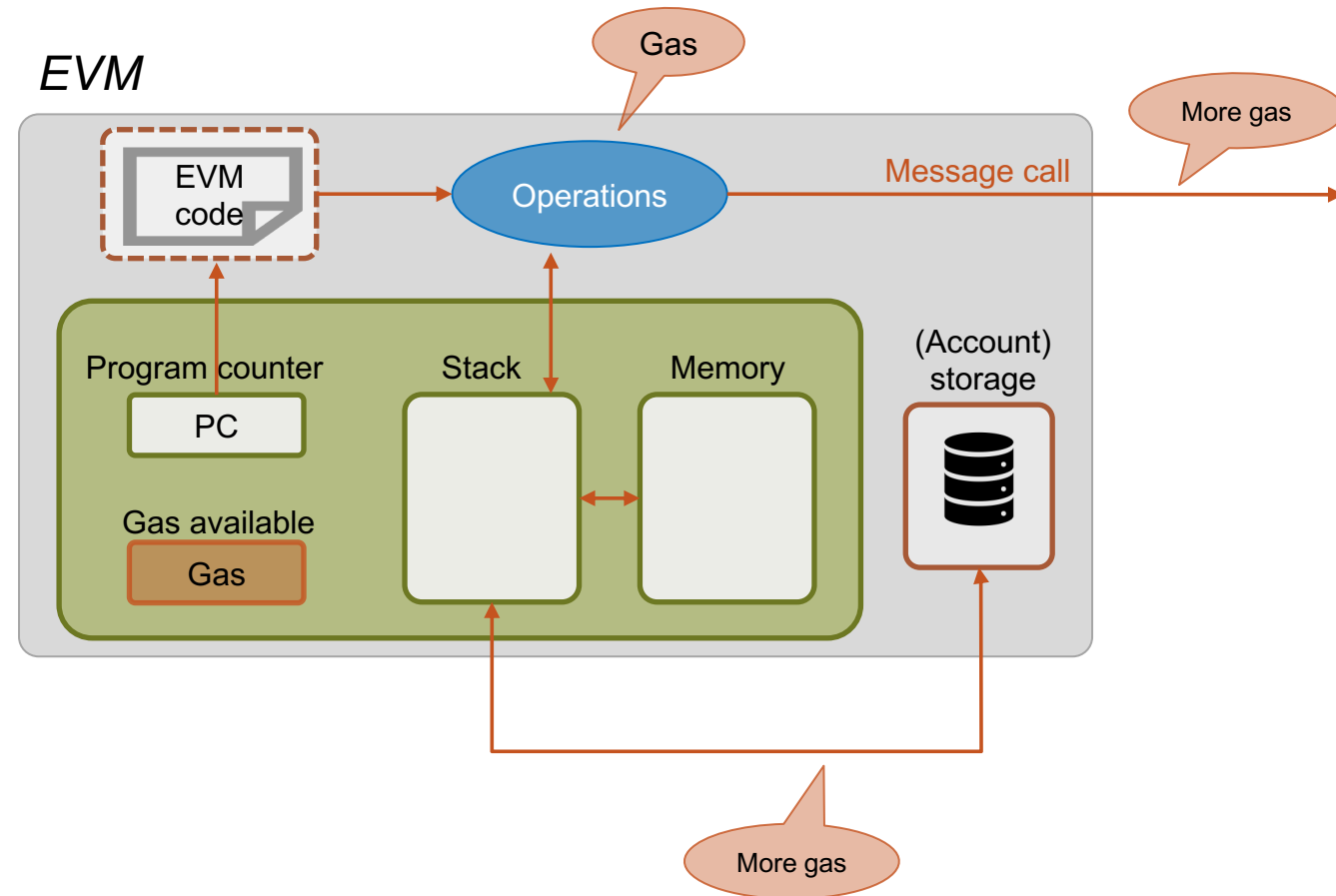
PC stands for “program counter”. The program counter is always initialized with 0 and points to the position of the current opcode instruction.

### **Simple Opcode Execution Example:**

```
0    PUSH1 0x60
1    PUSH1 0x40
2    MSTORE
3    PUSH1 0x04
4    CALLDATASIZE
5    LT
6    PUSH2 0x00b6
7    JUMPI
8    PUSH4 0xffffffff
```



- **Every executed opcode instruction** uses a miner's computational resources and therefore **costs a certain fee (called gas)**.
- Each opcode uses a certain amount of gas which may depend on the arguments of the operation, e.g., number of bytes to be allocated.
- The opcode for **selfdestruct (address)** uses **negative gas** because it frees up space from the blockchain.



## Legacy Transactions

- At the beginning of a transaction, the sender must specify a maximum amount of *gas* that he/she is willing to pay for the transaction to be executed.
- The sender can set an arbitrary amount of Ether he/she is willing to pay for each gas unit called *gasprice*.
- The final cost for each transaction is  $gas * gasprice$  and this fee is **received entirely by the miner**.
- If a transaction requires more gas than the maximum specified gas, the transaction will fail. On the other hand, if it takes less, the sender only pays for the gas that was used.
- The actual **size of a block** is **not limited** like in Bitcoin. Instead, the miner sets a *gaslimit* that defines how much computational resources he is willing to provide for each block.

## EIP-1559 Transactions (*since August 5<sup>th</sup>, 2021*)

- The Ethereum Improvement Proposal (EIP)-1559, enabled the deflationary functionality of the Ethereum blockchain and sought to alter how gas prices are calculated.
- The main motivation for EIP-1559 was the large volatility in transaction fees.
- With EIP-1559, gas price is now divided into two parts:
  - *base fee*: calculated by the network based on the demand level (**burnt**) – predictable
  - *priority fee (tip)*: determined by the transaction sender (**received by the miner**)
- Transaction fee =  $gas * (basefee + tip)$
- As of January 5<sup>th</sup>, 2022, the network burns an average of 6.18 ETH every minute. So far, the Ethereum blockchain has burned a total of 1,887,620 ETH.

Deflation information by Ultrasound: <https://ultrasound.money>

For further info about EIP-1559, see <https://eips.ethereum.org/EIPS/eip-1559>

# An EIP-1559 Transaction

② Value: 0.11670416 Ether (\$311.47)



② Transaction Fee: 0.001158913272762 Ether (\$3.09)

② Gas Price: 0.000000055186346322 Ether (55.186346322 Gwei)

② Ether Price: \$2,597.94 / ETH

② Gas Limit & Usage by Txn: 21,000 | 21,000 (100%)

② Gas Fees: Base: 55.086346322 Gwei | Max: 57.07 Gwei | Max Priority: 0.1 Gwei

② Burnt & Txn Savings Fees:  Burnt: 0.001156813272762 Ether (\$3.09)  Txn Savings: 0.000039556727238 Ether (\$0.11)

② Others: Txn Type: 2 (EIP-1559) Nonce: 0 Position: 104

Transaction: <https://etherscan.io/tx/0xf053bfba668022c43f837939373b2fc35672d04d67bbc48d3d7c42e9d90ecc7e>



# Properties of the Ethereum Blockchain

Ethereum (currently) is a Proof-of-Work (PoW) blockchain like Bitcoin.

- The **PoW algorithm** used by Ethereum is called **Ethash** and designed to be ASIC resistant.
- In Ethereum blockchain, when a miner creates a block successfully, they are rewarded with a digital token called "ether". Miners spend energy on mining and solving a puzzle. In return they get a block reward which is *transaction fee + mining reward*. Mining reward is 2 ETH (\$5,435 as of 25<sup>th</sup> of February 2022).
- The **mining difficulty** is **adjusted** after **each block**.
- Correctly mined blocks that are **outpaced** by a **block** of another miner are **not orphaned** like in Bitcoin but **added as uncle blocks**.
  - The idea behind this is to counter mining centralization because miners who mine a correct block are still rewarded.
  - As of January 2022, average block time of Ethereum blockchain is 12 to 14 seconds.<sup>1</sup>
- The **transactions** in the **uncle block** are considered **invalid**.

<sup>1</sup>Bitcoin's block time is 10 minutes in average.

- A **smart contract** is a **set of functions** that can be called by other users or contracts.
- They can be used to **execute functions, send ether or store data**.
- Each smart contract is an account holding object, i.e. **has its own address**.
- Smart contracts have some peculiarities compared to traditional software.

## Security

The **development process** of smart contracts **requires special attention on security**.

Once **deployed**, a **contract** is **publicly accessible** by anyone on the network with the following information:

- Address of the smart contract
  - OPCODE
  - Number of public functions and their hash signature
- 
- Furthermore, the whole transaction history is accessible (function calls + actual arguments).
  - **Smart contracts** – once **deployed** – **cannot be changed or patched** anymore.

➔ **All contracts deployed on the Ethereum blockchain are publicly accessible and can't be patched.**

On October 13th, 2018, Ethereum's creator, Vitalik Buterin, has expressed his regret about using the term "smart contracts". He states that he should have called them "persistent scripts", however, he thinks this wouldn't have been as catchy as calling them smart contracts.

Meanwhile, Ethereum associate, Vlad Zamfir replied to Vitalik's tweet by preferring the term "stored procedures" instead.



**vitalik.eth**  @VitalikButerin · 13 Oct 2018

To be clear, at this point I quite regret adopting the term "smart contracts". I should have called them something more boring and technical, perhaps something like "persistent scripts".



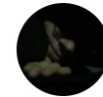
85



301



766



**Vlad Zamfir**  
@VladZamfir

Replying to @VitalikButerin @CleanApp and 4 others

I like "stored procedures"

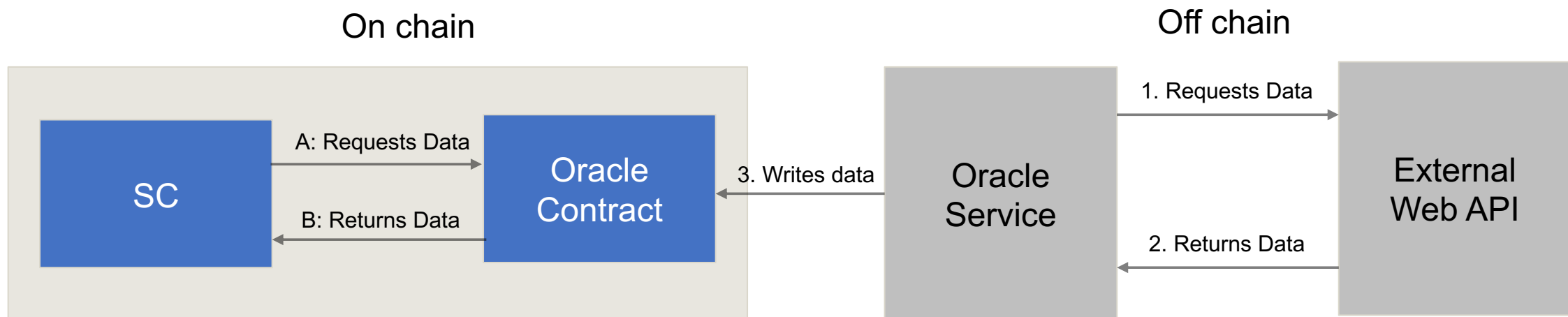
But I for one don't regret the terminology, it has been a great learning opportunity for everyone 😊

7:26 pm · 13 Oct 2018 · Twitter for Android

# Brief Insight into Smart Contracts as a Closed System

**Smart contracts can't access** any **data** from **outside** the **blockchain** on their own. There are no HTTP or similar network methods implemented to call external services. This is on purpose to **prevent non-deterministic behavior** once a function is called (there are also no functions to generate random values).

Currently, the **only way** to write smart contracts **using external data** (e.g. weather data, traffic data etc.) is to **use oracles**. Oracles are basically third-party services that verify data from web services and write the data via a special smart contract to the blockchain. Other smart contracts can now call the oracle contract to get the data.



Usually, smart contracts are not written as a sequence of opcodes instructions directly. **Solidity** is a high-level language with a JavaScript-like syntax and the de facto **standard** for writing **Ethereum smart contracts**. However, unlike JavaScript, Solidity is **statically typed**.

## Language properties



- Statically typed
- Object-oriented
- Supports inheritance
- Complex, user-defined types
- Public & private methods
- Dynamic binding
- Compiled to EVM opcode instructions

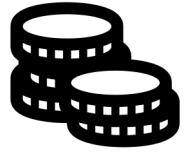
```
pragma solidity ^0.8.10;

contract HelloWorld {
    string public hello = "Hello World";
}
```

# Use Cases for Smart Contracts

## Fungible Token Systems

Fungible token systems are currently the largest use case for smart contracts, mostly used to collect money via initial coin offerings (ICOs). Usually, tokens work as a sub-currency of Ethereum and represent a certain asset such as a stock. *Popular examples:* Tether (USDT), Shiba Inu (SHIB), Polygon (MATIC)



## Non-Fungible Tokens (NFTs)

At a very high level, NFTs are unique cryptographic tokens that are stored on a blockchain. There are many types of NFTs. They can be anything digital such as drawings, music, media, virtual fashion items, etc. *Popular examples:* Bored Ape Yacht Club, CryptoPunks



## Play-to-Earn Games

Lately, smart contracts are also being used in *play-to-earn* games where users earn game-native tokens which they can use to buy or trade in-game assets. *Popular examples:* Axie Infinity, Decentraland



## Decentralized Autonomous Organization (DAO)

DAOs are non-hierarchical business models that are collectively owned by a group of members. Think of it as a generalization of multi-signature wallets where all decisions are taken based on members' votes. Thus, no single member can unilaterally take a decision (e.g., send a transaction). DAOs are useful for starting an organization with people you have limited trust (e.g., only communicating over the internet) since you only need to trust the system design and no other single member.

*Popular examples:* BitDAO, Moloch, Maker

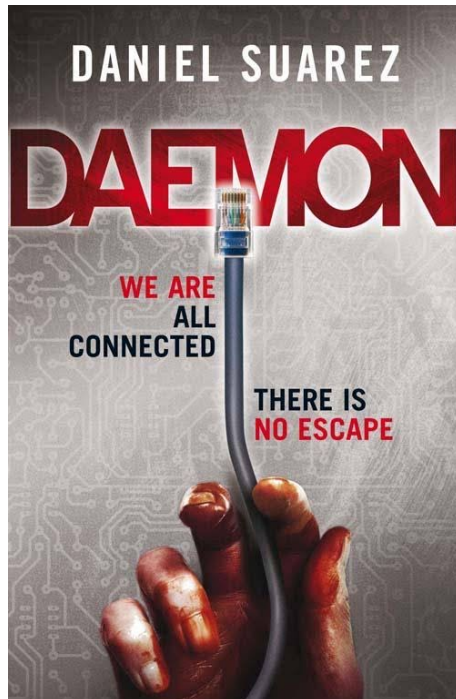
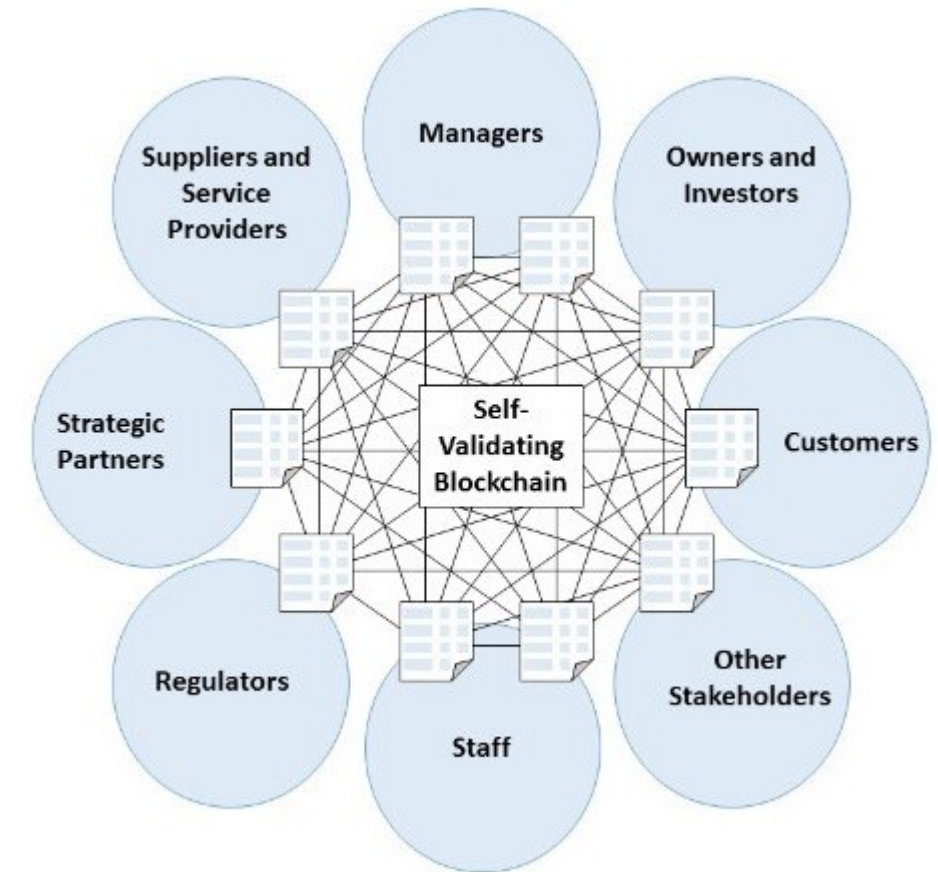




# Brief Insight into Decentralized Autonomous Organizations (DAOs)

## Vision

- Create a fully digital (virtual) organisation.
- The organisation exclusively uses smart contracts to interact with its shareholders, employees, customers, suppliers, partners and public authorities.
- These stakeholders can be humans or organizations in the “real world” or other DAOs.





## 1. Ecosystem

- Historical Overview
- Crowdsale Statistics
- Technical Papers
- Foundations
- Network Metrics

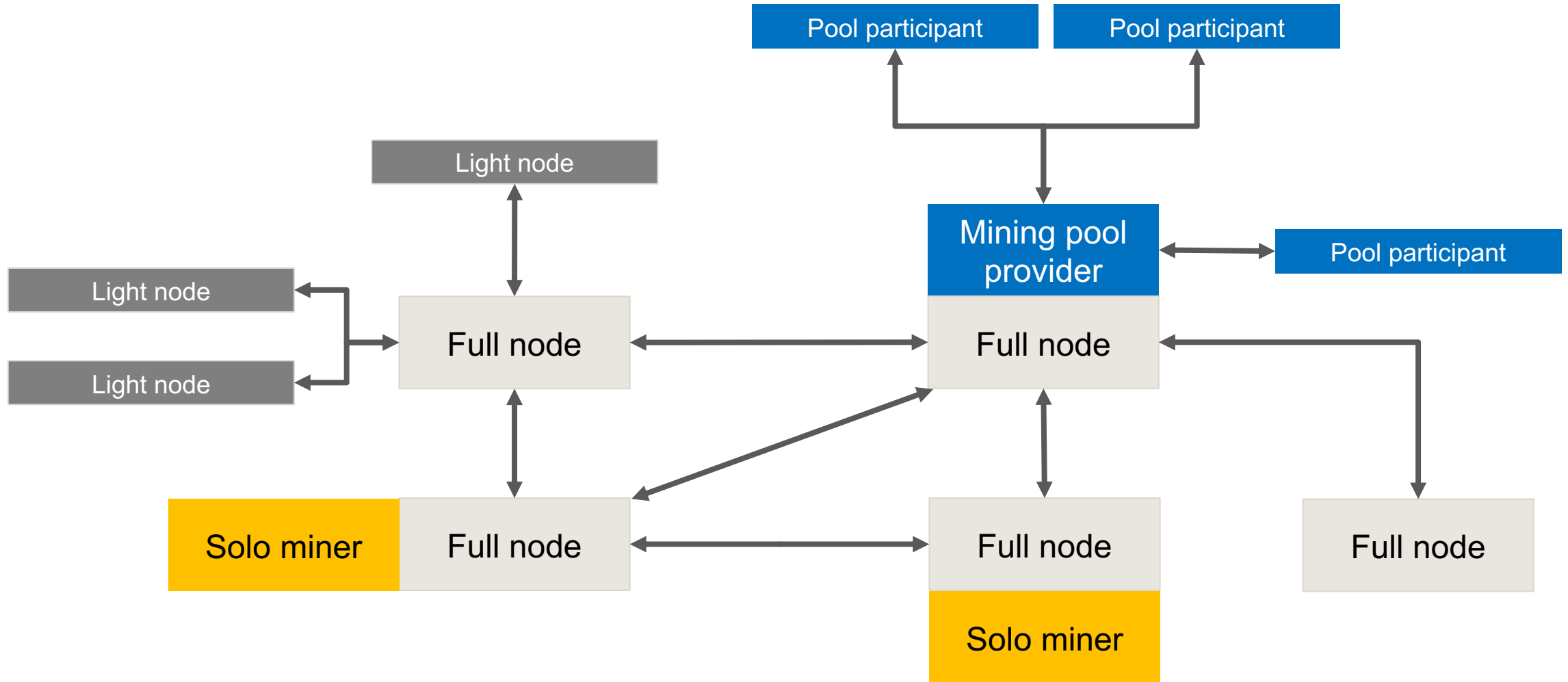
## 2. System Architecture

- Concept of a World Computer
- EVM
- Accounts
- Blockchain Properties
- Smart Contracts

## 3. Network Architecture

- Overview
- Node Types
- Clients
  - Geth
  - OpenEthereum

# Network Architecture Overview



## Full nodes

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.

## Light nodes

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).

## Solo miner

A solo miner is an entity that tries to mine a block on its own. At the current network hashing rate this is practically impossible. However, in order to mine a block it is required to have a synced copy of the full blockchain.

## Mining pools

Mining pools are a coalition of entities combining their hash power to solve a mining problem. A pool consists of a **controller** that **splits** and **distributes** the **mining puzzle** among the participants.

# Popular Ethereum Implementations

## Nodes with Clients

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

The most commonly used and **official Ethereum implementation**. Geth is implemented in Go and provides a command-line interface for running a full node. Geth comes with a JavaScript console and a JSON RPC server. Through which – if publicly exposed – other (light) nodes could connect to the network.

```
C:\Users\steve\AppData\Roaming\Ethereum\Wallet\binaries\Geth\unpacked>geth.exe attach
Welcome to the Geth JavaScript console!

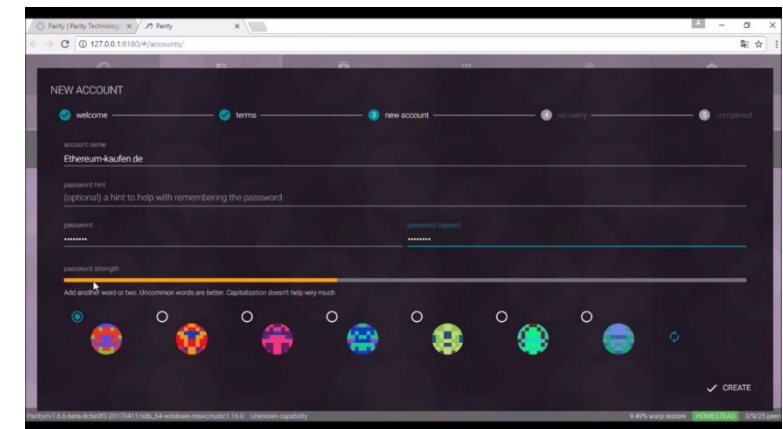
instance: Geth/v1.6.6-stable-10a45cb5/windows-386/go1.8.3
coinbase: 0xb134e0fd9df89f416084ff5d5f1addf65f866128
at block: 0 (Thu, 01 Jan 1970 07:00:00 +00)
datadir: C:\Users\steve\AppData\Roaming\Ethereum\testnet
modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> miner.start();
null
> miner.start(1);
null
>
```

### OpenEthereum (former: Parity)

OpenEthereum is another popular Ethereum implementation in Rust with the goal to be “*the fastest, lightest, and most secure Ethereum client*”.

OpenEthereum ships with a browser-based UI which is considered as a very user-friendly way to interact with Ethereum. However, the multi-signature wallet used by OpenEthereum was responsible for the *biggest (based on USD) hack in Ethereum’s history*. The company Parity transitioned the Parity codebase and maintenance to a DAO to allow for a continued development.



# Peer Discovery

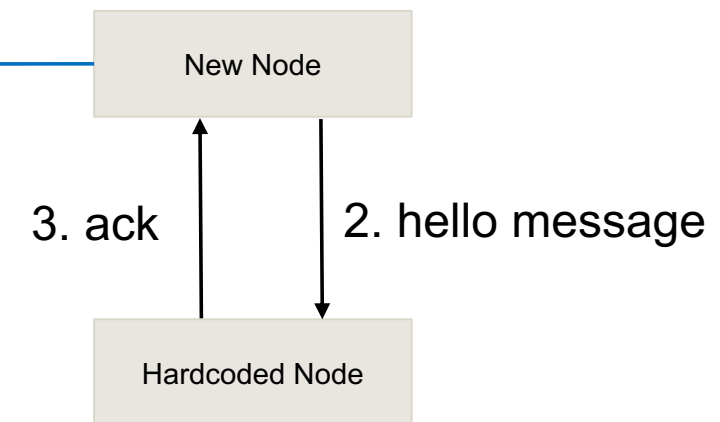
Both Geth and OpenEthereum have a maintained list of default peers hardcoded into their source code. Otherwise, it would be possible that no nodes are found, and the sync will always fail.

The Geth client comes with 6 hardcoded peers:

```
var MainnetBootnodes = []string{
    // Ethereum Foundation Go Bootnodes
    "enode://a979fb...57549@52.16.188.185:30303", // IE
    "enode://4c1d22...de0a99@13.93.211.84:30303", // US-WEST
    "enode://431217...efd0a99@191.235.84.50:30303", // BR
    "enode://1fddd...23d0a99@13.75.154.138:30303", // AU
    "enode://992aac...12a0a99@52.74.57.123:30303", // SG

    // Ethereum Foundation C++ Bootnodes
    "enode://979b7f...37f9@5.1.83.226:30303", // DE
}
```

1. Looks up



Once a node is selected, a hello message is sent to make an initial connection with the node.