Lecture 02:
Ethereum Mechanics

Akash Khosla

HOMESTEAD

ethereum.org

BLOCKCHAIN

AT BERKELEY

LECTURE OUTLINE

1. ETHEREUM PREVIEW
2. BITCOIN LIMITATIONS
3. ETHEREUM DEEP DIVE
4. SMART CONTRACTS

1 ETHEREUM PREVIEW

ethereum

~~buzzwords~~

overview

BLOCKCHAIN

TRUSTLESS

DECENTRALIZED APPS

SMART CONTRACTS

DAOs

BITCOIN 2.0

AUTHOR: PHILIP HAYES

WHAT IS ETHEREUM?

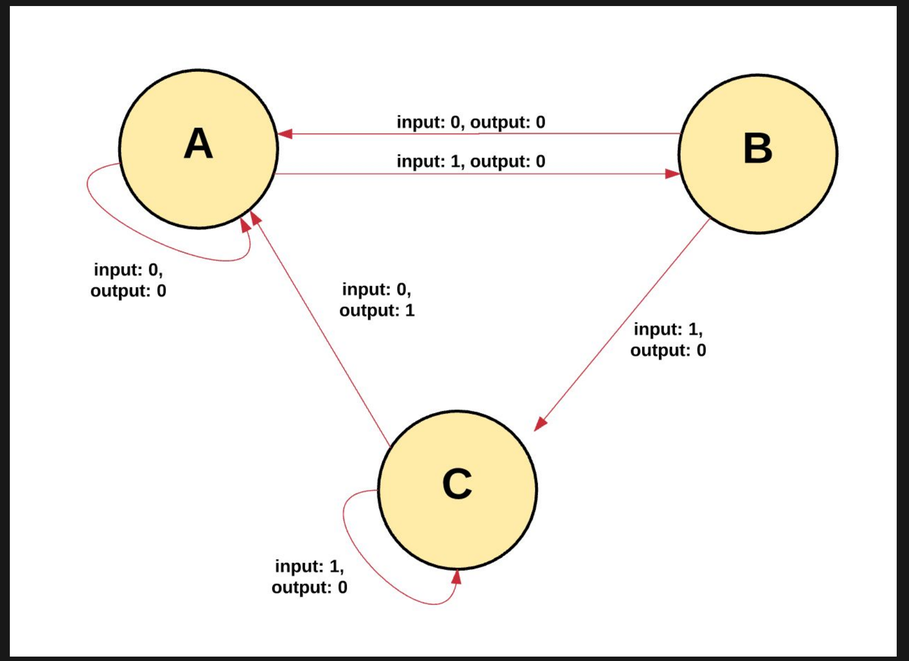
VITALIK'S CHILD

- Ethereum is a **decentralized** platform designed to run **smart contracts**.
  - Resistant to censorship
  - **Distributed server**
- Ethereum maintains a native asset called **ether** that is the basis of value in the Ethereum ecosystem, allows for aligning incentives
- **Account based**, not **UTXO based**
  - Remember: a UTXO is not a transaction, but simple a mechanism to manipulate and maintain state for transactions
  - **Ethereum has transactions which are used to manipulate the global account state**

WHAT IS THE BLOCKCHAIN FOR ETHEREUM

PUTIN'S FAVORITE CRYPTOCURRENCY

- For Ethereum - a blockchain is a "cryptographically secure transactional singleton machine with shared-state."
  - **Cryptographically secure** - Can't create fake transactions, erase transactions because of complex mathematical algorithms.
  - **Transaction singleton machine** - single instance of the machine for all the transactions being created in the system ("global truth")
  - **Shared-state** - state stored on this machine is shared and open to everyone

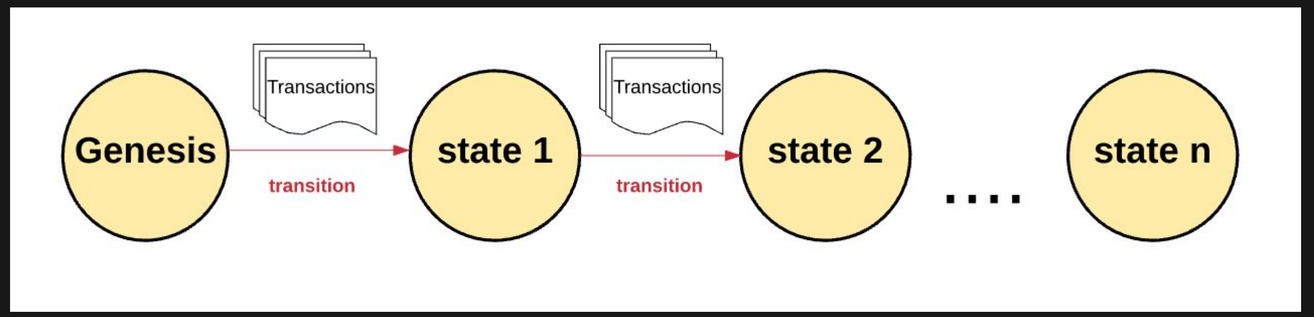


Source

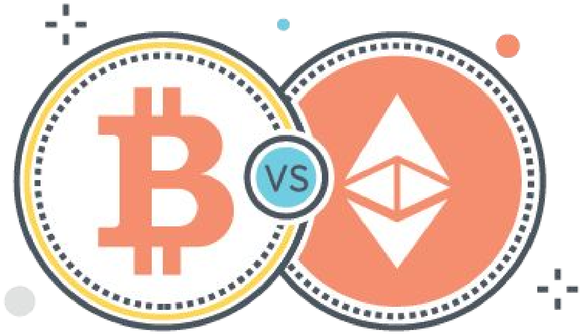
WHAT IS A STATE MACHINE

PUTIN'S FAVORITE CRYPTOCURRENCY

- Ethereum blockchain is essentially a **transaction based state machine**
  - A **state machine** refers to something that will read a series of inputs and based on those inputs, will transition to new state
  - This state machine begins with **genesis state** - state at which no transactions have occurred on the network
  - Once transactions are executed, the genesis state transitions to another state, where the **final state** is the current state of the Ethereum network.



Source



2 BITCOIN LIMITATIONS

HOW BITCOIN WORKS

STATE TRANSITION PERSPECTIVE

- Bitcoin operates on running client software that agrees on state transitions
- **State**: the ownership status of all existing bitcoins
- **State Transition Function**: takes a state and transaction and outputs a new result
- APPLY(S, TX) -> S'

```
/* Bitcoin blockchain update algorithm */
def APPLY(state, TX):
  For each input in TX: # Input includes UTXO and signature
    if input.UTXO not in state: return ERROR # referenced UTXO not in state
    # provided signature doesn't match the owner of the UTXO
    if input.signature != lookup_UTXO(input).signature: return ERROR

  if sum(input UTXOs) < sum(output UTXOs): return ERROR
  S.update # Remove input UTXOs and add all output UTXOs
```

HOW BTC BLOCK VALIDATION WORKS

STATE TRANSITION PERSPECTIVE

- The previous algorithm works perfect for a centralized currency system
- To make a decentralized currency system, need **consensus** and **state transition** systems
- Produce a package of transactions every 10 minutes called **blocks**.

```
def VALIDATE_BLOCK(block):
  is_valid(block.prev)
  assert(block.timestamp > block.prev.timestamp)
  # less than 2 hours (120 min) into the future
  assert(block.timestamp < block.prev.timestamp + 120)
  check_pow(block) # valid Proof of Work, used for consensus
  TX_list = block.prev.transactions
  for i in range(len(TX_list)):
    # S[0] is the state, block.prev.S[0] end of the prev block state
    block.S[i] = APPLY(block.prev.S[i], TX_list[i]) # return False on error
  return True
```

ALTERNATIVE BLOCKCHAIN APPLICATIONS

BITCOIN PERSPECTIVE

- **Namecoin** - Decentralized Name Registration
  - Tor, Bitcoin and other protocols often use a pseudorandom hash like `1LW79wp5ZBqaHW1jL5TCiBCrhQYtHagUWy` to identify them
- Why not have human readable names on chain
- Oldest implementation of name reg to use first-to-file paradigm
  - First registration succeeds and the second fails
  - Bitcoin consensus a perfect platform
  - Not easy to write with Bitcoin script

ALTERNATIVE BLOCKCHAIN APPLICATIONS

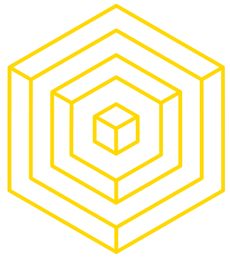
BITCOIN PERSPECTIVE

- **Colored coins** - Allow people to create their own currencies
  - Issue a new currency by publicly assigning color to a specific Bitcoin UTXO
  - Recursively define the color of UTXOs to be the same as the color of the inputs that the transaction creating them spent
  - Therefore can maintain wallet of colored UTXOs and send them around like regular bitcoins
- **Metacoins -** Protocols that live on top of Bitcoin
  - Provide an alternative state transition function
  - Let bitcoin handle the mining and networking infrastructure
  - Financial contracts, name registration, decentralized exchange

LIMITATIONS OF BITCOIN SCRIPT

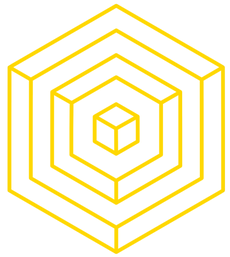
BITCOIN PERSPECTIVE

- **Lack of Turing Completeness**
  - Bitcoin script supports a fair amount of computation but is missing loops
  - Why might that be a good thing?
- **Value-blindness**
  - No way for a UTXO script to provide fine grained control over the amount withdrawn
  - UTXOs are all or nothing, and it’s expensive to have varying denominations
- **Lack of State**
  - Only unspent and spent UTXOs, binary state, no “efficient” withdrawal limits
- **Blockchain-blindness**
  - UTXOs are blind to blockchain data like the nonce, timestamp and prev block hash
  - Lost out on potential source of randomness



3 ETHEREUM DEEP DIVE

Ethereum	Bitcoin
Smart Contract Platform ○ Complex and feature-rich ○ **Turing complete scripting language**	Decentralized Asset ○ Simple and robust ○ Simple stack-based language; not Turing complete
Account-based for transaction	**UTXO-based for transactions**
Turing complete scripting language that enables smart contracts. Ether asset is not the primary goal. Uses proof of work, plans to do proof of stake.	**Uses Bitcoin Script, a very simple stack based language.** Currency itself the main goal. Uses proof of work.
~12 sec block time - using ethash mining	~10 min block time - using sha256 mining



3.1 ACCOUNTS

UTXO VS. ACCOUNTS

THE ULTIMATE COMPARISON

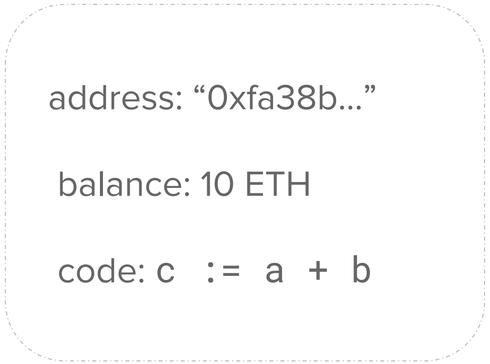
- A Bitcoin user's available **balance** is the **sum of unspent transaction outputs** for which they own the private keys to the output addresses
- Instead Ethereum uses a different concept, called **Accounts**, which already keeps track of **balance**

Bitcoin:

Bob owns private keys to set of UTXOs

Ethereum:

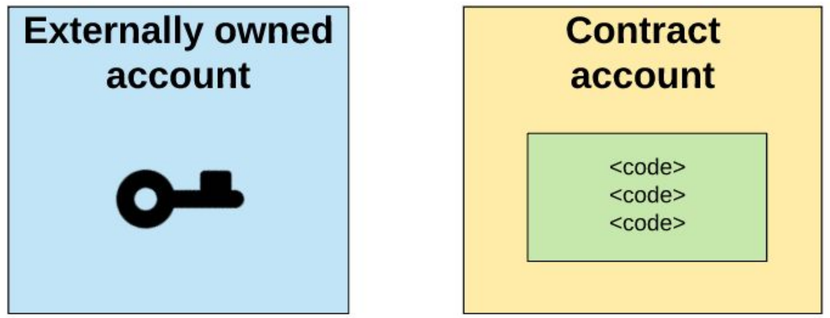
Evan owns private keys to an account



ACCOUNTS

HOW ETHEREUM MANAGES IDENTITY

- The global shared state of ethereum is comprised of many small objects **(Accounts)** that are able to interact with one another through a message-passing framework
- An **Account** has state and 20 byte/160 bit byte identifier, i.e:
  - `0x91fff4cbd6159a527ca4dcce2e39374310 86c662`
- Two Types of Accounts:
  - **Externally owned**, which are controlled private keys and have no code associated with them.
  - **Contract accounts**, which are controlled by their code and have code associated with them.

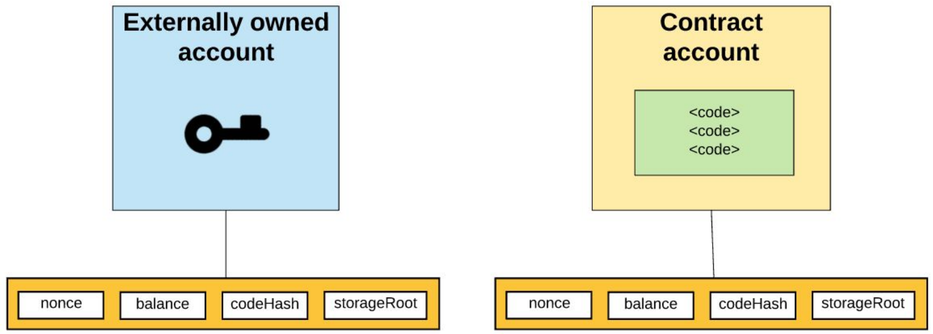


Source

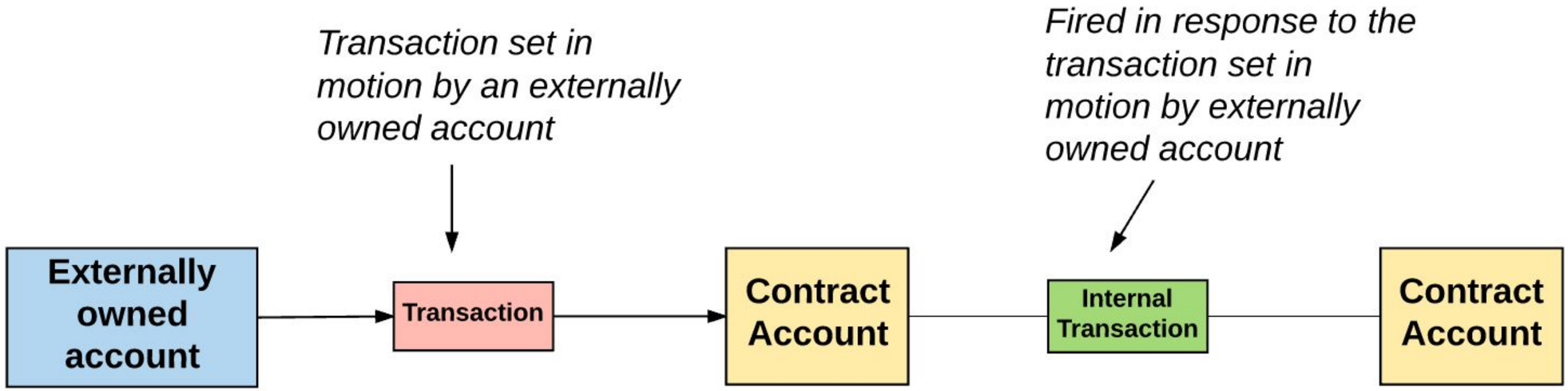
WHAT IS STORED IN ACCOUNT STATE

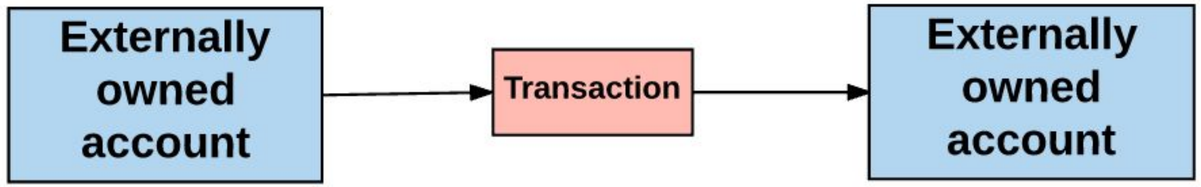
FOUR COMPONENTS

- **nonce**: number of transactions sent from external account, number of contracts created if contract account
- **balance**: The number of Wei owned by this address. 1e18 Wei per 1 Ether.
- **storageRoot**: A hash of the root node of a Merkle Patricia tree. This tree encodes the hash of the storage contents of this account, and is empty by default.
- **codeHash**: The hash of the EVM (Ethereum Virtual Machine—more on this later) code of contract account account. `Hash("")` for external accounts.



EXTERNALLY OWNED VS CONTRACT ACCOUNTS





Any action that occurs on the Ethereum blockchain is always set in motion by transactions fired from externally controlled accounts.

Source

ALL ACCOUNTS == NETWORK STATE

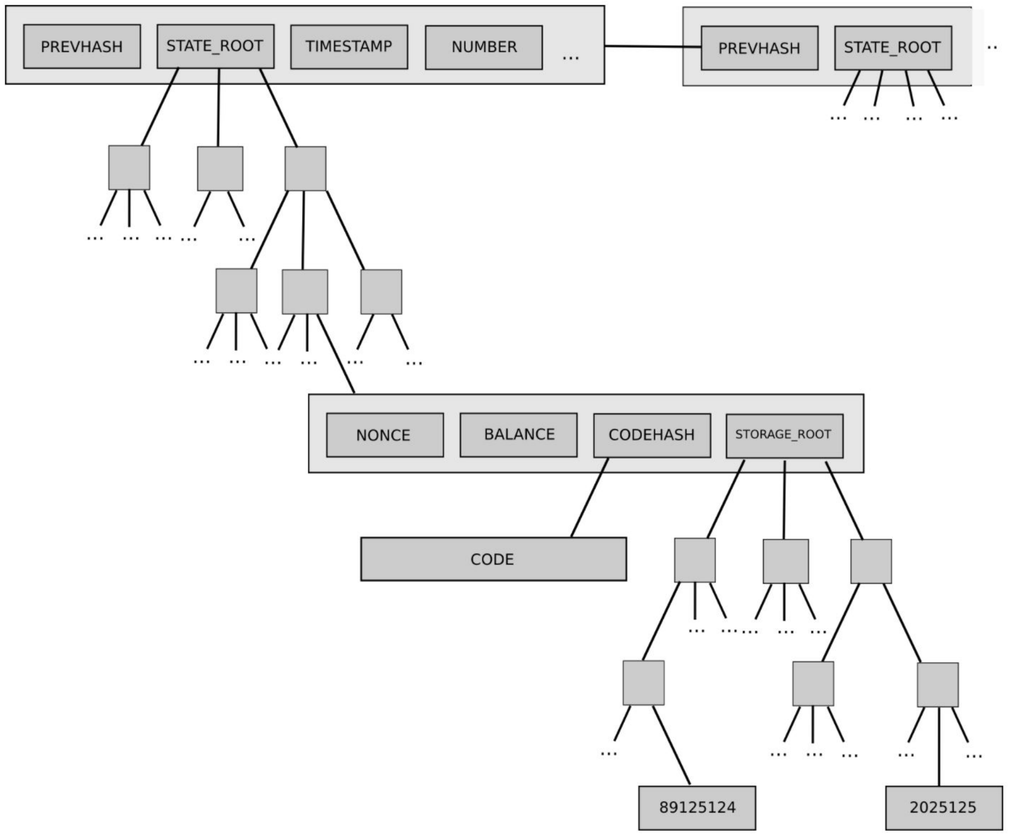
GLOBAL “TRUTH”

- **State of all accounts is the state of the Ethereum network**
  - Entire Ethereum network agrees on the current balance, storage state, contract code, etc. of every single account
- **Ethereum network state is updated with every block**
  - A block takes the previous state and produces a new network state
    - every node has to agree upon new network state
- **Accounts interact** with network, other accounts, other contracts, and contract state **through transactions**

ACCOUNTS RATIONALE

WHY USE ACCOUNTS

- **Space Savings**
  - Nodes only need to update each account's balance instead of storing every UTXO
- **More Intuitive**
  - Smart contracts are more easier to program when transferring between accounts with a balance vs. constantly updating a UTXO set to compute user's available balance.
- **Comparable efficiency due to "Merklization"**
  - Use of merkle patricia tries allows for SPV light clients to work



3.2 FEES

GAS AND PAYMENT

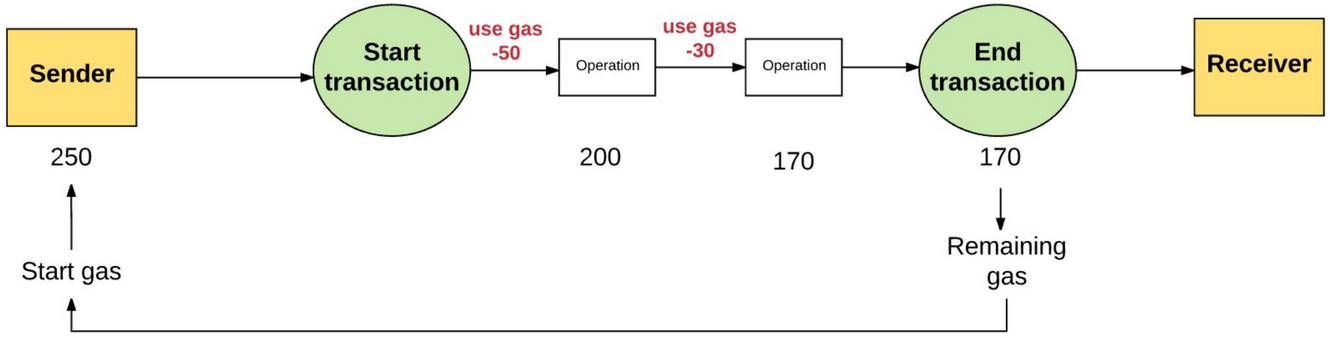
FEES

- Every computation that occurs as a result of a transaction on the Ethereum network incurs a fee called **gas**
- **Gas** is the unit used to measure the fees for a particular computation
- **Gas price** is the amount of Ether you are willing to spend on every unit of gas
    - Measured in “gwei” - 1E18 wei = 1 ETH, 1 gwei = 1,000,000,000 wei
- With every transaction, a sender sets a **gas limit** and a **gas price**
    - **gas price** * **gas limit** = max **amount** of wei **sender is willing to pay** for transaction

GAS AND PAYMENT

FEES

- Example: **Gas Limit:** 50,000, **Gas Price:** 20 gwei
  - Max transaction fee: 50,000 * 20 gwei = 1,000,000,000,000,000,000 = 0.001 Ether
- Since the **gas limit** is the maximum gas the sender is willing to spend money on, if they have enough Ether in their account balance to cover this maximum and the **gas limit** is enough to execute the transaction, transaction will execute
  - `startGas == gasLimit == gasUsed`
- Unused gas is refunded to the sender

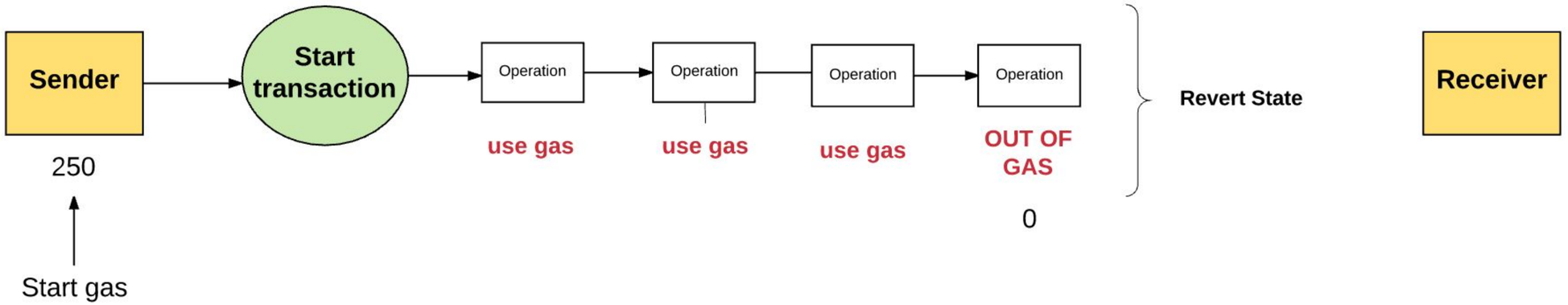


Source

GAS AND PAYMENT

WHAT IF NOT ENOUGH IS SENT?

- Not enough gas to execute the transaction?
  - Transaction runs out of gas and therefore is considered invalid
  - State changes are reversed, failing transaction recorded
  - Since computation was already expended by the network, **none of the gas is refunded**

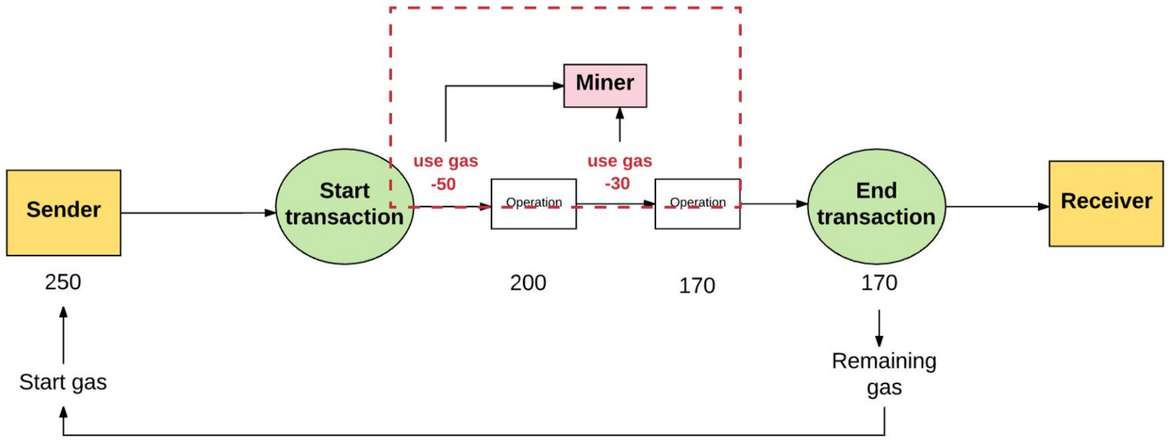


Source

GAS AND PAYMENT

MINER INCENTIVES

- Where does the fee go?
  - **The miner's address**, since they are expanding the effort to run computations and validate transactions - it's the **reward**!
- The higher the gas price the sender is willing to pay, the greater the value the miner derives
  - Therefore more likely to select in block, as miners can choose which transactions to validate and ignore (gas price advertising also possible)



Source

STORAGE FEES

NOT JUST FOR COMPUTATION

- **Gas** is also used to pay for **storage**, proportion to the smallest multiple of **32 bytes** used
- Increased storage however increases the size of the Ethereum state on all nodes
    - **Therefore incentives to keep data small**
- So if a transaction has a step that clears and entry in storage, the fee is waved and a **refund** is given for free storage space

WHY DO WE NEED FEES?

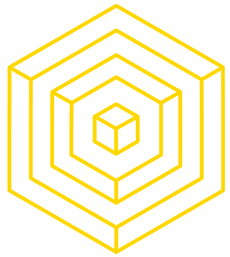
PARALLEL, DISTRIBUTED COMPUTATION

- Every single operation executed by the network is simultaneously affected by every full node
  - Computation steps on the EVM are therefore very expensive
- **Smart contracts are therefore best used for simple tasks**
  - Business logic or verifying signatures rather than machine learning or file storage
  - Redundantly parallel, no asynchronous or performant parallel execution
- **Turing completeness** allows for loops and susceptibility to the **halting problem**
  - **Halting problem**: inability to determine whether or not a program will run indefinitely
  - No fees means DDOS through infinite loops

SIGN IN:

tinyurl.com/developers2

CODE: FEES



3.2 TRANSACTIONS AND MESSAGES

WHAT IS A TRANSACTION

STATE CHANGERS

- Transactions move the state of an account within the global state - one state to the next
- **Formal Definition:** A transaction is a cryptographically signed piece of instruction that is generated by an externally owned account, serialized, and then submitted to a blockchain
- Two types: **Message calls and contract creations**

TRANSACTION COMPONENTS

STATE CHANGERS

- **nonce**: number of transactions sent by sender
- **gasPrice**: amount of Wei sender is willing to pay per unit of gas required to execute the transaction
- **gasLimit**: max amount of gas the sender is willing to pay for executing this transaction, set before any computation is done
- **to**: address of the recipient
- **value**: the amount of Wei to be transferred from the sender to the recipient
- **v, r, s**: used to generate the signature that identifies the sender of the transaction.

  init (only exists for contract-creating transactions): An EVM code fragment that is used to initialize the new contract account
- **data** (optional field that only exists for message calls): the input data (i.e. parameters) of the message call

MESSAGE CALLS

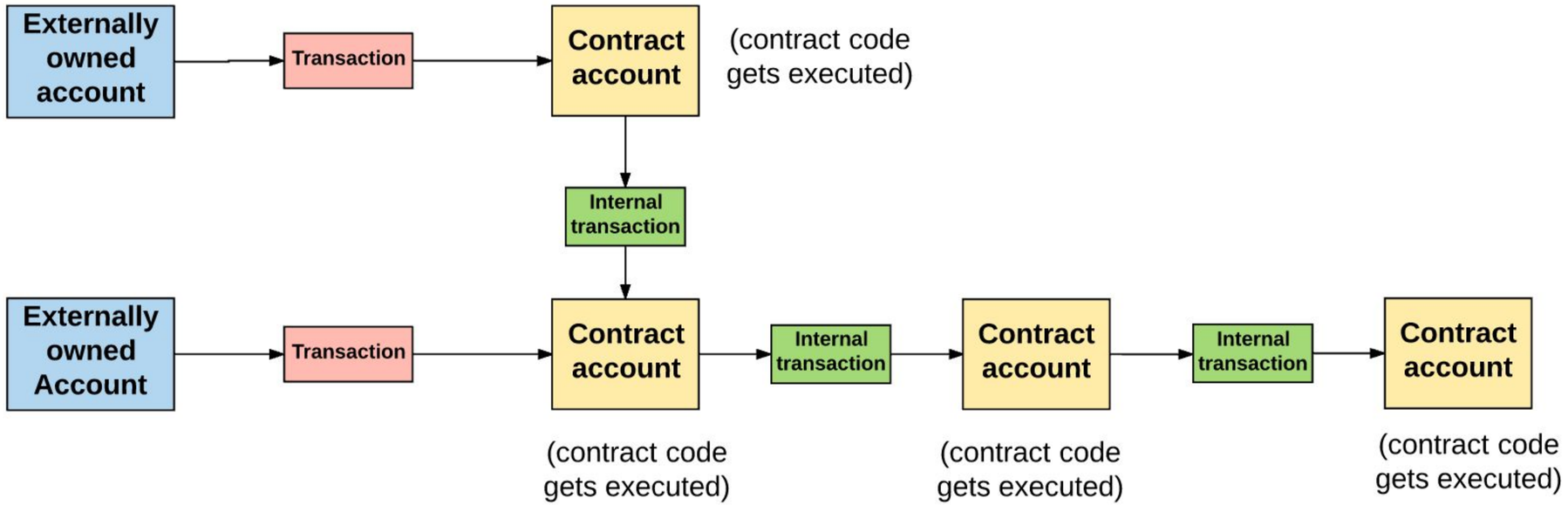
TRANSACTIONS

- Both **message calls** and **contract creating** transactions are always initiated by **externally owned accounts**
  - Transactions bridge the external world to the internal state of Ethereum
- Contracts that exist within the global scope of Ethereum can talk to other contracts using **messages** (internal transactions) to other contracts
- We can think of messages as being similar to transactions, except they are not generated by externally owned accounts, only by contracts
  - Virtual objects within the Ethereum execution environment

NESTED MESSAGE CALLS

TRANSACTIONS

When one contract sends an internal transaction to another contract, the associated code that exists on the recipient contract account is executed.

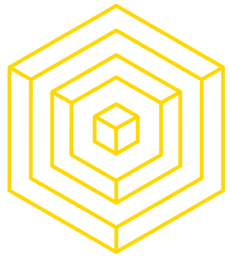


Source

NESTED MESSAGES

INTERNAL TRANSACTIONS

- Messages do not contain a **gasLimit**
- **gasLimit** determined by the external creator of the original transaction
  - Therefore the **gasLimit** that the externally owned account sets must be high enough to carry out the transaction and any other sub executions that occur as a result of that transaction
  - i.e. Contract-to-Contract messages
- **What if we run out of gas within a parent execution?**
  - Current and subsequent message executions will revert, however the parent execution need not revert



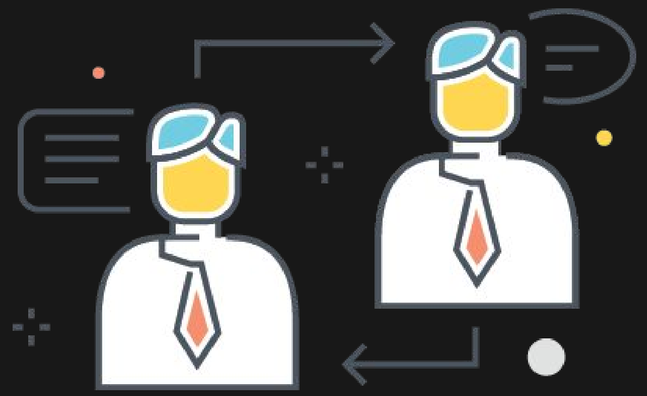
4 SMART CONTRACTS

WHAT IS A SMART CONTRACT?

AUTONOMOUS AGENTS

In Ethereum, smart contracts are executed by the network itself!

- **Network consensus** removes need for Trusted Third Party
- Violation of contracts requires **subverting the entire network**
- Allows for secure Peer-to-Peer agreements that live on the blockchain forever



WHAT IS A SMART CONTRACT?

CODE YOU CAN POKE

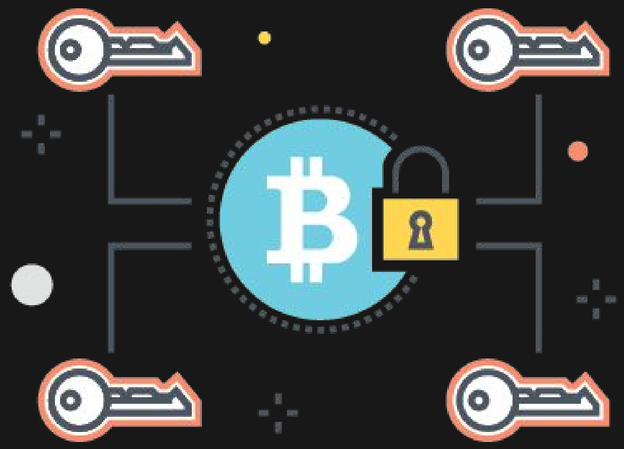
Contracts in Ethereum are like autonomous agents that live inside of Ethereum network

- React to external world when "poked" by transactions (which call functions)
- Have direct control over:
  - internal ether balance
  - internal contract state
  - permanent storage

SMART CONTRACTS

SERVERLESS EXECUTION PROTOCOL

- Ethereum Contracts generally serve four purposes:
  - **Store and maintain data**
    - representing something useful to users or other contracts
    - ex: a token currency or organization’s membership.
  - **Manage contract or relationship between untrusting users**
    - ex: financial contracts, escrow, insurance.
  - **Provide functions to other contracts**, serving as a software library.
  - **Complex Authentication**
    - ex: M-of-N multisignature access



APPLICATIONS

WHAT'S COOL?

- Token Systems
- Financial derivatives and stable value currencies
- Identity and Reputation Systems
- Decentralized File Storage
- Decentralized Autonomous Organizations
- Savings Wallet
- Decentralized Data Feed
- Multisignature Escrow
- Peer-to-Peer Gambling
- Prediction Markets

TO BE DISCUSSED LATER...

PROTOCOL SEGMENT

- Ethereum Block Structure
  - Logs
- Omners and Uncles in Ethereum
- GHOST/SPECTRE Protocols
- Merkle Patricia Tries
- Block Difficulty
- Ethereum Execution Model via the EVM
- Formal State Transitions to Finalize a Block
- Ethereum Proof of Work