Ethereum Basics

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline

History

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 **24th of January 2014** at the Bitcoin conference in Miami

- On the **7th of July 2014**, Buterin **announced the start** of the **public crowd sale**

- The **sale lasted 42 days** until 2nd 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

06 Ethereum Basics - Öz, B., Hoops, F., Gallersdörfer, U., & Matthes, F. (2022). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

History of Ethereum http://ethdocs.org/en/latest/introduction/history-of-ethereum.html

CC BY-SA 4.0   3

Fun Fact

"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
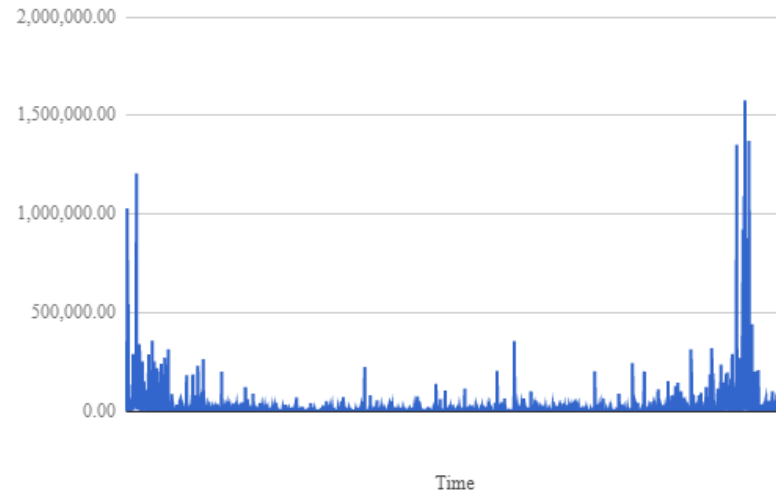
ETH sold

Cumulative ETH sold
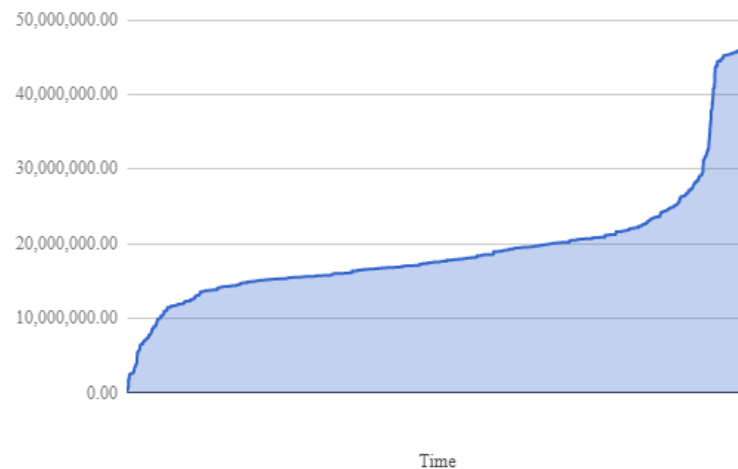
- Around **48 million ETH** were **sold during** the **first price period** of 14 days

- **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/0x5abfec25f74cd88437 631a7731906932776356f9)

White Paper

- **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

Yellow Paper

ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

EIP-150 REVISION

DR. GAVIN WOOD
FOUNDER, ETHEREUM & ETHCORE
GAVIN@ETHCORE.IO

ABSTRACT. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, not least Bitcoin. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

1. INTRODUCTION

With ubiquitous internet connections in most places of the world, global information transmission has become incredibly cheap. Technology-rooted movements like Bitcoin have demonstrated, through the power of the default, consensus mechanisms and voluntary respect of the social contract that it is possible to use the internet to make a decentralised value-transfer system, shared across the

information is often lacking, and plain old prejudices are difficult to shake.

Overall, I wish to provide a system such that users can be guaranteed that no matter with which other individuals, systems or organisations they interact, they can do so with absolute confidence in the possible outcomes and how those outcomes might come about.

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**

Foundation

*"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

Official website of the Ethereum foundation https://www.ethereum.org/foundation

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

Confirmed Transactions Per Day

The total number of confirmed transactions per day.

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

- **439.549** peak of transactions per day[1]

[1]The given graph shows 7-day average.

Ethereum transaction chart by Etherscan: https://etherscan.io/chart/tx
Bitcoin transaction chart by Blockchain.com: https://www.blockchain.com/charts/n-transactions

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

Blockchain.com Wallets

The total number of unique Blockchain.com wallets created.
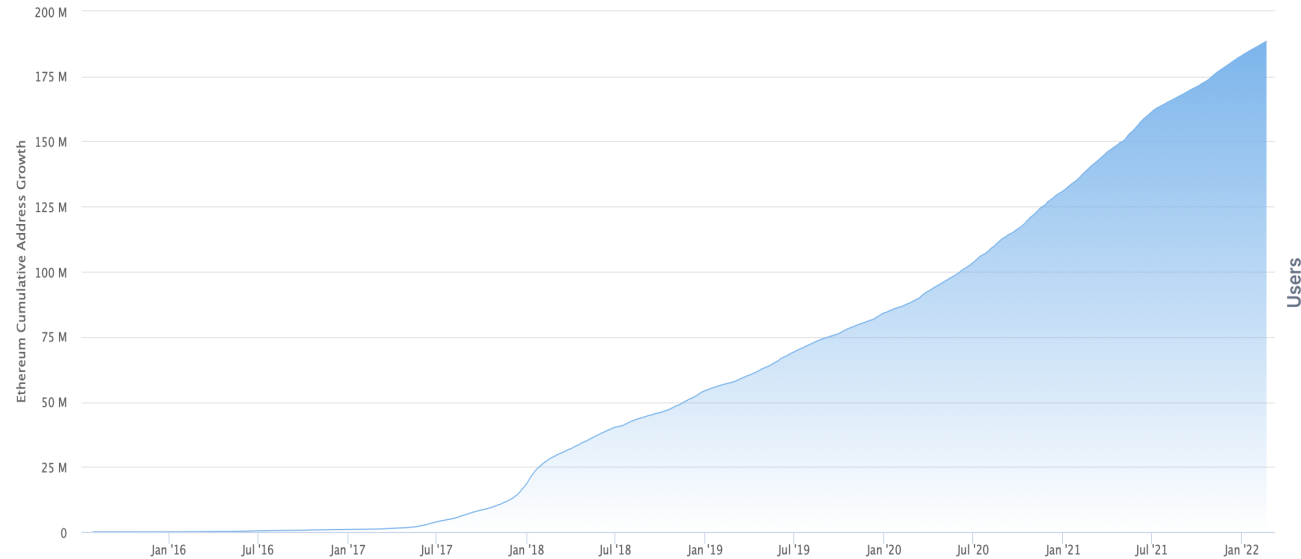
- Currently around **188 million unique wallets** with at least one incoming or outgoing transaction

- 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**[1]

[1] Remember: Bitcoins annual electicity consumption is 204.50 TW/h

Energy consumption estimation by Digiconomist: https://digiconomist.net/ethereum-energy-consumption
Ethereum network hash rate chart by Etherscan: https://etherscan.io/chart/hashrate

Outline

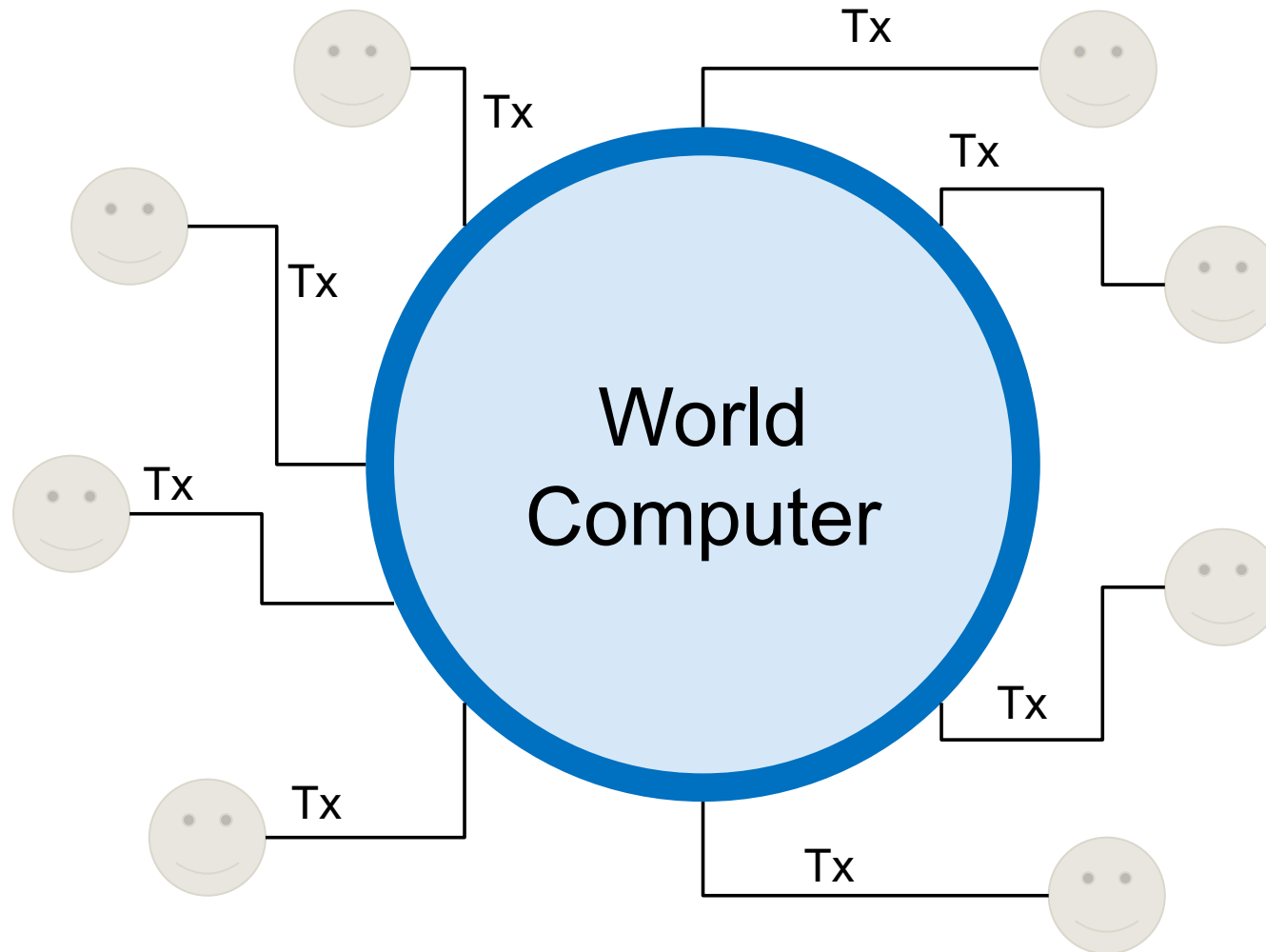1. Ecosystem
    - Historical Overview
    - Crowdsale Statistics
    - Technical Papers
    - Foundations
    - Network Metrics

2. System Architecture
    - Concept of a World Computer
    - EVM
    - Accounts
    - Blockchain Properties
    - Smart Contracts

3. Network Architecture
    - Overview
    - Node Types
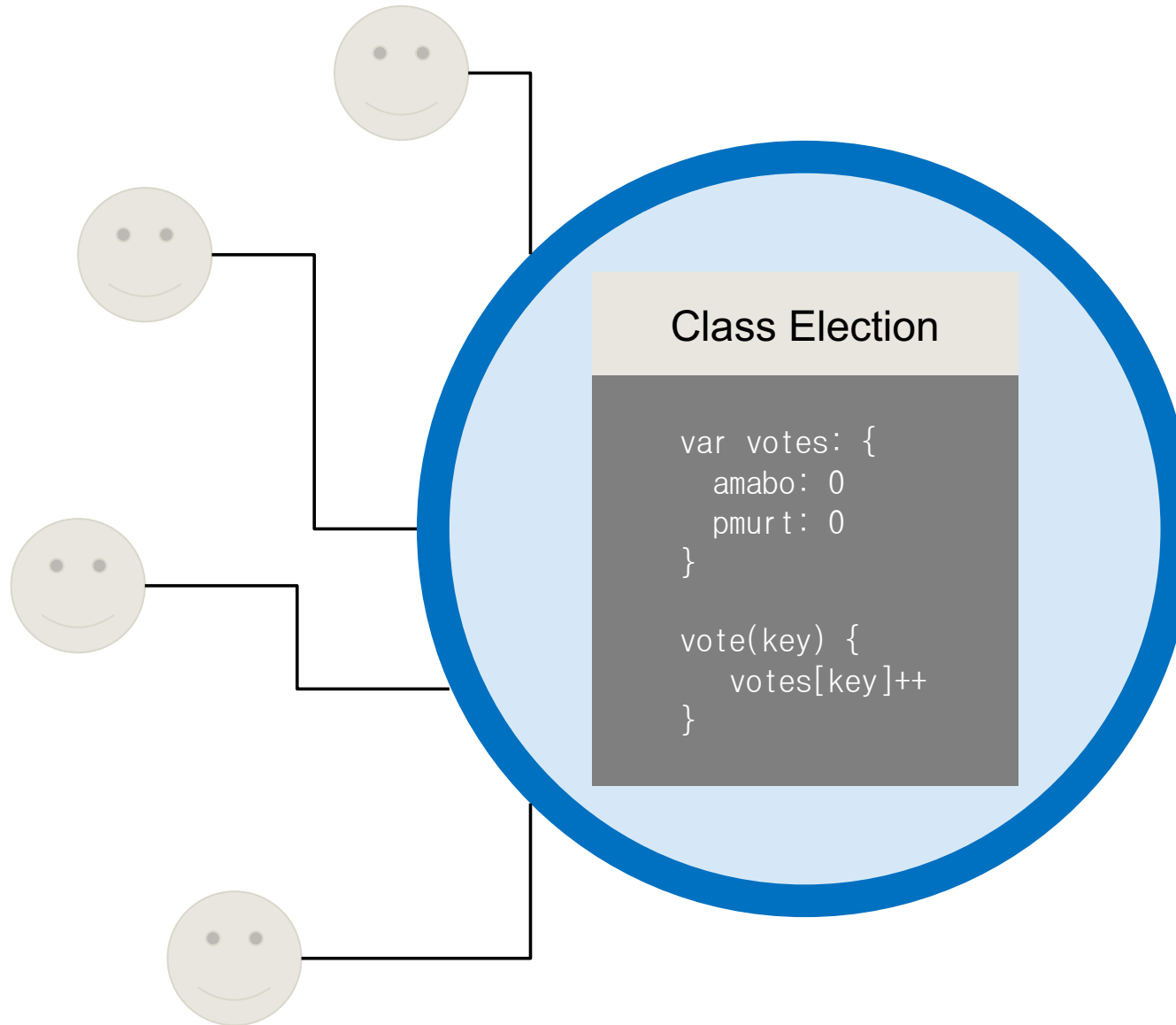   - Clients
        - Geth
        - OpenEthereum

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**

06 Ethereum Basics - Öz, B., Hoops, F., Gallersdörfer, U., & Matthes, F. (2022). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

CC BY-SA 4.0   13

Election Example Using a World Computer

State of the world

State 0 (initial)
(nothing happened yet)

Class Election

```
var votes: {
    amabo: 0
    pmurt: 0
}

vote(key) {
    votes[key]++
}
```

Election Example Using a World Computer (cont.)

`WorldComputer.execute(Election.vote("amabo"))`

State of the world

State 0 (initial)
(nothing happened yet)

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

Election

```
var votes: {
    amabo: 1
    pmurt: 0
}

vote(key) {
    votes[key]++
}
```

Election Example Using a World Computer (cont.)

`WorldComputer.execute(Election.vote("amabo"))`

Election

```
var votes: {
    amabo: 2
    pmurt: 0
}

vote(key) {
    votes[key]++
}
```

State of the world
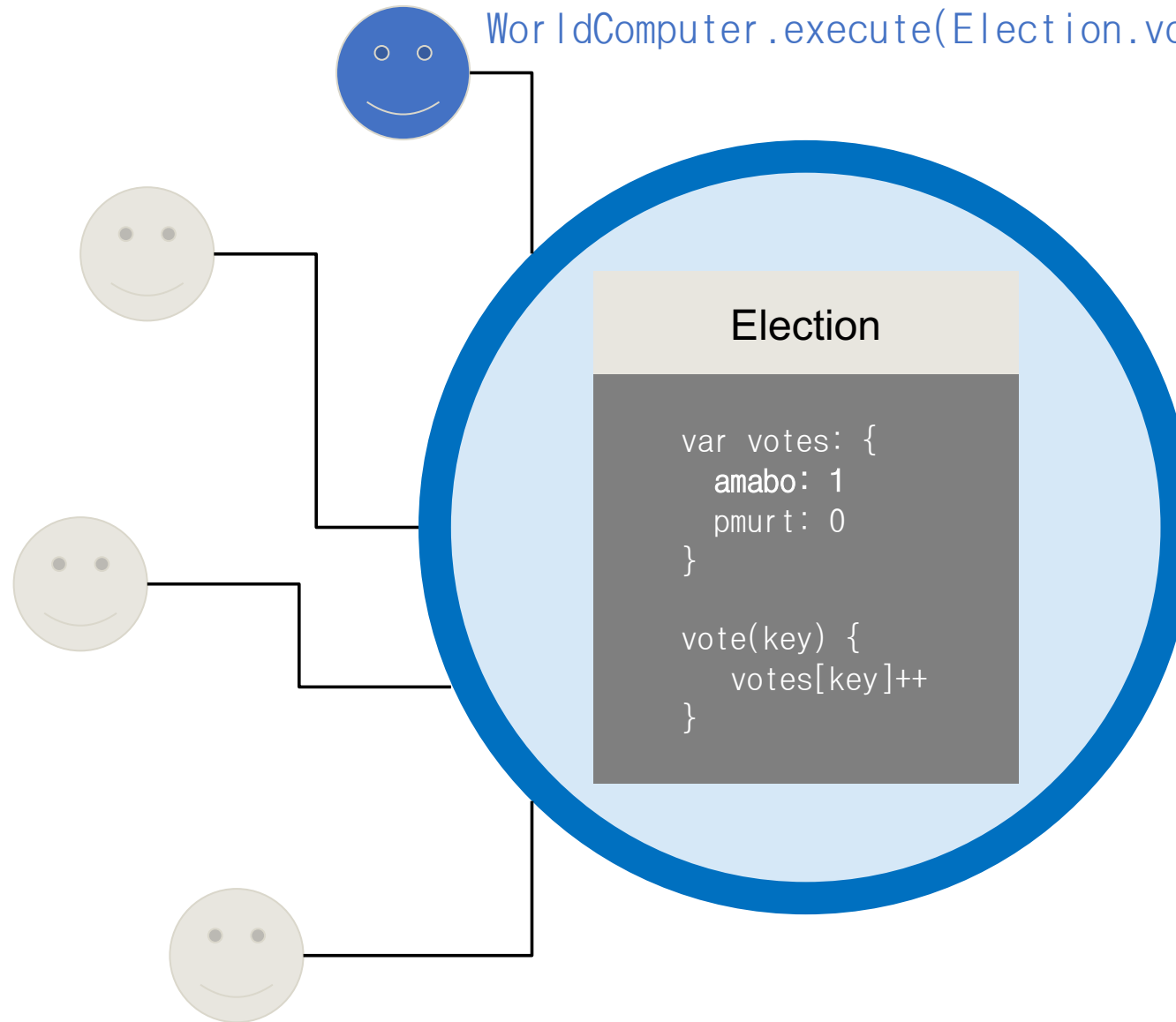
State 0 (initial)
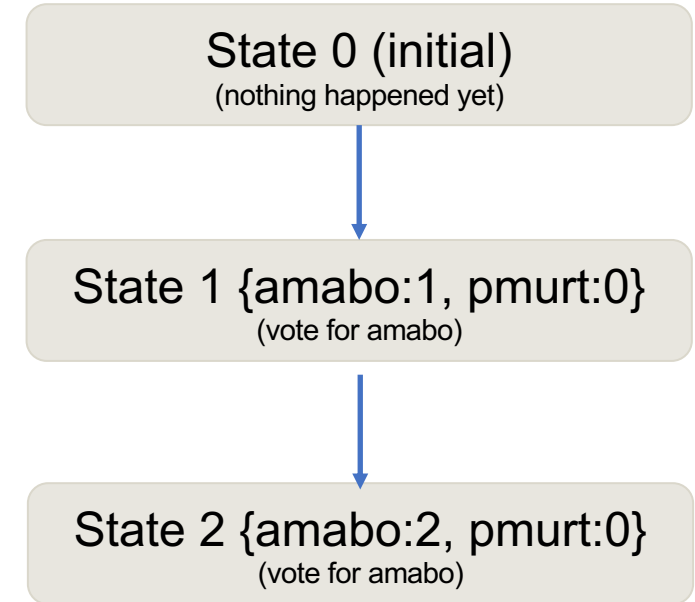(nothing happened yet)

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

State 2 {amabo:2, pmurt:0}
(vote for amabo)

Election Example Using a World Computer (cont.)


```
Election

var votes: {
    amabo: 2
    pmurt: 1
}

vote(key) {
    votes[key]++
}
```

`WorldComputer.execute(Election.vote("pmurt"))`

State of the world

State 0 (initial)
(nothing happened yet)

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

State 2 {amabo:2, pmurt:0}
(vote for amabo)

State 3 {amabo:2, pmurt:1}
(vote for pmurt)

Election Example Using a World Computer (cont.)

TUM
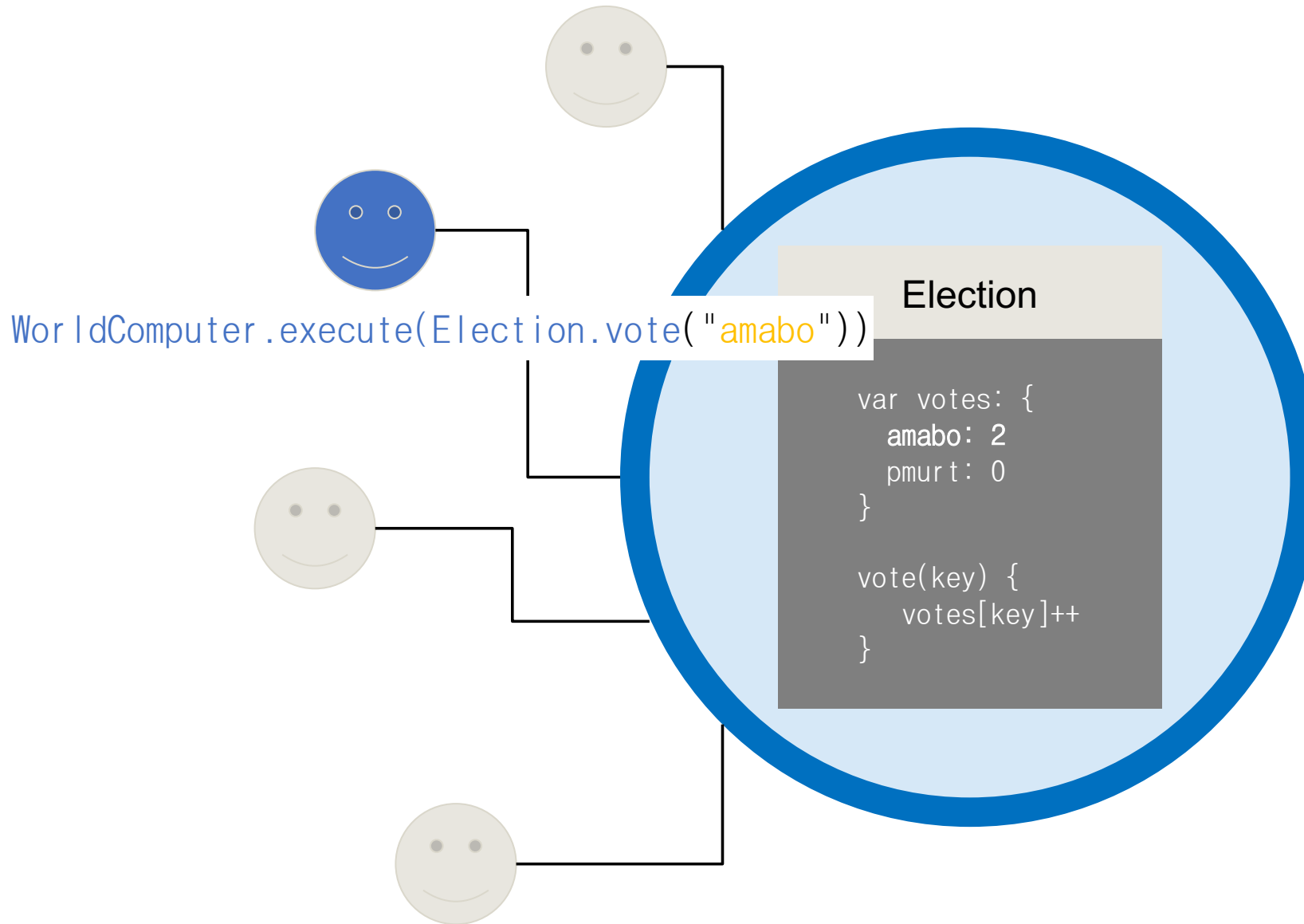
State of the world


```
Election

var votes: {
    amabo: 3
    pmurt: 1
}

vote(key) {
    votes[key]++
}
```

WorldComputer.execute(Election.vote("amabo"))

State 0 (initial)
(nothing happened yet)

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

State 2 {amabo:2, pmurt:0}
(vote for amabo)

State 3 {amabo:2, pmurt:1}
(vote for pmurt)

State 4 {amabo:3, pmurt:1}
(vote for amabo)

Blockchain as a State Machine

State of the world

Blockchain

State 0 (initial)
(nothing happened yet)

Tx

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

Tx

State 2 {amabo:2, pmurt:0}
(vote for amabo)

Tx

State 3 {amabo:2, pmurt:1}
(vote for pmurt)

Tx

State 4 {amabo:3, pmurt:1}
(vote for amabo)

Transactions lead to a new state (block added)

Genesis Block

Genesis Block — Block 1

Genesis Block — Block 1 — Block 2

Genesis Block — Block 1 — Block 2 — Block 3

Genesis Block — Block 1 — Block 2 — Block 3 — Block 4
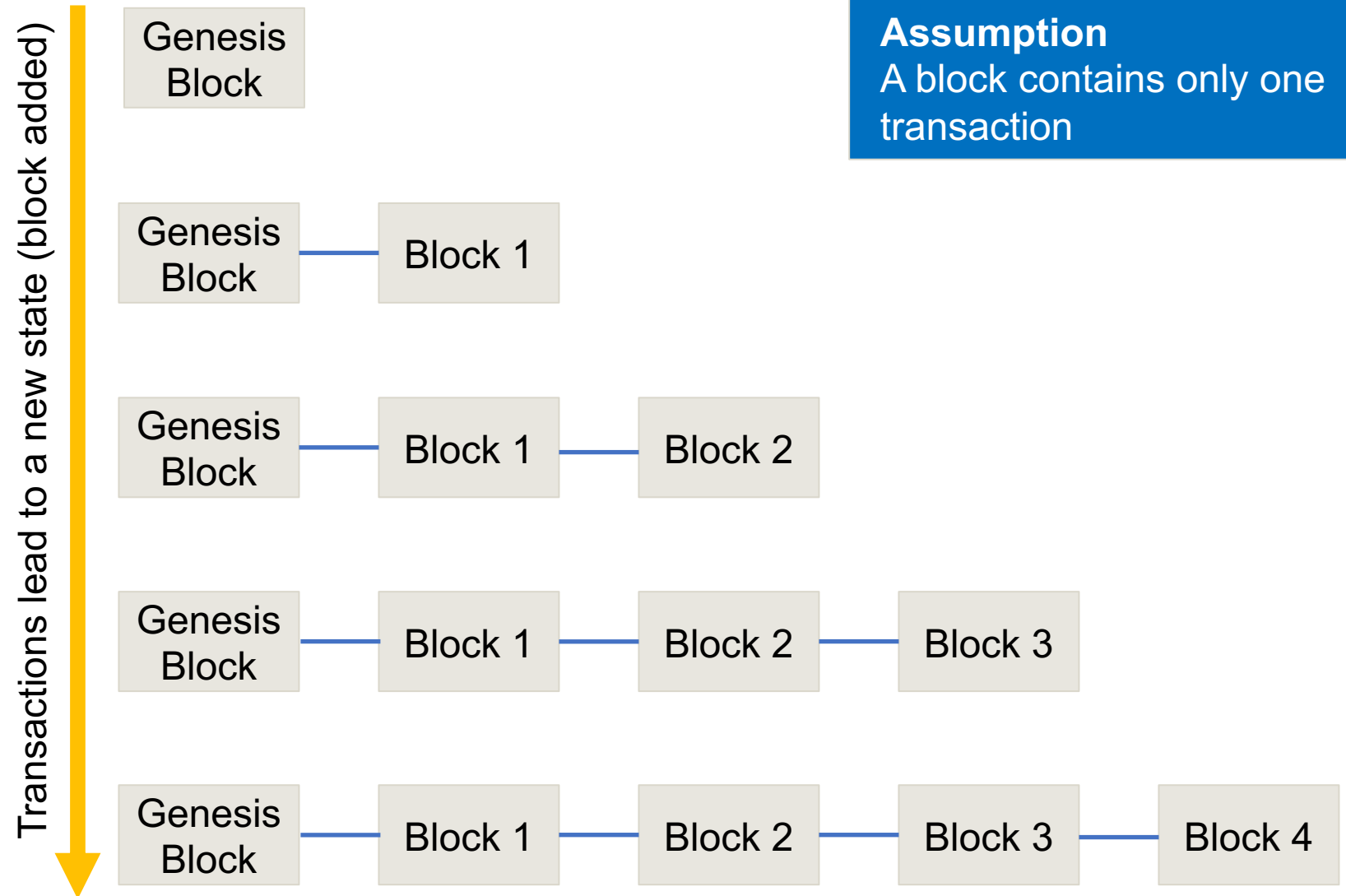
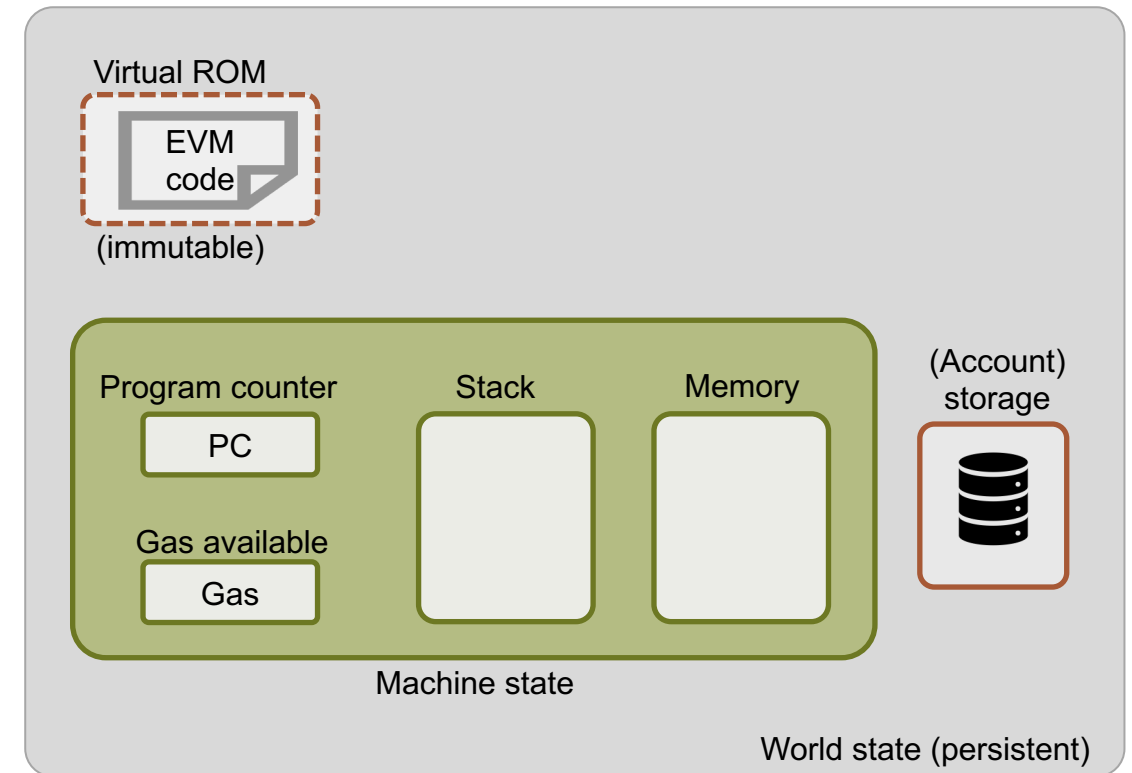# Brief Insight into the Ethereum Virtual Machine (EVM)

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)

Virtual ROM

EVM code

(immutable)

Program counter
PC

Gas available
Gas

Stack

Memory

(Account) storage

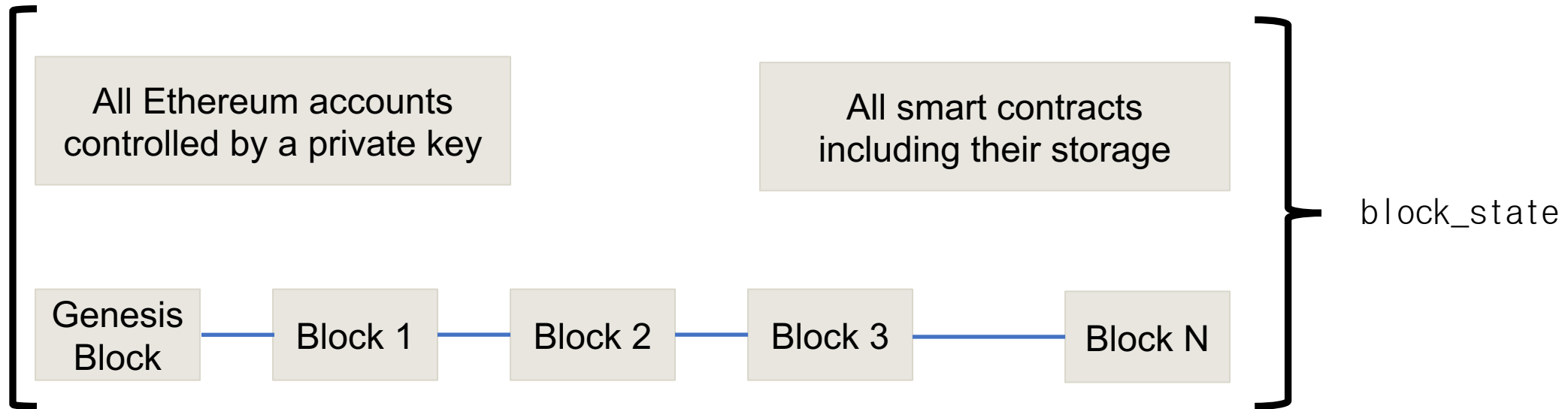Machine state

World state (persistent)

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**

| All Ethereum accounts controlled by a private key | | | All smart contracts including their storage | |

block_state

| Genesis Block | Block 1 | Block 2 | Block 3 | Block N |

Ethereum Account Types

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**[1]. 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[2] accounts which are controlled by their code**
   - Smart contracts are treated as **account** entities with their **own**, unique **address**.
   - Contracts[3] **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.

[1]In this context, a user is an human being. Thus, EOAs are controlled by humans.
[2]Deeper overview on smart contracts can be found in the following slides.
[3]"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[1]* 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

```
EOA  --Transaction-->  Smart Contract
```

Type 2: EOA to EOA
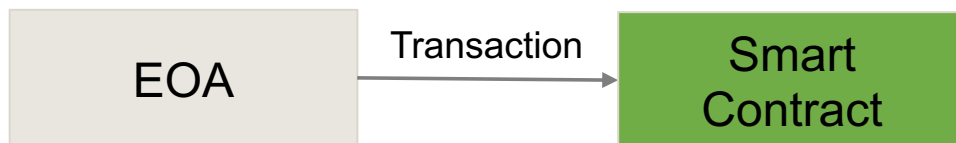
```
EOA  --Transaction-->  EOA
```
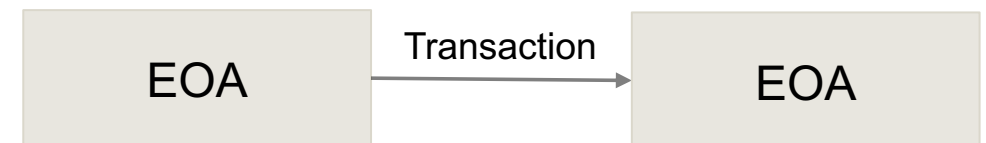
[1]Same terminology is also used for defining the maximum amount of gas a block uses.

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.
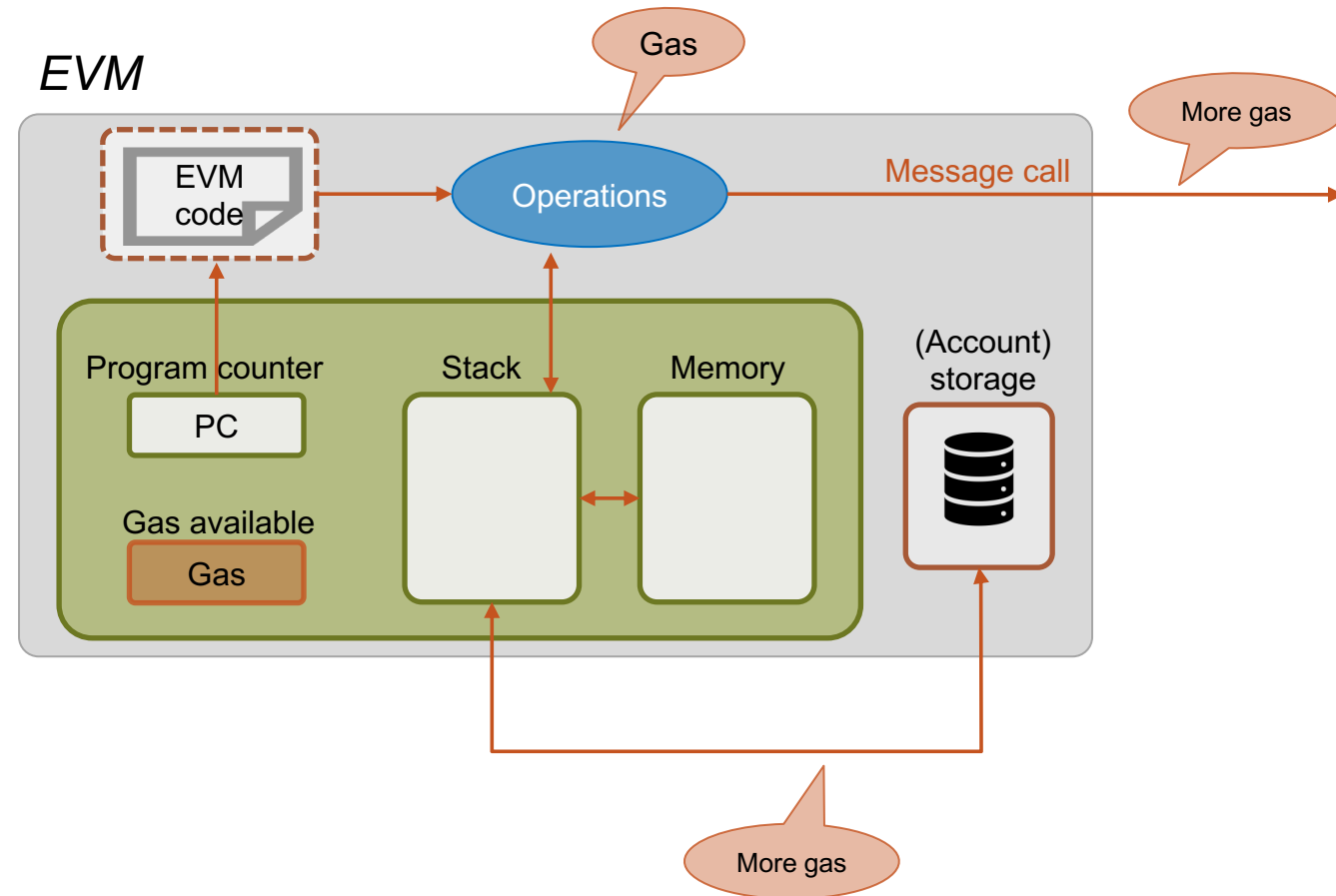
pc

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
```

Program Counter (2)
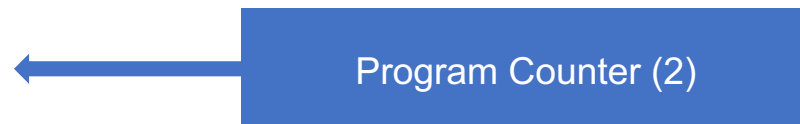
- **Every executed opcode instruction** uses a miner's computational resources and therefore **costs a certain fee (called gas)**.

- 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.

EVM

Gas

Operations

Message call

More gas

EVM code

Program counter — PC

Gas available — Gas

Stack

Memory

(Account) storage

More gas

The Economic Side of Ethereum

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 5th, 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 5th, 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

⑦ Value:	0.11670416 Ether ($311.47)
⑦ Transaction Fee:	0.001158913272762 Ether ($3.09)
⑦ Gas Price:	0.000000055186346322 Ether (55.186346322 Gwei)
⑦ Ether Price:	$2,597.94 / ETH
⑦ Gas Limit & Usage by Txn:	21,000	21,000 (100%)
⑦ Gas Fees:	Base: 55.086346322 Gwei	Max: 57.07 Gwei	Max Priority: 0.1 Gwei
⑦ Burnt & Txn Savings Fees:	🔥 Burnt: 0.001156813272762 Ether ($3.09) 🦦 Txn Savings: 0.000039556727238 Ether ($0.11)
⑦ 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 25th 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.[1]

    - The **transactions** in the **uncle block** are considered **invalid.**

[1]Bitcoin's block time is 10 minutes in average.

For further resources, see *https://www.notion.so/BBSE-Additional-Content-85f1424258e64f679d3a6efac0d8c683*

Brief Insight into Smart Contracts

- 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.**

Fun Fact

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".
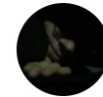
💬 85   ⟲ 301   ♡ 766   ↥

Vlad Zamfir   ···
@VladZamfir

Replying to @VitalikButerin @CleanApp and 4 others
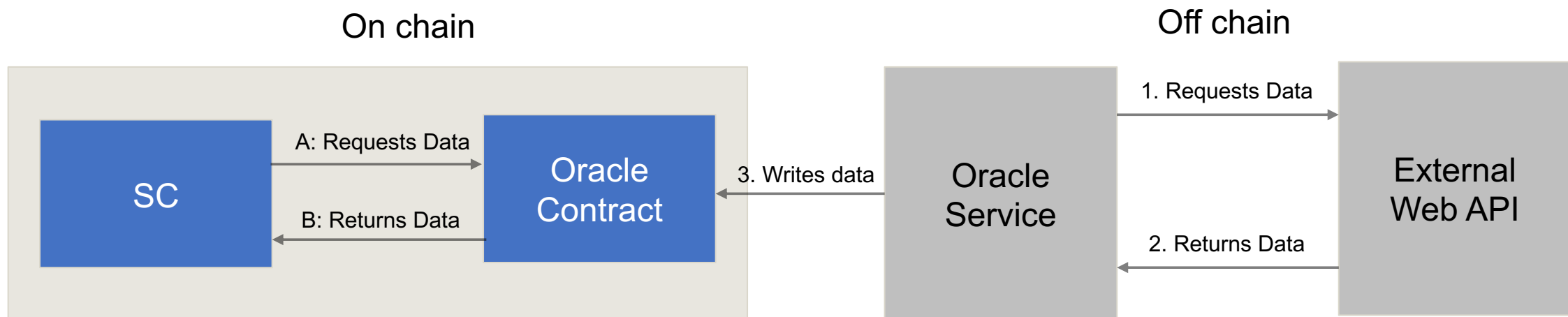
I like "stored procedures"

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.

On chain   Off chain

```
┌─────────────────────────────────────────┐      ┌──────────────┐      ┌──────────────┐
│                    A: Requests Data      │      │              │  1. Requests Data │              │
│   ┌──────────┐  ──────────────▶ ┌──────┐│      │   Oracle     │ ───────────────▶ │  External    │
│   │    SC    │                  │Oracle ││◀─────│   Service    │                  │  Web API     │
│   │          │  ◀────────────── │Contract│ 3. Writes data     │ ◀─────────────── │              │
│   └──────────┘   B: Returns Data└──────┘│      │              │  2. Returns Data  │              │
└─────────────────────────────────────────┘      └──────────────┘      └──────────────┘
```

Brief Insight into Solidity

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

```solidity
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)
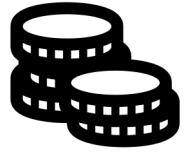
Non-Fungible Tokens (NFTs)

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 unliterally 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.
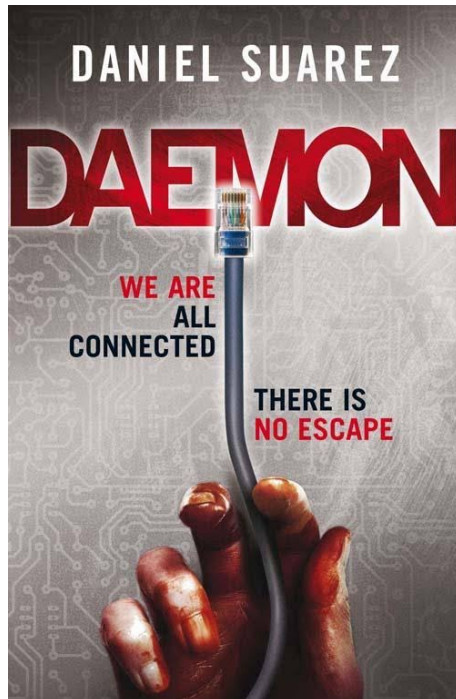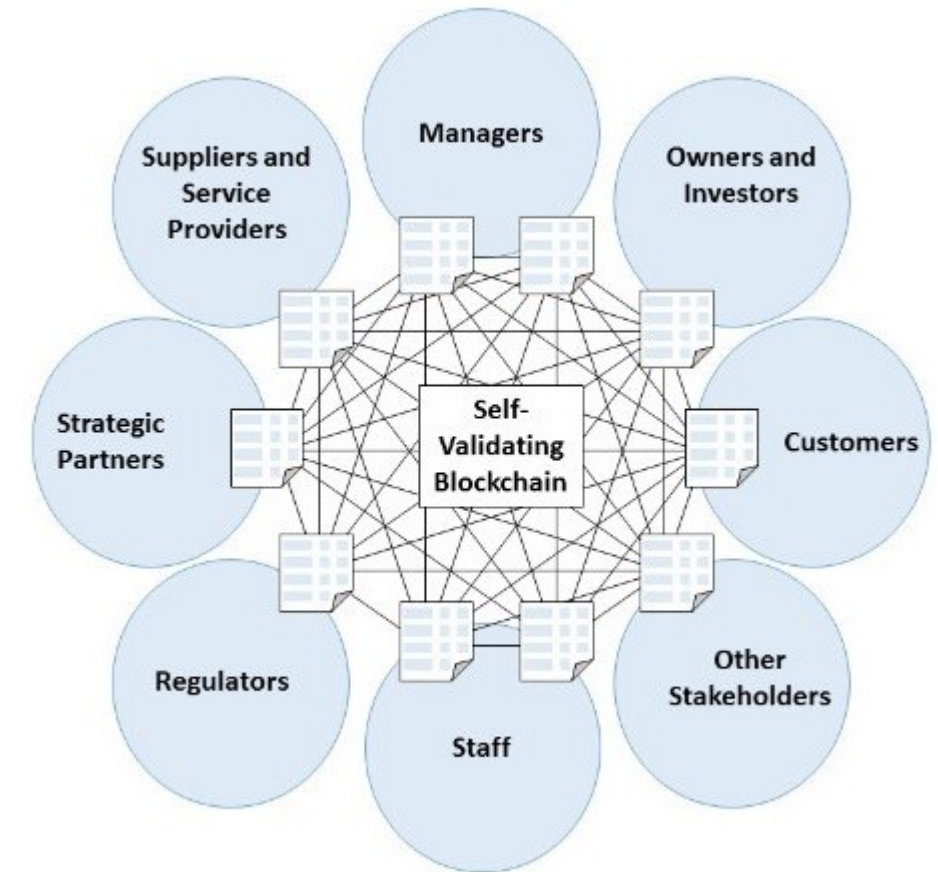
Outline

Network Architecture Overview

Node Types
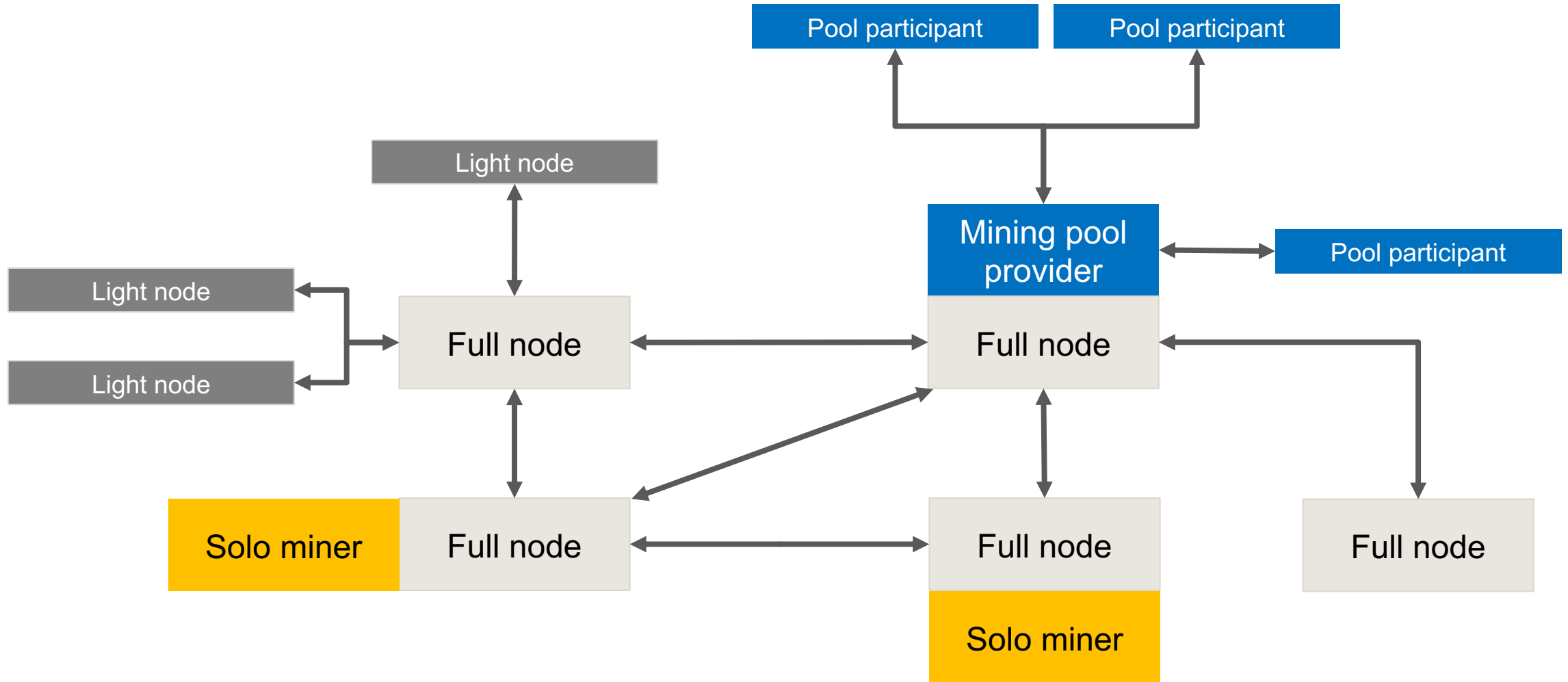
Full nodes
Full nodes are the foundation of the Ethereum network. Each full node holds a copy of the entire blockchain and syncs it with other nodes. Transactions must be sent to a full node which distributes it among the network participants.

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.

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.
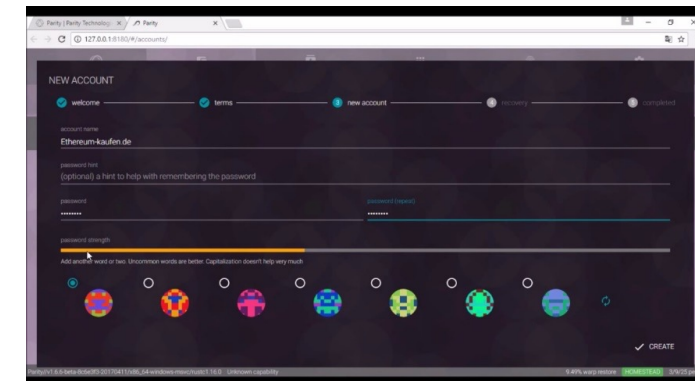
The Geth node is available at https://geth.ethereum.org/
The OpenEthereum node is available at https://github.com/openethereum/openethereum
Announcement by Parity Technologies: Transitioning Parity Ethereum to OpenEthereum DAO

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://1fdddd...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 → New Node

3. ack   2. hello message

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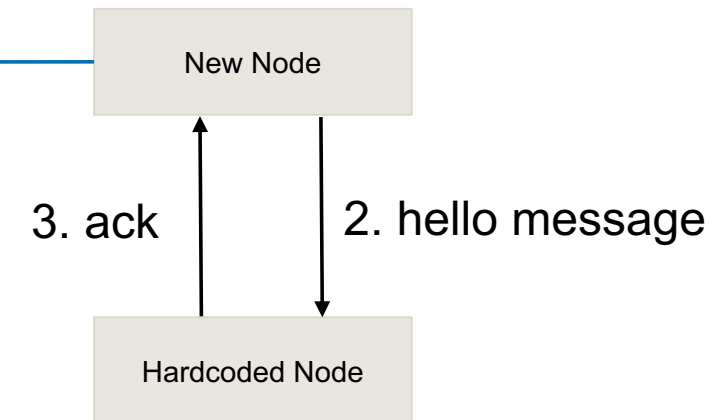
Once a node is selected, a hello message is sent to make an initial connection with the node.