Decentralized Systems

Ethereum

Blockchains

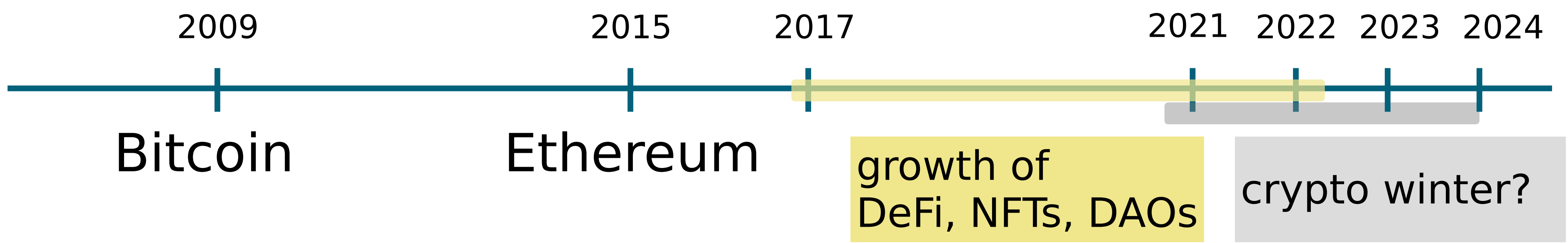
- Public **append-only** data structure, secured by replication and incentives
- Provides **coordination between many parties**, when there is **no single trusted party**
- **Fixed supply asset** (21 million BTC) used for digital payments, and more

Blockchains

- Blockchain computer: a **fully programmable environment**

  $\Longrightarrow$ public programs that manage digital and financial assets
- Composability: applications running on chain can call each other
- *The "slowest computer" in the world*

Blockchains



- **Crypto winter**: prolonged period of pricing weakness in the cryptocurrency market
  - Current crypto winter began in **late 2021**, and it is difficult to say when it will end
  - Need for new killer applications?
  - The market will eventually recover and cryptocurrencies will continue to play a role in the global financial system?

Ethereum

Ethereum is an open source, public, blockchain-based distributed computing platform and operating system featuring smart contract functionality

Vitalik Buterin (https://vitalik.ca/)

Launched on 30 July 2015 with 11.9 million coins pre-mined thanks to a crowd sale

Ethereum

- Designed to create, develop and spread **Smart Contracts**
- Uses its own ledger, different from the Bitcoin blockchain
- Implements a built-in **powerful scripting language** (Turing-complete)

Ethereum cryptocurrency

- The **native currency** of the Ethereum blockchain is called **Ether**
  - listed under the diminutive **ETH** and traded on cryptocurrency exchanges
  - also used to **pay for transaction fees and computational services** on the Ethereum network

- **Oct. 15, 2023:** 1 ETH is $1,554.19 USD (CoinMarketCap)
- **Oct. 15, 2024:** 1 ETH is $2,583.30 USD

Ethereum cryptocurrency

- **Ethereum Classic** (**ETC**) is another blockchain/cryptocurrency **created in 2016** as a result of a **hack of The DAO** (more later on this)
- Theft of over $60 million ETH, and the Ethereum community was divided on how to respond:
  - **reverse the hack** and return the stolen funds (Ethereum)
  - **code is law**, it is not possible to violate the principle of immutability (Ethereum Classic)
- Ethereum Classic is the original Ethereum blockchain, with a smaller community and market capitalization
  - **Oct. 15, 2023:** 1 ETC is $14.99 USD (CoinMarketCap)
  - **Oct. 15, 2024:** 1 ETC is $19.32 USD

Ethereum cryptocurrency

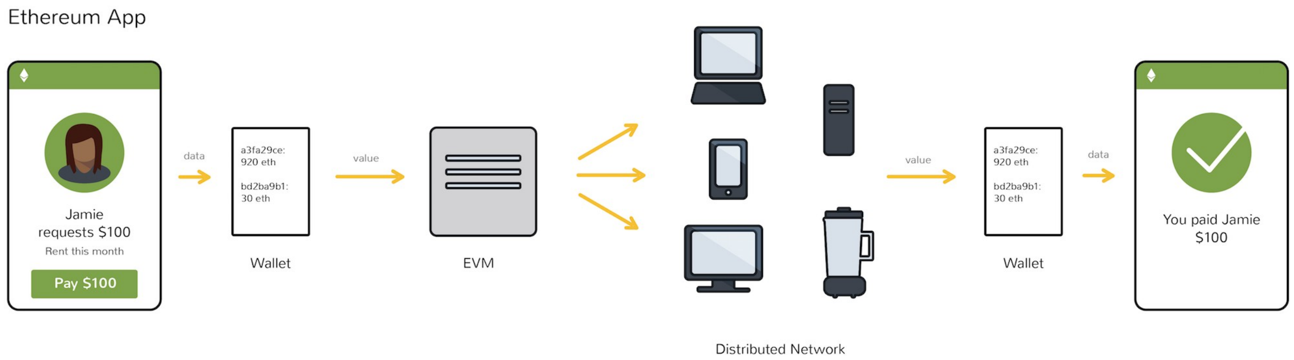
- Many other cryptocurrencies are based on Ethereum
- These are known as **Ethereum tokens**, and they are created and deployed on the Ethereum blockchain. Ethereum tokens can be used for a variety of purposes, such as representing assets, paying for goods and services, or granting voting rights (more later on this)

Ethereum denominations

Unit	Denominations	
Wei	1	1
Kwei	1,000	10^3
Mwei	1,000,000	10^6
Gwei	1,000,000,000	10^9
Szabo	1,000,000,000,000	10^12
Finney	1,000,000,000,000,000	10^15
Ether	1,000,000,000,000,000,000	10^18
KEther	1,000,000,000,000,000,000,000	10^24
MEther	1,000,000,000,000,000,000,000,000	10^24
GEther	1,000,000,000,000,000,000,000,000,000	10^27
TEther	1,000,000,000,000,000,000,000,000,000,000	10^30

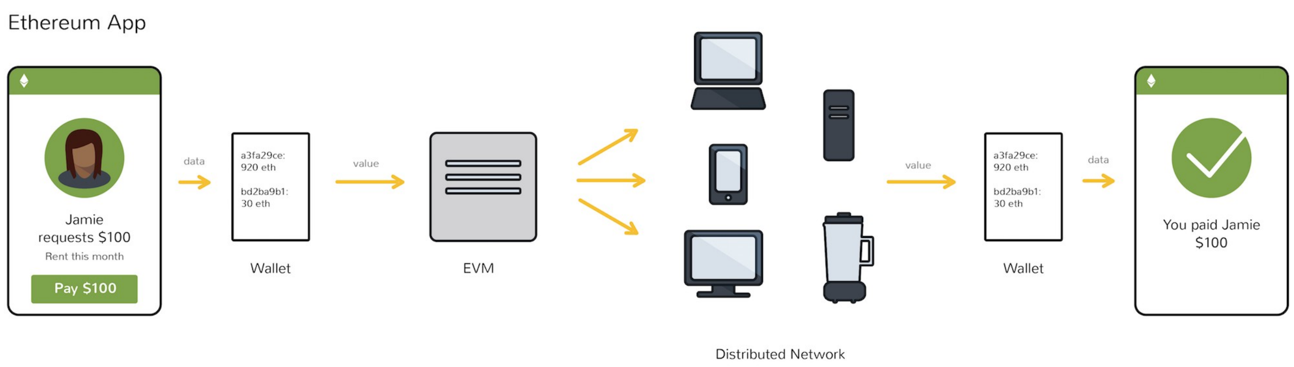
https://www.geeksforgeeks.org/what-are-the-different-units-used-in-ethereum/

Ethereum Virtual Machine



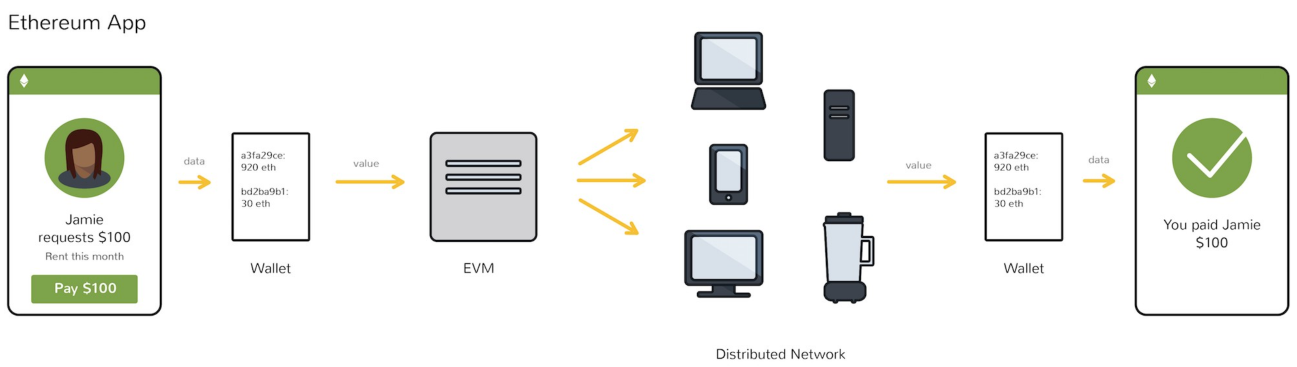
- The entire Ethereum **P2P network** is a mass of nodes (computers) connected to one another, all understanding the same protocols
- Can be visualized as a **single** (giant) **entity** called the Ethereum Virtual Machine or **EVM**
- The EVM provides the environment in which users can exchange cryptocurrencies and smart contracts can be deployed and run

Ethereum Virtual Machine



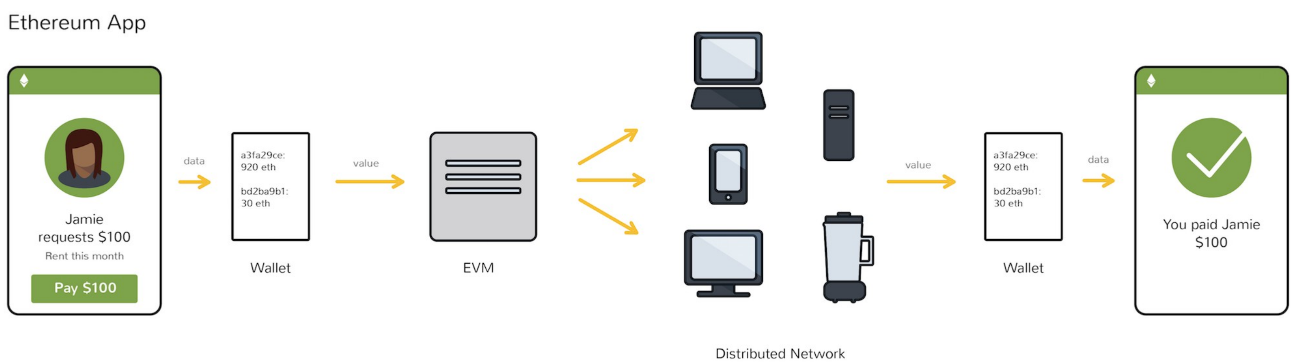
- **Smart contracts** are written in a variety of programming languages, and **compiled into EVM bytecode**, which is the machine-readable code executed by the EVM
- The bytecode has its own **instruction set**, which is used to perform operations such as arithmetic, logical operations, and memory access, including the **JUMP instruction** which is used to implement a variety of control flow statements such as loops, conditional statements, and functions

Ethereum Virtual Machine



- The **state of Ethereum** is a **snapshot of the blockchain** at a given point in time. It includes all account balances, smart contract storage, and other data necessary to execute transactions and run smart contracts
- A **change of state** occurs when a **transaction is successfully processed and added to the blockchain**. This can change the accounts' balances, the storage of smart contracts, or other data in the state

Ethereum Virtual Machine



- The vision of the Ethereum project is to have **one computer distributed across the entire Internet**
- **Each full node joining the Ethereum network** must **keep in its memory** the **whole state** of this Ethereum computer. This state will include
  - all smart contracts bytecode
  - all input and output to the smart contracts (past and present)
  - all communications among smart contracts

Ethereum accounts

- Two types of accounts
  - User account (Externally Owned Account, **EOA**)
  - Contract account (**CA**)
- **EOAs** are similar to Bitcoin accounts
  - 20-byte **address** (hash(S_k))
  - controlled by **key pairs** (ECDSA)
  - **balance**
  - can send messages by creating and signing **transactions**
  - **nonce** (number of transactions coming from that address)

Ethereum accounts

- **CAs** (Contract Accounts)
  - 20-byte **address** (hash(CreatorAddr, CreatorNonce))
  - controlled by their **contract code**
  - can **read** and **write** to internal storage, **send messages** to other contracts or **create new contracts**
  - **nonce** (number of creations of contract with that address)

UTXO vs Account model

- Bitcoin uses the **UTXO model** which is based entirely on **individual transactions**, grouped in blocks
- The **global state** is represented with the **entire graph** of transactions
- Account balances are calculated on the client-side (wallet) by adding the UTXOs
- The verification process checks if transaction output is unspent

https://www.horizen.io/academy/utxo-vs-account-model/

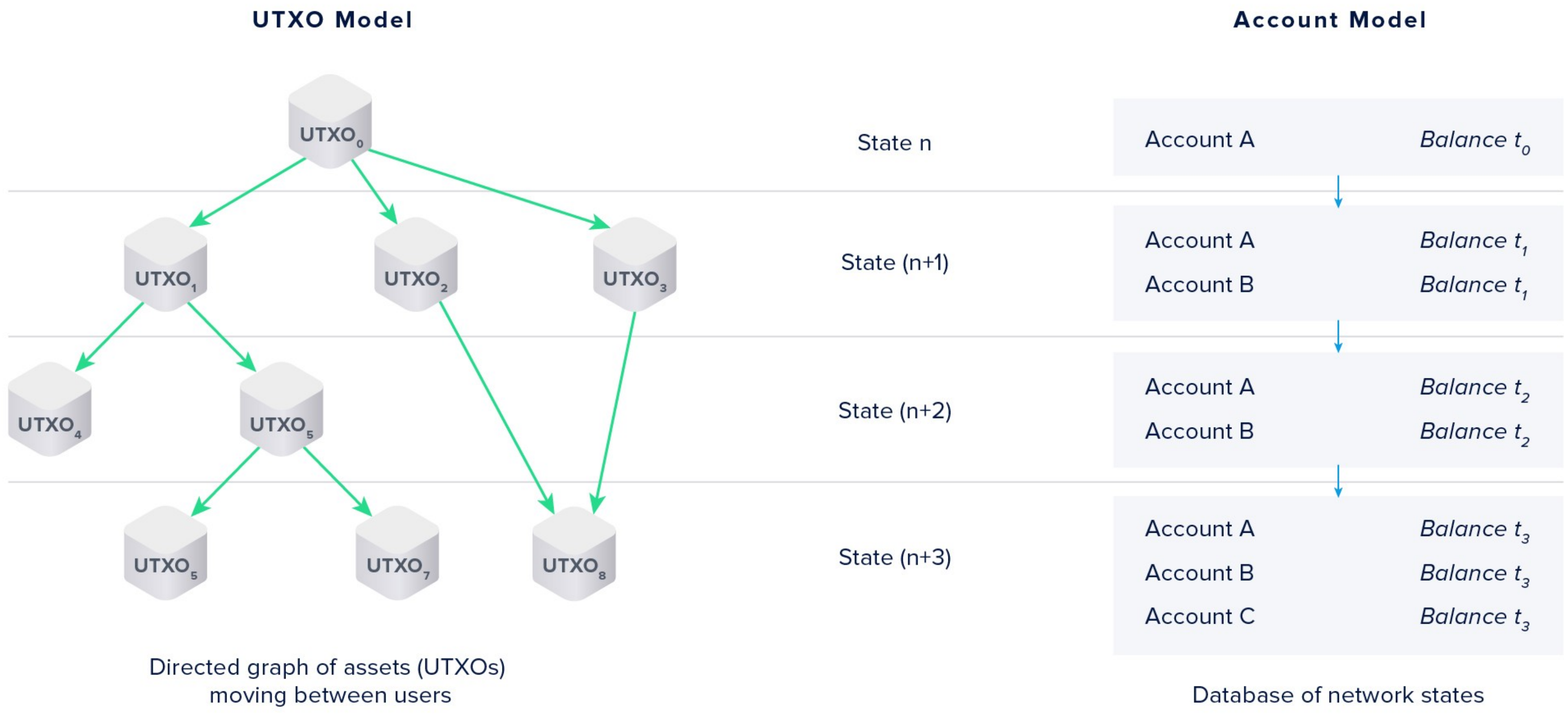
UTXO vs Account model

- Ethereum uses the **account model**, based on balances within accounts, similar to bank accounts
- A transaction in this model triggers nodes to **decrement the balance of the sender**’s account and **increment the balance of the receiver**’s account
- To prevent **replay attacks**, e.g., a malicious node retransmits a valid transaction that has already been executed, when a **wallet** sends a transaction, it **increments the nonce** for the sender's account
- This ensures that each transaction has a unique identifier

UTXO vs Account model

The first significant difference between the two balance models is how the state of the system is recorded

HORIZEN ACADEMY



Transactions

- Transactions are used to send to the EVM
  - **payments**, **smart contracts**, **calls to contracts methods**
- Many fields
  - timestamp
  - sender address and signature
  - receiver address
  - amount of Ether to transfer
  - gas detail
  - optional data
  - nonce

Smart contract: deployment

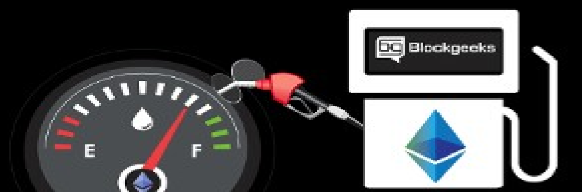
- To deploy a smart contract, a **transaction** that contains EVM bytecode in its data field is **sent to address 0**
- The contract will be accessible under an **address** that is **derived** from the deploy **transaction sender's address** and their **nonce** (the count of how many transactions they have sent)

Smart contract: validity 

- Smart contracts do what they are programmed to do
  - the code **automatically executes exactly as programmed** without possibility of downtime, censorship or third-party interference
- If a "buggy" smart contract has a bad behavior or a contract is deemed invalid in a court, how is this solved?
  - Still an open problem

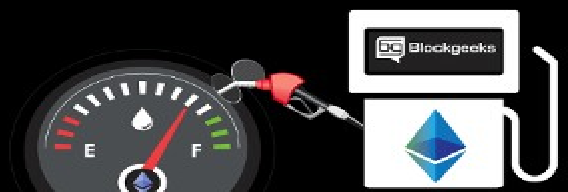
Ethereum gas

- To send a **transaction** or interact with **on chain applications** requires network's computation and users need to **pay a fee** (for miners/validators)
- Fees are commonly referred to as **gas**
  - gas is essentially a measurement of the computational effort needed to execute an operation on Ethereum



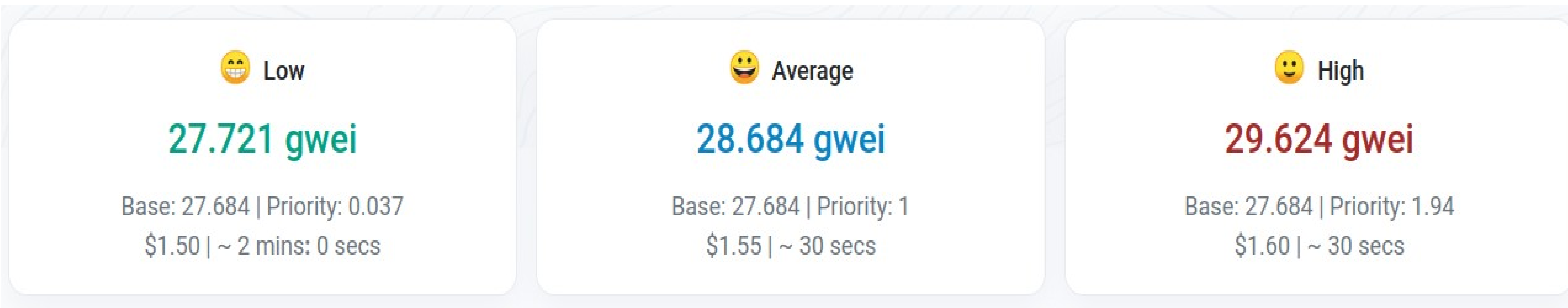
Ethereum gas

- EVM is a **Turing-complete** machine **limited by the amount of gas** required to run any instruction
- The **fee for the execution** depends on the operation performed, for example
  - a single step calculation like if (2 > 1) will cost 1 gas unit
  - a single hash operation will cost 20 gas unit
  - each byte in the data field will cost 5 gas unit



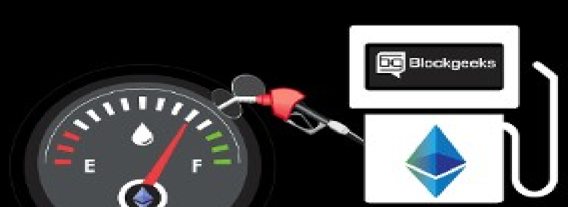
Ethereum gas price

- The **gas price** determines the amount the user pays **per unit of gas** used and does not change the amount of gas needed to execute the transaction



- A sender can specify any gas price they want, according to the current estimation

https://etherscan.io/gastracker



Ethereum gas cost

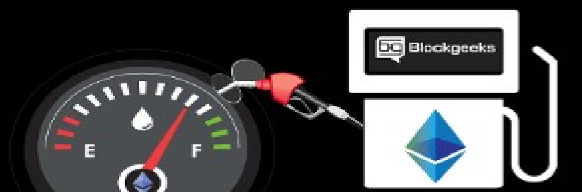
- Once fixed the gas price, the **gas cost** depends on the number of **gas unit** needed to perform operations on chain
- Different transactions require different amount of computation
  - token transfers require relatively small amount of gas: 21,000 unit
  - more complex transactions need a larger amount of gas unit

Ethereum gas limit

- The **gas limit** is the maximum amount of gas that a user is willing to use for a single transaction
- Ethereum users can specify their desired gas limit when sending a transaction
  - changing the gas limit does not change the actual amount of gas that is needed to execute an operation
  - the gas limit is a **safeguard** that protects users from malicious or buggy applications on chain that may try to use too much gas

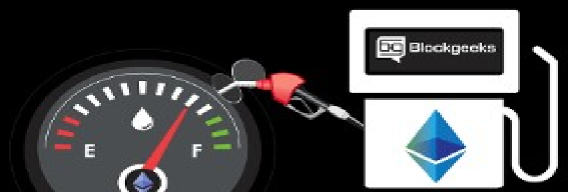
Ethereum gas limit

- When submitting a transaction users (their wallets) need to specify the gas limit
  - if the transaction requires more gas than the gas limit, then the transaction will fail
  - any unused gas below the gas limit is returned to the sender’s wallet
- **October 15, 2023:** the maximum gas limit for Ethereum blocks is **30 million gas** (after London fork)



Ethereum gas auction

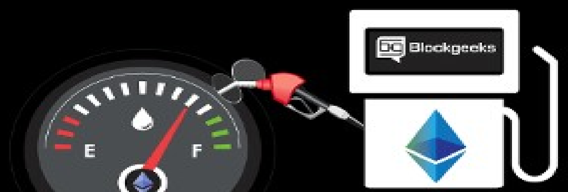
- Why would a user bid to pay a high gas price when they can choose to pay the minimum?



Ethereum gas auction

- Why would a user bid to pay a high gas price when they can choose to pay the minimum?

Higher gas leads to faster transaction confirmation, since miners would prioritize more convenient transactions



Ethereum gas

- **Before** London upgrade (hard fork)
  - Users could decide how much they wanted to pay for their transactions and this led to **high fees**, especially during times of high **network congestion**
  - Example

    The gas limit for transactions is 21,000 gas unit
    The chosen gas fee is 20 Gwei per unit
    Total tx fee (gas limit X gas fee) = 21,000 * 20 = 420,000 Gwei (0.00042 ETH)

London upgrade: EIP-1559, August 2021, block 12965000

Ethereum's London Hardfork Upgrade

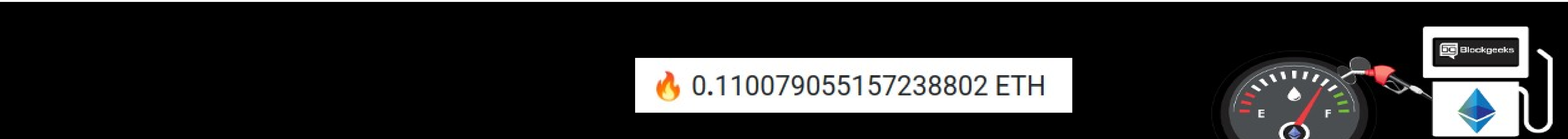
Overview

Block Height:	12965000
Status:	Finalized
Timestamp:	1169 days ago (Aug-05-2021 12:33:42 PM +UTC)
Transactions:	259 transactions and 129 contract internal transactions in this block
Mined by:	Krptex Mining 1 in 10 secs
Block Reward:	2.813216092377312642 ETH (2 + 0.813216092377312642)
Uncles Reward:	0
Difficulty:	7,742,494,561,645,080
Total Difficulty:	28,494,409,340,649,014,490,153
Size:	137,049 bytes
Gas Used:	30,025,257(99.99%)  +100% Gas Target
Gas Limit:	30,029,122

Interpret this chart with IMAGE

Ethereum gas

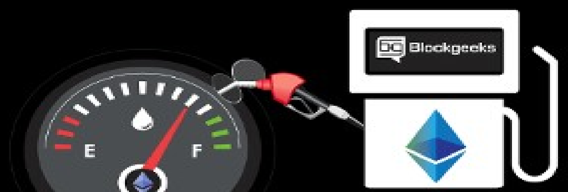
- **After** London upgrade (EIP-1559)
  - Major change to the way fees are computed
  - **Base fee**, e.g., the minimun amount of gas per unit to be paid per each transaction
  - **Adjusted by the protocol**, e.g., when the previous block is filled more than 50%, the base fee of the next block will increase, and vice versa
  - The base fee is **burned** to help ETH to steadily **increase in value**, as supply decreases (deflationary coin)



Ethereum gas

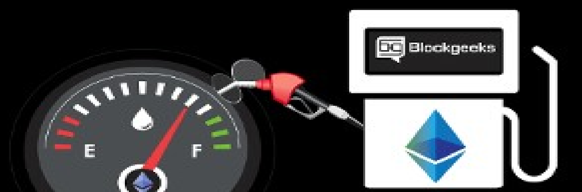
- **After** London upgrade (EIP-1559)
  - Users can add a **priority fee**, or **tip**, to **speed up their transactions**
  - The gas limit per block is doubled, but the target gas usage is set to 50% of the new limit

- Many miners were not happy as this upgrade reduced their rewards for completing a new block



Ethereum gas summary

- Price to be paid for the computation performed by the network
- Incentive ensuring that developers write quality applications
  - Infinite loops cannot run forever: **Out-of-Gas exception** once the gas is exhausted
  - **Wasteful code costs more**



Ethereum Yellow Paper

Appendix G. Fee Schedule

The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{suicide}$	24000	Refund given (added into refund counter) for suiciding an account.
$G_{suicide}$	5000	Amount of gas to pay for a SUICIDE operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SUICIDE operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	10	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead transition*.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.