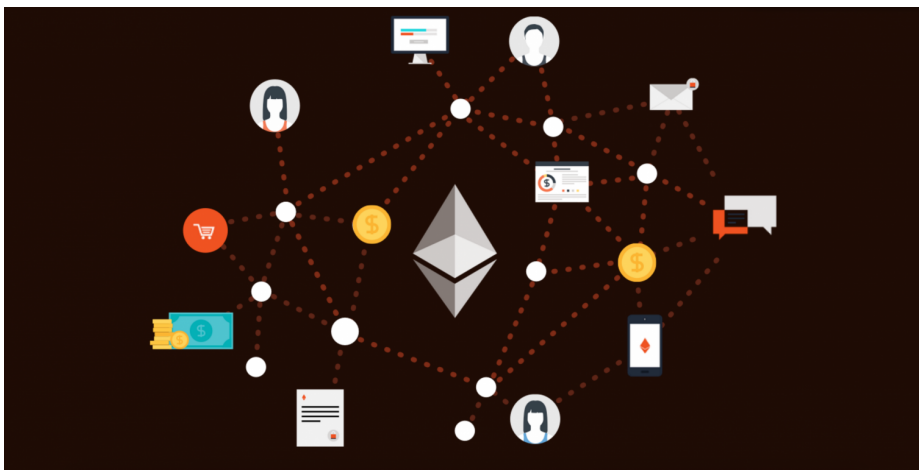
A Basic Overview of the Ethereum Architecture

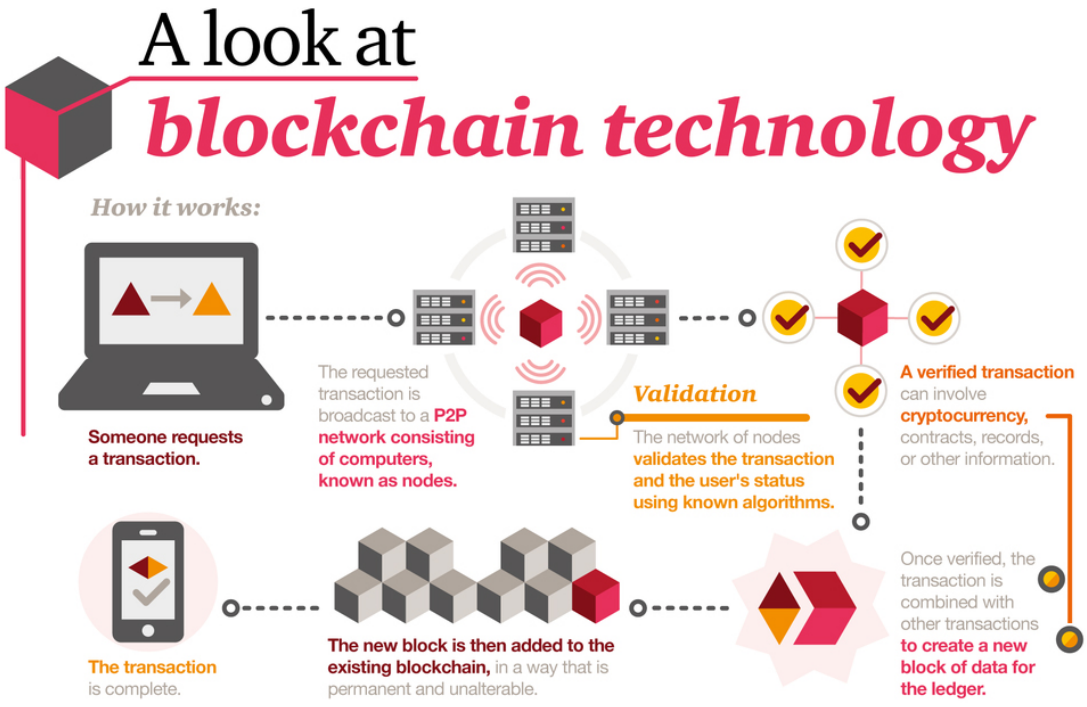
Overview

Ethereum is a public database that keeps a permanent record of digital transactions. Crucially, this database doesn't require any central authority to maintain it or to secure it. It enables a more 'trustful' transactional system, a framework that allows people to make peer-to-peer transactions without needing to trust a third party.



I'm going to cover how Ethereum functions at a technical level, without needing to go into the math formulas involved. Right now, we don't need to understand every single detail, let's just focus on understanding things at a high level.

The Ethereum Blockchain



First of all, a blockchain is a data structure that enables identifying and tracking transactions digitally and sharing this information across a distributed network of computers, creating a distributed trust network.

Blockchains are

Features of Blockchain

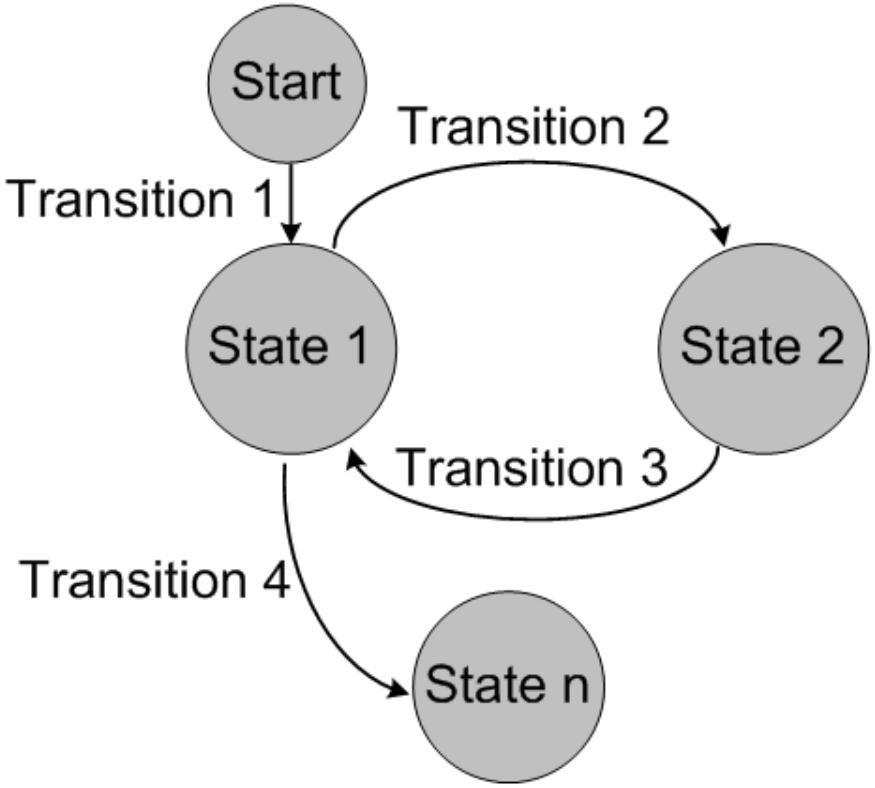
- **Distributed/decentralised**
  - Data are replicated on all the nodes in a distributed P2P network, and each copy of the ledger is identical to others. It can also be decentralised with some lighter nodes not having full data storage with limited connection.
- **Consensus mechanism**
  - All users in the network can come to a pre-determined programmable agreement on the method of validation and can be by consensus.
- **Irreversibility and crypto security**
  - One would need to command at least 51% of the computing power (or nodes or stake) to take control of the bitcoin blockchain. (or others).

- Cryptographically secure

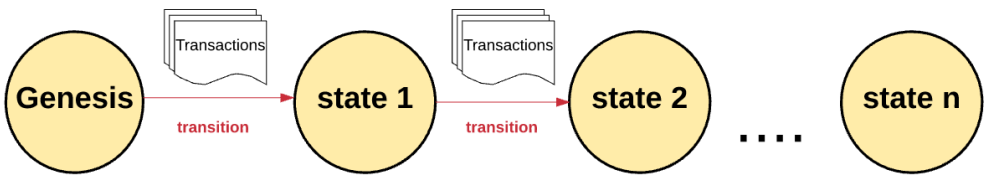
That means that the minting of digital currency is secured by mathematical algorithms that make it very difficult to break. It prevents bad actors from creating fake transactions, erasing transactions, stealing funds, etc.

- Finite-state machines

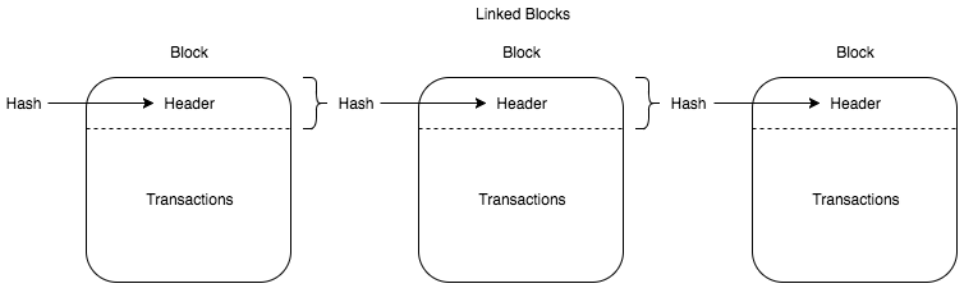
In Computer Science, a state machine is a machine that will analyze a series of inputs and based on those inputs will transition to a new state. Blockchains have one instance responsible for all the transactions being created by the system. There's one global truth that all nodes adhere to, they all share the same state.



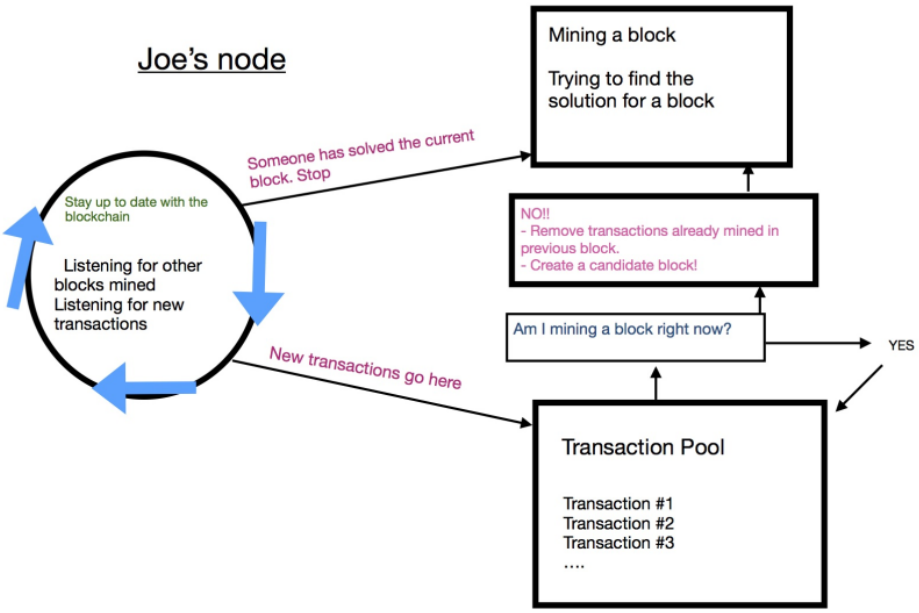
Blockchains all start with a 'genesis state' and Ethereum is no different. The genesis state is akin to a blank canvas, the transactions that will later be executed on the blockchain akin to strokes of a paintbrush. When transactions begin to get executed, the genesis state transitions to its next state. That next state is the current state of the Ethereum Blockchain.



Each transaction in the Ethereum network is grouped into what are called 'blocks'. A single block contains a set of transactions and each block points to the next block. This makes a chain of blocks aka blockchain.



But in order for a state transition to occur, the transactions in a block must be valid. Validity is a crucial concept when it comes to blockchains. The process of validating blocks is called 'mining'. Mining is when a specific group of nodes (miners) uses their computing power to create a block of valid transactions.



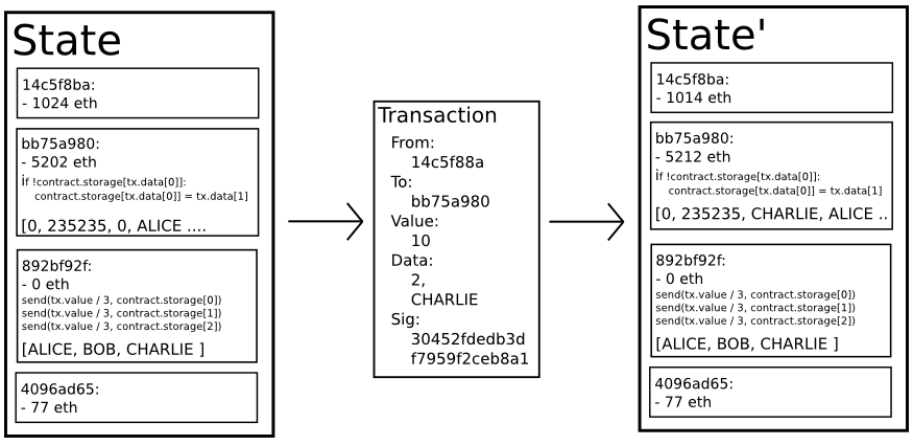
Anyone can be a miner in the network, that means they use their computing power to validate blocks. There is a global community of miners all trying to create and validate blocks simultaneously. These miners provide mathematical proofs when submitting blocks to the blockchain. The proof is the source of truth when it comes to blockchains.

Miners must prove their block is valid faster than the other competing miners for it to be added to the main blockchain. This processes of validating each block by having a miner provide a mathematical proof are called the 'proof of work' algorithm.

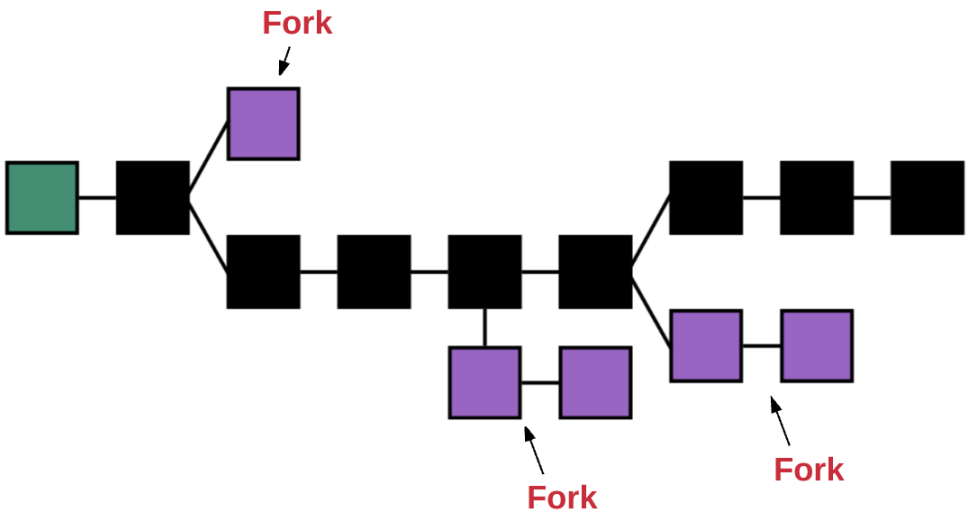
Miners who validate new blocks get rewarded with an intrinsic digital token called 'Ether'. Every single time a miner proves a block as valid, the network generates new Ethers and awards the miner.

	ethereum	ethereum tokens
concept	smart contracts platform	digital assets on top of ethereum
market cap (as of may 2017)	~$17 billion	~$1.5 billion
native currency	ether	augur (rep), golem (gnt), aragon (ant), & many more
founder	vitalik buterin and team	varies by project
release method	presale raised $18M in bitcoin	typically through crowd sales

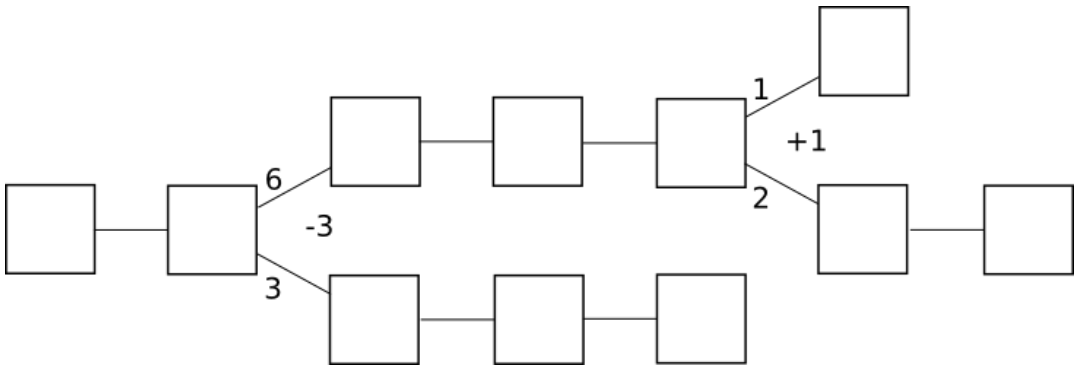
The question arises though, how does the network guarantee that all nodes share the same, single chain of blocks? As in, how can the network prevent a subset of miners from creating their own chain of blocks?



The network needs to have a single, shared state. If there were multiple states, it would cause chaos in the system because it would be nearly impossible for nodes to agree on which state is the correct one. If the blockchain diverged into multiple chains, a person could have different account balances on each one, so it'd be impossible to determine which chain was the most 'valid' chain.



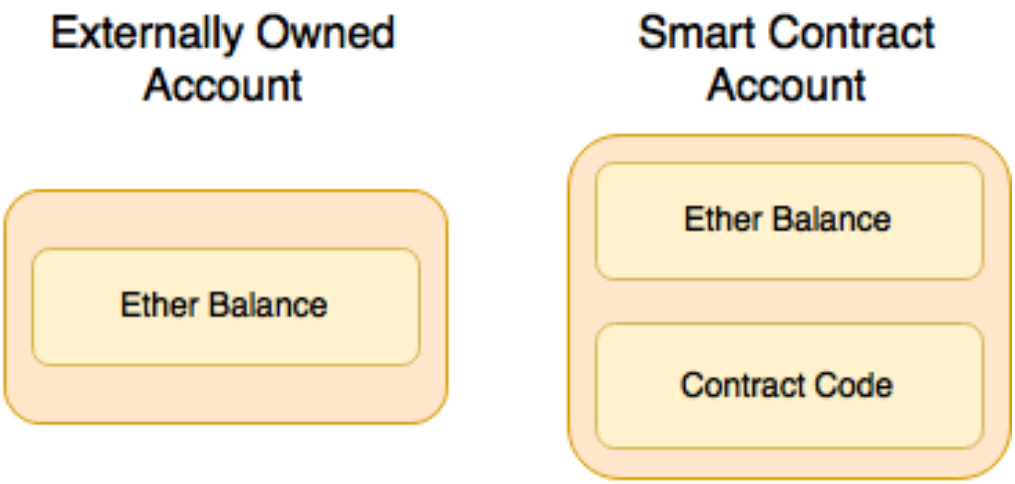
In scenarios where multiple paths are generated by miners, a 'fork' happens. Forks are to be avoided since they are disruptive to the system and force nodes to choose which chain they believe to be the most valid.



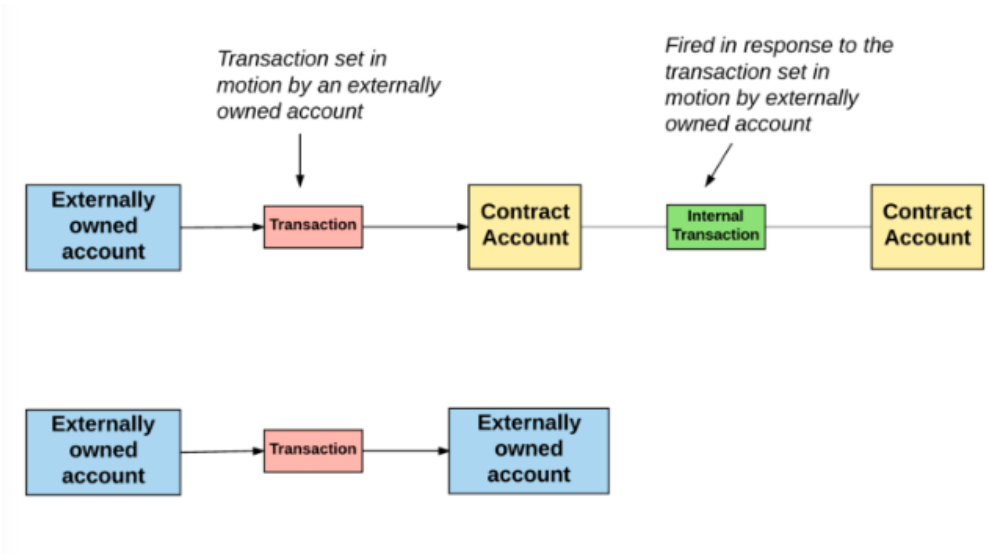
Ethereum's mechanism to choose the most valid chain is called the "GHOST protocol". GHOST stands for "Greedy Heaviest Observed Subtree". Essentially, it picks the path that has had the most computation done on it. The protocol uses the block number of the most recent block, this represents the total number of blocks in the current path. The higher the block number, the longer the path and as such the larger the mining effort that had to have gone into arriving at the most recent block. This allows the network to agree on the correct version of the current state.

Let's now dive into some of the components of the Ethereum Blockchain

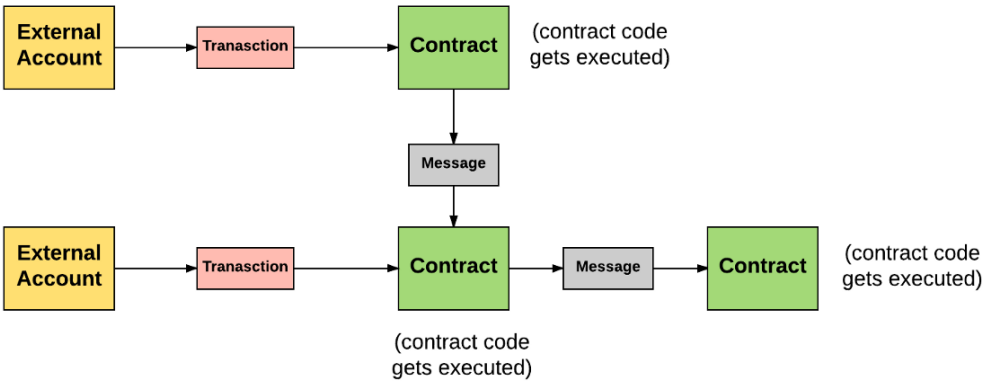
Accounts



In Ethereum, there are lots of small objects that are apart of the shared-state that interact with each other through a message passing framework. Every account has its own state and an address. Ethereum addresses use a 160-bit identifier. And there isn't just one account type; there are two. Externally owned accounts are controlled by private keys and have absolutely no code associated with them. Contract accounts, however, are controlled by their contract code and do have code associated with them.

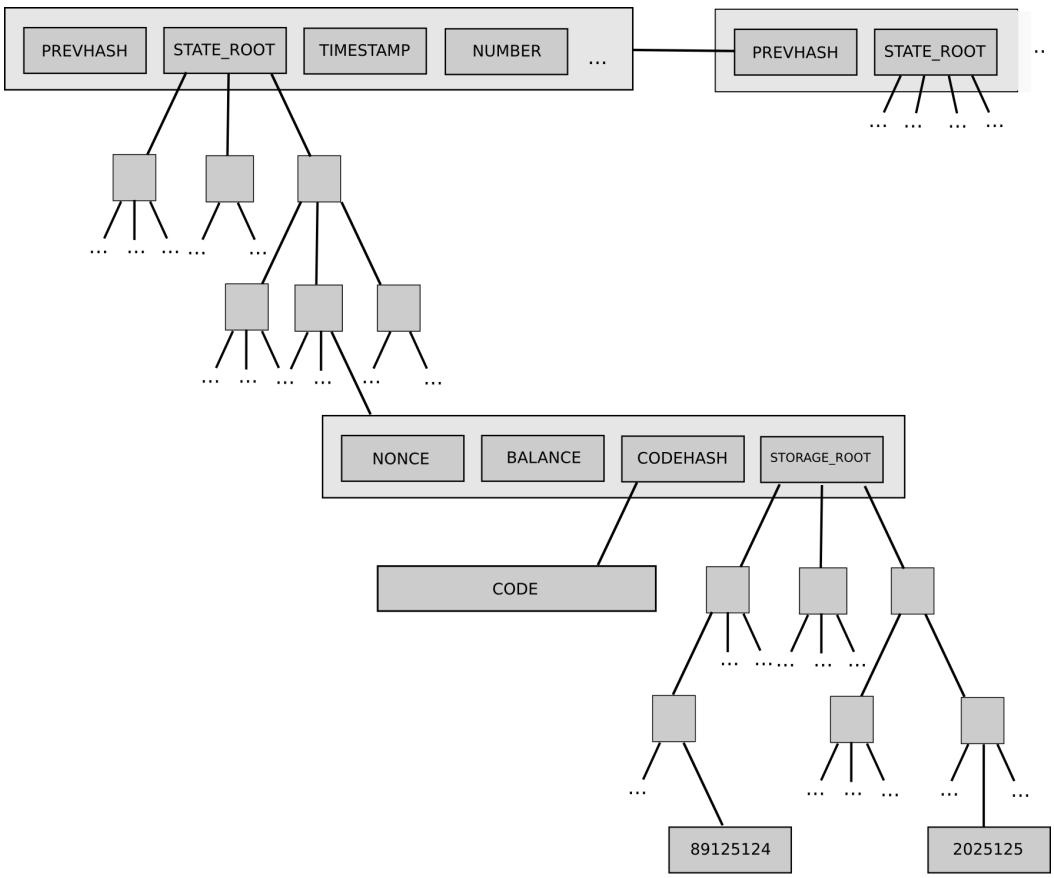


While an externally owned account can send messages to other externally owned accounts or to other contract accounts by signing transactions using its private key, contract accounts can't initiate new transactions on their own. Messages between externally owned accounts is a value transfer, while messages between an externally owned account to a contract account activates the contract accounts code. This allows it to perform a whole range of possible actions (token transfer/minting, writing to internal storage, etc.). Contract accounts only execute transactions in response to other transactions they've received.

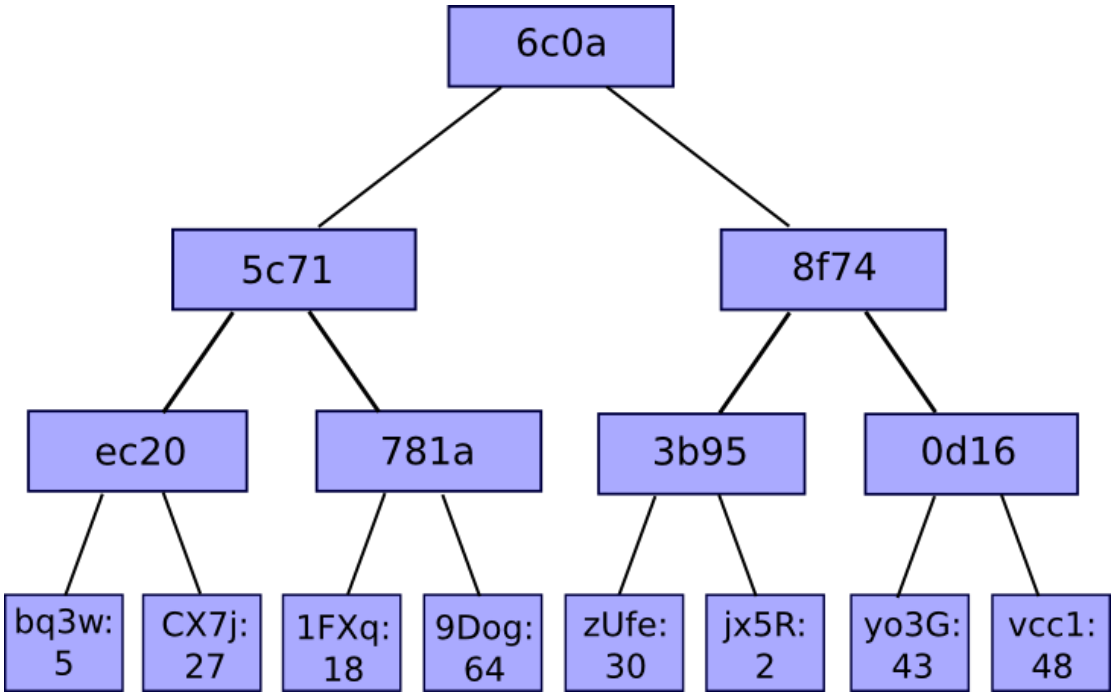


Actions on the Ethereum blockchain happen due to transactions fired from externally controlled accounts.

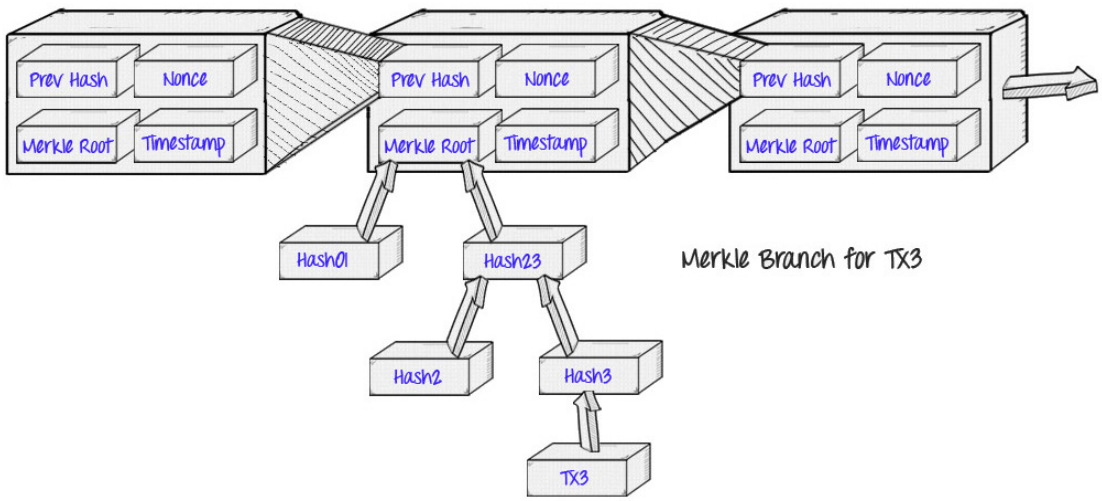
Every account has a state. This account state consists of four components; the nonce, the balance, the storageRoot, and the codeHash. The nonce represents the number of transactions sent from the address of the account. The balance is the amount of Ether owned by the address. The storageRoot is the of the root node of its Merkle tree. And the codeHash is the hash of the Ethereum Virtual Machine code of this account.



A Merkle tree is a type of binary tree composed of a set of nodes with lots of leaf nodes at the bottom of the tree that contains data. Intermediate nodes in a Merkle tree consist of nodes that have a hash of its two child nodes, and the root node is made up of the hash of its two child nodes, representing the top of the tree.

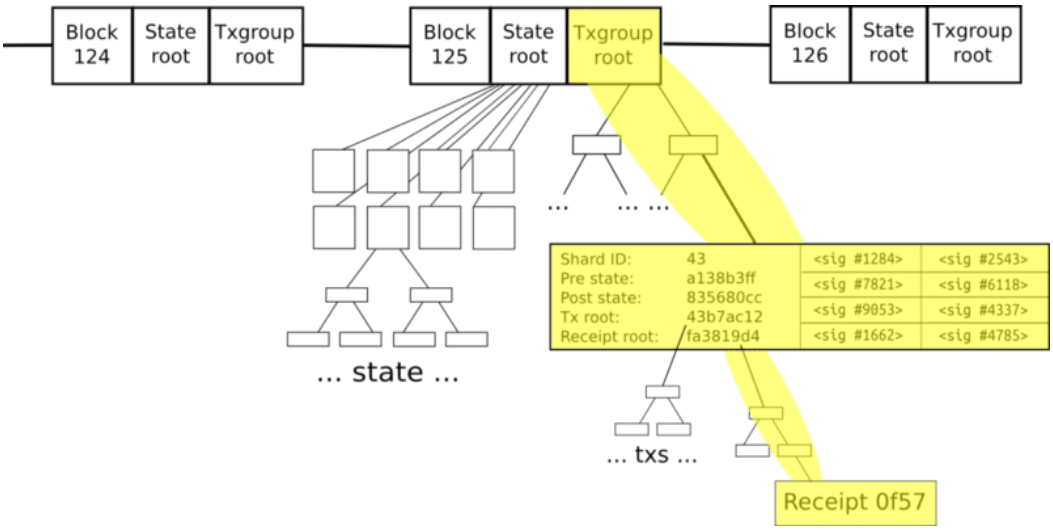


Data at the bottom of the Merkle tree is generated by splitting it into chunks, splitting chunks into buckets, and repeating the process using bucket hashes until there is only a single hash remaining (the root hash).



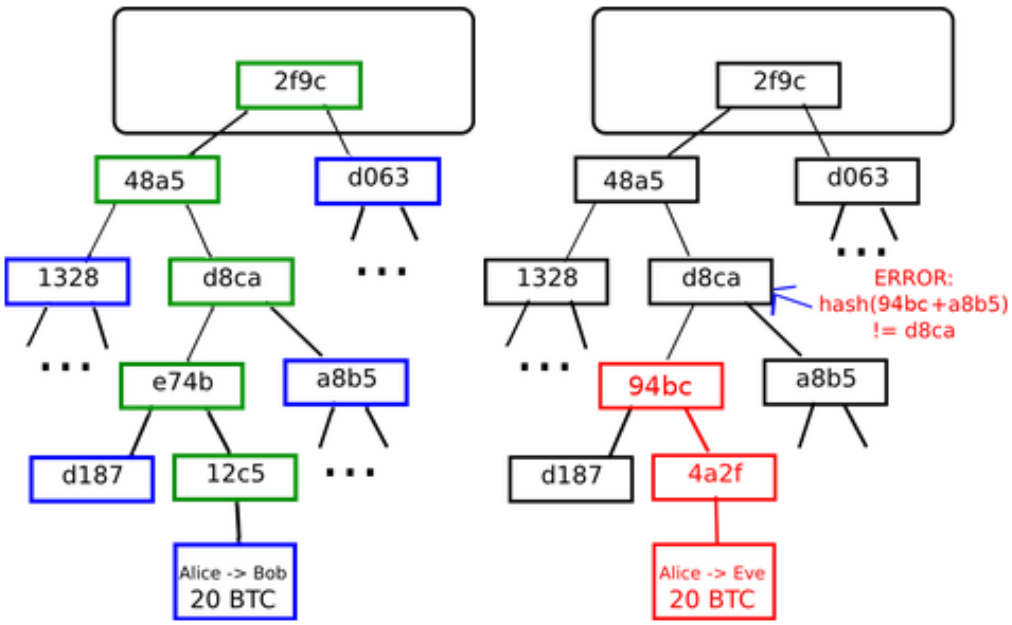
Each node of the tree has a key with an associated value; the key tells us which child node to follow to get to the corresponding value, which is stored in the leaf nodes. In the case of ethereum, the key/value mapping for the state tree is between the addresses and their accounts.

Every blocks header stores the hash of the root node of what are essentially three different Merkle tree structures for the state, transactions, and receipts.



Merkle trees are supremely useful because its a very efficient way to store all this information, especially for light nodes. Ethereum uses light nodes and full nodes; full nodes must download the full blockchain, executing all the transactions contained in it. Light nodes must download only the chain of headers, without needing to execute any transactions or retrieving any associated state. This allows them to easily generate and receive verifiable answers about balances, events, etc.

Merkle trees serve to help secure the blockchain because hashes in the Merkle tree propagate upward. If a bad actor tries to swap a fraudulent transaction into the bottom of the Merkle tree, it will cause a chain reaction in all the hashes in all the nodes above it.



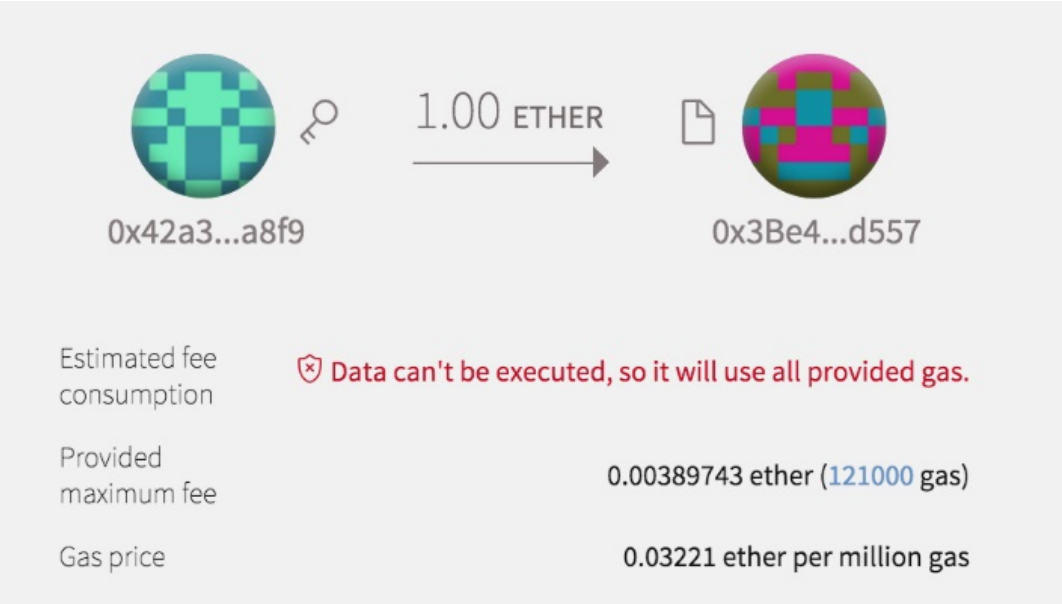
A node that wants to verify some data can use whats called a 'Merkle proof". This contains some data to be verified, the root hash of the tree, and the tree branch (partner hashes going up along the path from the data chunk to the root hash). People who read the proof can verify that branch hashing is consistent throughout the tree.

Fees

Every single computation/transaction on the Ethereum blockchain requires a fee. That fee is paid in whats called 'gas'. Gas is the unit Ethereum uses to measure computation fees. Gas price is an amount of Ether a node is willing to spend on every gas unit, measured in 'gwei' (since 'wei' is the smallest unit of Ether).

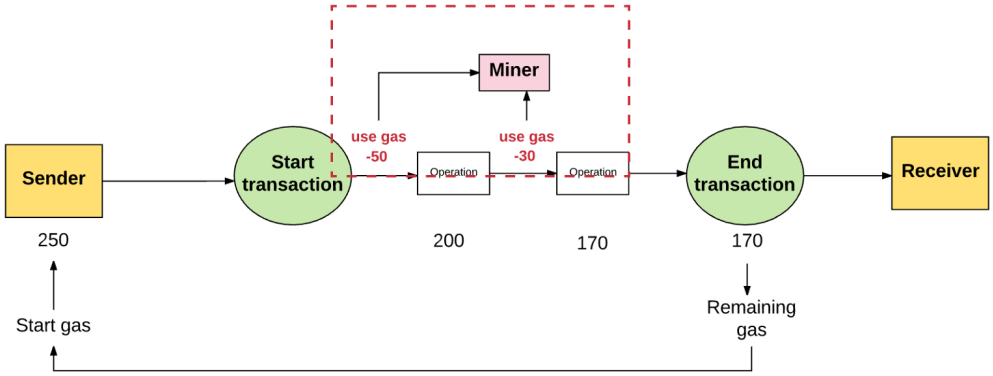
Unit	Wei
Wei	1
Kwei / ada / femtotether	1,000
Mwei / babbage / picoether	1,000,000
Gwei / shannon / nanoether / nano	1,000,000,000
Szabo / microether / micro	1,000,000,000,000
Finney / milliether / milli	1,000,000,000,000,000
Ether	1,000,000,000,000,000,000

For each transaction, a sender sets a gas limit and a gas price. The gas limit represents the maximum gas the sender is willing to pay. If the sender doesn't provide the necessary gas to execute a transaction, the transaction is considered invalid. And since the Ethereum network had to expend computational effort to run the calculations before running out of gas, none of the gas gets refunded to the sender. The money spent on gas by the sender is sent to some miners address since miners are expanding the computational effort to validate transactions. The gas fee acts as a reward for the miners. Importantly, gas is used to pay for storage usage as well.



Fees help prevent users from overtaxing the Ethereum network. Its very computationally expensive to run computational steps on the Ethereum Virtual Machine, so smart contracts should be used for simple tasks like verifying ownership instead of more complex tasks like machine learning or file storage. Fees also help protect the network from malicious attacks. Ethereum has its own Turing complete programming language called Solidity for creating smart contracts. Turing complete means it can simulate any computer algorithm. It allows

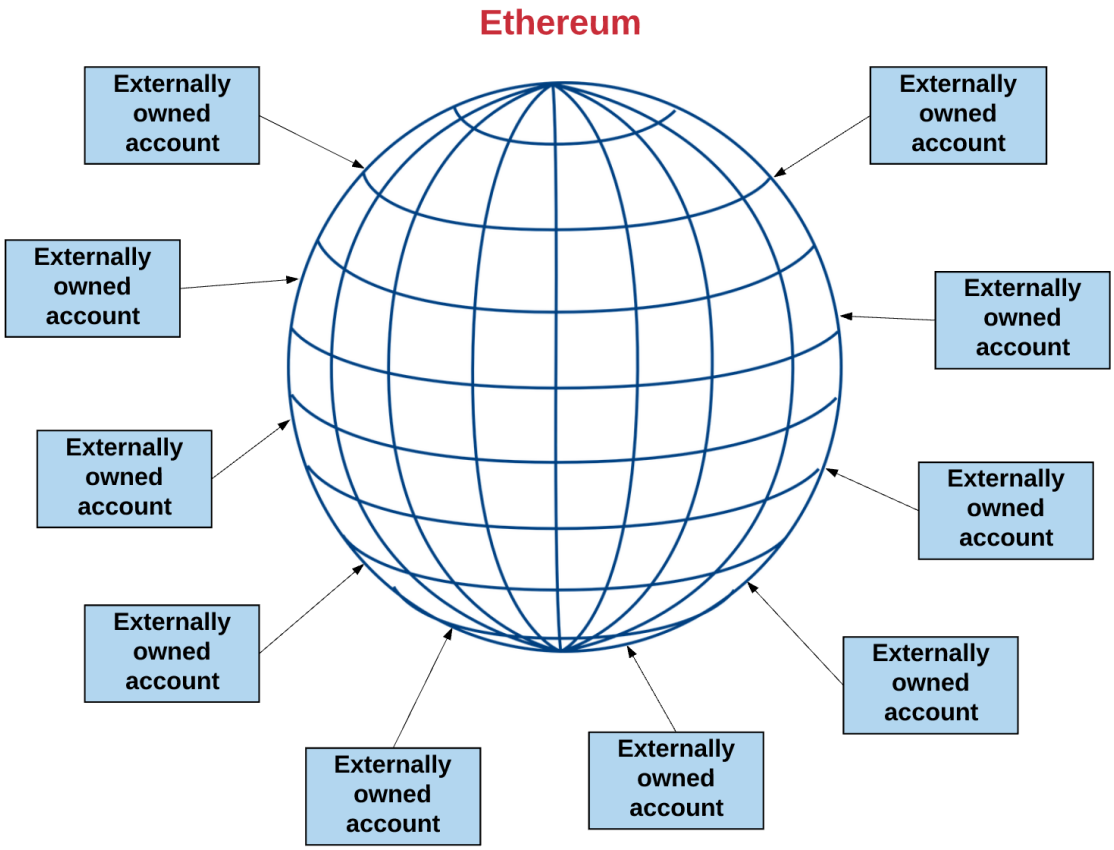
for-loops, so a bad actor could disrupt the network by executing an infinite loop within a transaction, but thanks to fees this becomes infeasible.



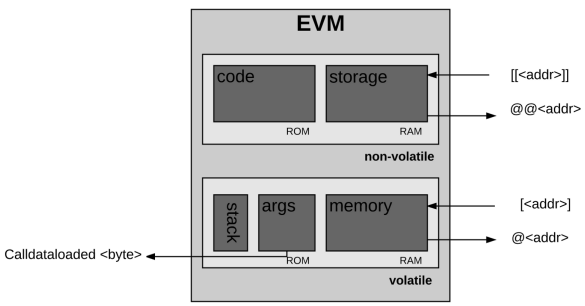
Transactions

Sending a transaction to a contract causes its code to execute. Contracts can store data, send transactions and interact with other contracts.

There are two types of transactions in Ethereum, the message call and the contract creation (creates new smart contracts). Both are initiated by externally owned accounts and submitted to the blockchain. They are what bridge the external world to the internal state of Ethereum. Contracts can talk to other contracts via messages. Messages are like internal transactions. These messages are generated by contracts, and when one contract sends a message to another, the code that exists on the recipient contract account is executed.



Ethereum Virtual Machine



The EVM is a complete virtual machine, and its only limitation is that its bound by gas. Meaning the total amount of computation it can do is limited by the amount of gas provided. Its a stack-based architecture (last-in, first-out). It has temporary memory and long-term storage. It even has its own language! (called EVM bytecode). When we write smart contracts, it's in a higher level language like Solidity, but this compiles down to EVM bytecode.

There's a lot more we can get into, but hopefully, this post helped you understand some of the important modules that make up the Ethereum network. Let's keep going!