GAS AND GAS SPONSORSHIP IN CONFLUX

TOPICS WE WILL COVER TODAY

- transaction fee in Bitcoin
- transaction fee in Ethereum
- transaction fee in Conflux
- transaction sponsorship
- blockchain applications and tools

TOPICS WE WILL COVER TODAY

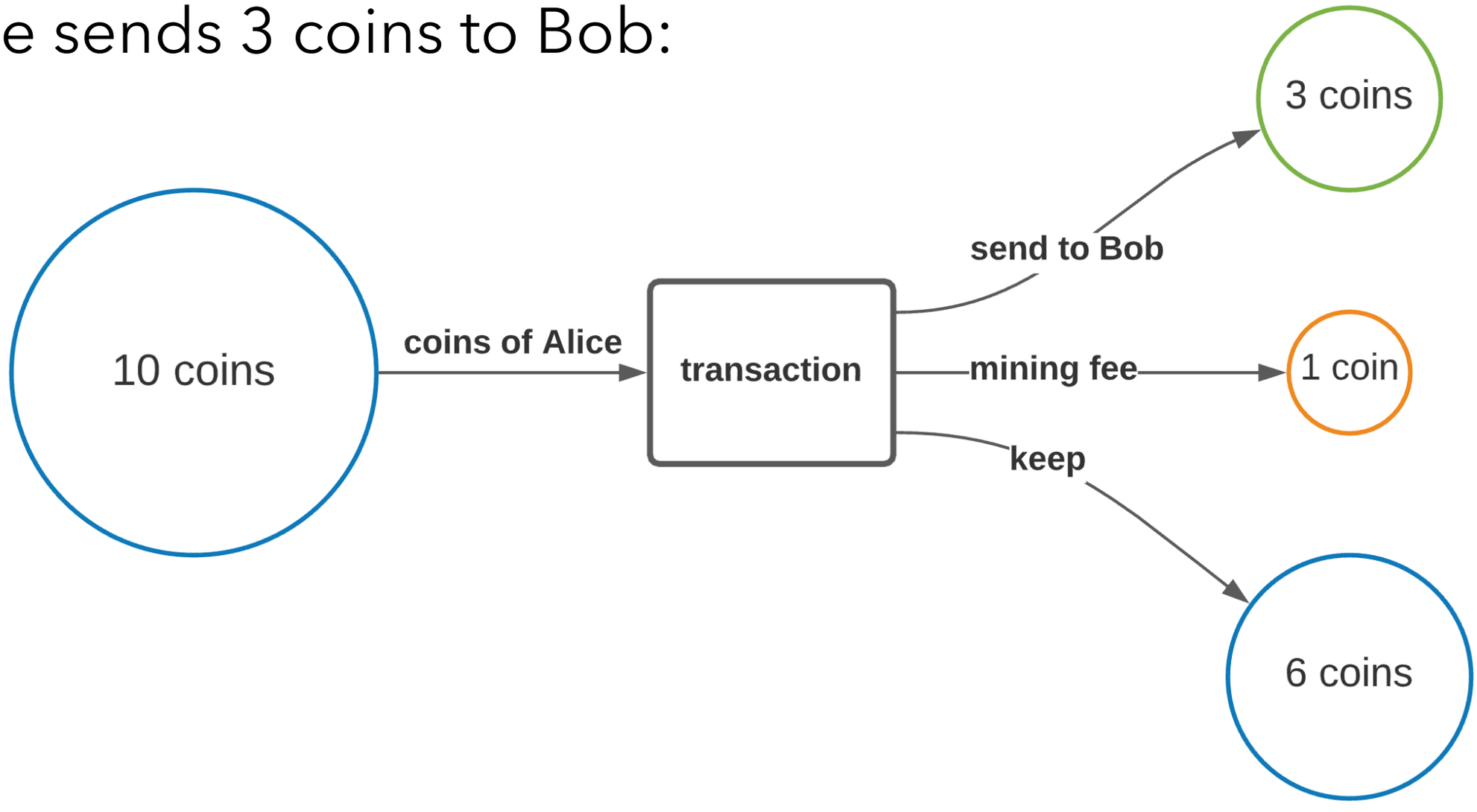
- **transaction fee in Bitcoin**
- transaction fee in Ethereum
- transaction fee in Conflux
- transaction sponsorship
- blockchain applications and tools

TRANSACTION FEE IN BITCOIN

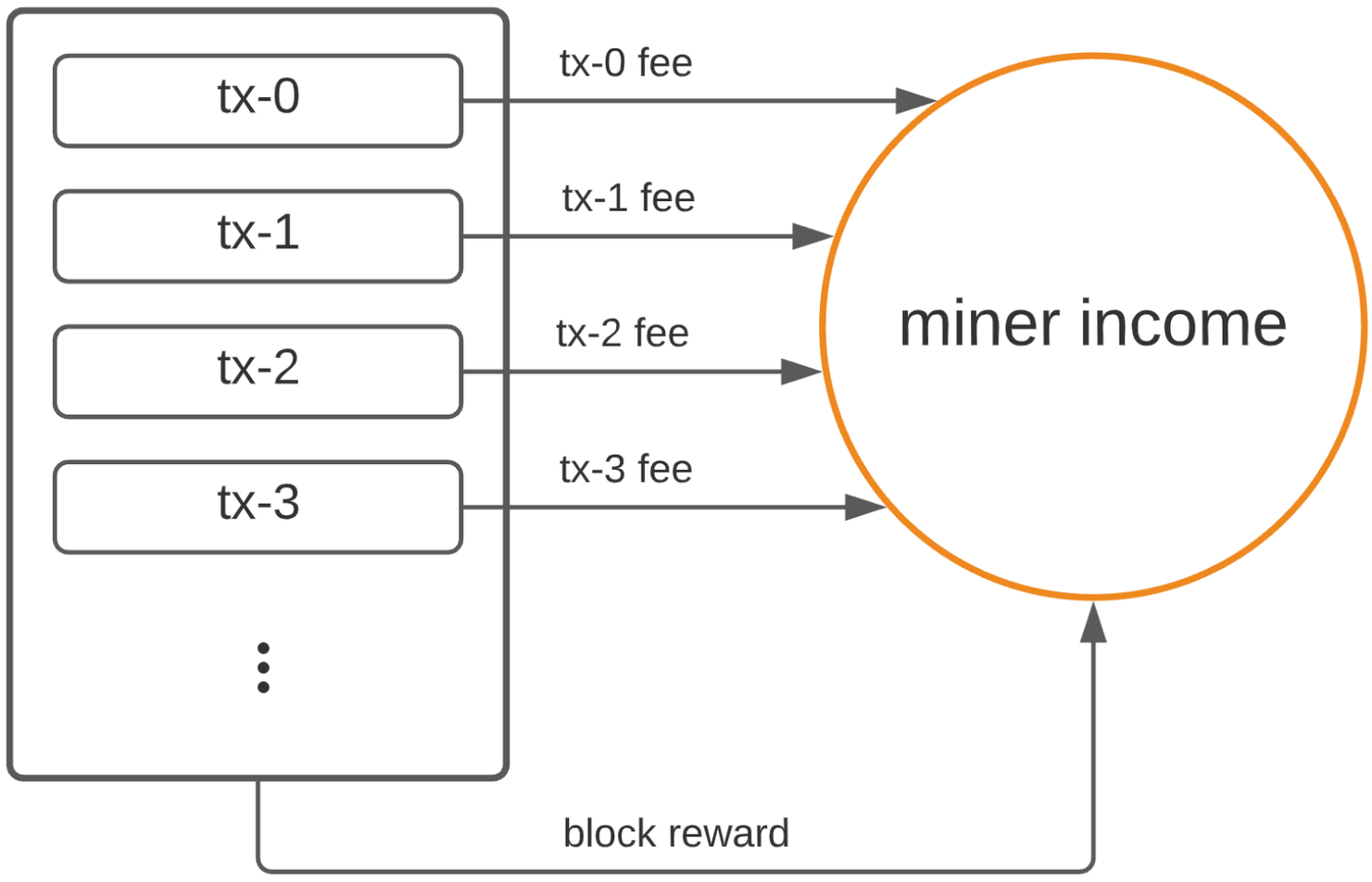
- Bitcoin, created by Satoshi Nakamoto, is the first blockchain system
- the main innovation of Bitcoin is incentive alignment
- miners make money by processing transactions (transaction fee) and by creating blocks (block reward)

HOW DOES TRANSACTION FEE WORK?

- Alice sends 3 coins to Bob:



MINER INCOME

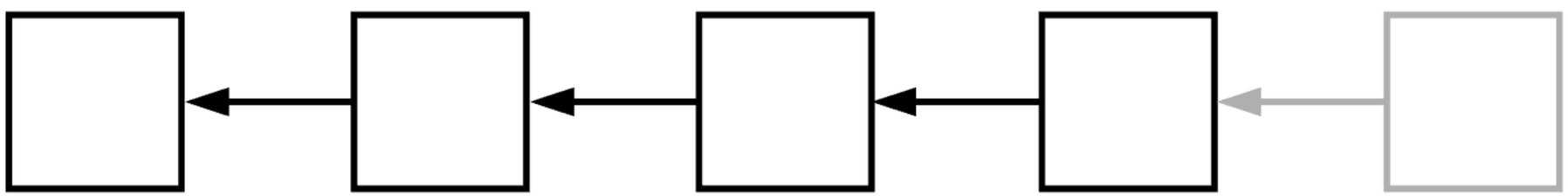


WHAT IS INCENTIVE ALIGNMENT?

- if you deviate from the rules, you will lose money
- instead of attacking the system, you can earn more money by actually participating (as a miner)
- this is the reason a native currency is necessary for permissionless blockchains like Bitcoin and Conflux

TRANSACTION FEE AS AN AUCTION

- the transaction fee market is basically an auction
- you are competing for the limited slots in the next block
- rational miners will prioritize transactions with higher fees
- as a result, the transaction fee also expresses the urgency



BITCOIN'S INCENTIVES ARE NOT PERFECT

- transaction fee market is not fully incentive compatible
- selfish mining might allow miners to earn more
- there is no direct incentive for validating the ledger and propagating transactions
- the mining algorithm and low block creation rate incentivizes miner centralization

TOPICS WE WILL COVER TODAY

- transaction fee in Bitcoin
- **transaction fee in Ethereum**
- transaction fee in Conflux
- transaction sponsorship
- blockchain applications and tools

THE COST OF ETHEREUM TRANSACTIONS

- Ethereum has smart contracts
- before creating a block, the miner has to execute all the transactions it wants to include in that block
- but two smart contract calls can be very different!

  execution cost: maybe it will run a loop for millions of cycles

  storage cost: maybe it will write millions of items to disk

THE PROBLEM OF TRANSACTION TERMINATION

- the halting problem: just by looking at the transaction, you cannot decide if it will ever finish running
- how to address this?

THE PROBLEM OF TRANSACTION TERMINATION

- the halting problem: just by looking at the transaction, you cannot decide if it will ever finish running
- how to address this?
  - limit the language so that you cannot even write such code (Bitcoin)
  - this works but might severely limit the types of applications you are able to express

THE PROBLEM OF TRANSACTION TERMINATION

- the halting problem: just by looking at the transaction, you cannot decide if it will ever finish running
- how to address this?
  - run for a fixed duration then terminate (Hyperledger)
  - this might harm determinism: code that timeouts on one machine might terminate on another, any they will be unable to reach consensus

THE PROBLEM OF TRANSACTION TERMINATION

- the halting problem: just by looking at the transaction, you cannot decide if it will ever finish running
- how to address this?
  - terminate after a certain number of steps (a certain amount of computation) has been performed
  - you pay for each computational unit so that miners get properly compensated

GAS IN ETHEREUM

- Ethereum: “pay-per-computational-step” model
- instead of pricing instructions directly in ETH (the price of which fluctuates), we use another metric: GAS
- each EVM instruction has a predefined GAS cost that is independent of the value of ETH
- the gas cost of a transaction is the sum of the cost for each instruction executed

GAS IN ETHEREUM -- INSTRUCTION GAS COSTS

ADD	3
SLOAD	200
LOG	375 + 8 * (number of bytes in log data)
SSTORE	20000 / 5000
CREATE	32000

source: https://github.com/djrtwo/evm-opcode-gas-costs/blob/master/opcode-gas-costs_EIP-150_revision-1e18248_2017-04-12.csv

GAS IN ETHEREUM -- GAS LIMIT

- when you send a transaction, you specify a **gas limit**: the maximum amount of GAS you are willing to spend
- if the execution reaches this limit before finishing, the transaction is reverted and you pay the corresponding fee
- if the gas used by the execution is below the limit, the remaining gas is refunded to your account
- this basically solves the halting problem: you can write infinite loops but you'll have to pay for them

GAS IN ETHEREUM -- GAS PRICE

- when you send a transaction, you specify a **gas price**: the amount of ETH you're willing to pay per gas
- you are free to choose what gas price to set, similarly to Bitcoin where you can set the transaction fee freely
- the miner is free to choose which transactions to include. it is likely to choose transactions with higher gas price.

GAS IN ETHEREUM -- HOW MUCH DO WE PAY?

- Let's say you can a transaction with

  gas limit: 50000

  gas price: 0.00000008 Ether (80 Gwei)

- you will have to make an up-front payment of

  50000 x 0.00000008 = 0.004 ETH

- if execution uses 41172 gas, after refund you'll effectively pay

  41172 x 0.00000008 = 0.00329376 ETH

GAS IN ETHEREUM -- EXAMPLE

- let us discuss this example contract:

example.sol

```
contract Test {
    uint256 num_a = 0;
    uint256 num_b = 0;

    function increment() public {
        num_a += 1;
        num_b += 1;




    }
}
```

GAS IN ETHEREUM -- EXAMPLE

- let us discuss this example contract:

example.sol

```
contract Test {
    uint256 num_a = 0;
    uint256 num_b = 0;

    function increment() public {
        num_a += 1;
        num_b += 1;

        // incrementing is essentially equivalent to:
        // uint256 current_value = number;
        // uint256 new_value = current_value + 1;
        // number = new_value;
    }
}
```

GAS IN ETHEREUM -- EXAMPLE

- let us discuss this example contract:

example.sol

```
contract Test {
    uint256 num_a = 0;
    uint256 num_b = 0;

    function increment() public {
        num_a += 1;
        num_b += 1;

        // solc compiles it like this:
        // SLOAD ADD SWAP3 POP POP DUP2 SWAP1 SSTORE POP PUSH1 0x1 DUP1
        // PUSH1 0x0 DUP3 DUP3 SLOAD ADD SWAP3 POP POP DUP2 SWAP1 SSTORE

    }
}
```

GAS IN ETHEREUM -- EXAMPLE

- let us discuss this example contract:

example.sol

```
contract Test {
    uint256 num_a = 0;
    uint256 num_b = 0;

    function increment() public {
        num_a += 1;
        num_b += 1;

        // we will simply treat it like this:
        // SLOAD ADD SSTORE SLOAD ADD SSTORE
        // gas cost: 2 x (3 + 200 + 5000) = 10406 GAS

    }
}
```

GAS IN ETHEREUM -- EXAMPLE

- what happens if we provide a gas limit of **5203** GAS?

example.sol

```
contract Test {
    uint256 num_a = 0;
    uint256 num_b = 0;

    function increment() public {
        num_a += 1; // this succeeds
                    // <- we run out of gas at this point
        num_b += 1;

        // the whole transaction is reverted
        // neither num_a nor num_b is updated
        // the sender pays for the executed 5203 GAS
    }
}
```

GAS IN ETHEREUM -- EXAMPLE

- what happens if we provide a gas limit of **12000** GAS?

example.sol

```
contract Test {
    uint256 num_a = 0;
    uint256 num_b = 0;

    function increment() public {
        num_a += 1; // this succeeds
        num_b += 1; // this also succeeds
                    // we finish the execution

        // the whole transaction succeeds
        // both num_a and num_b will be updated
        // the sender pays for the executed 10406 GAS, 1594 is refunded
    }
}
```

GAS IN ETHEREUM -- BASE COST

- transactions actually have a base cost of 21000 GAS
- covers checking the signature, storing the transaction, etc.
- sending a simple payment transaction will cost 21000 gas

WHY THE GAS ANALOGY?

- gasoline is what really moves your car, just as it moves the execution of smart contracts
- your car's gas consumption is roughly static, while the price of gasoline changes
- you could try stuffing dollar bills into the fuel tank of your car, but instead you buy gasoline using dollars, and use that to run it

TOPICS WE WILL COVER TODAY

- transaction fee in Bitcoin
- transaction fee in Ethereum
- **transaction fee in Conflux**
- transaction sponsorship
- blockchain applications and tools

GAS IN CONFLUX

- Conflux inherited the same fee mechanism from Ethereum, with a few changes:
  - limited gas refund
  - separate payment for execution and storage
  - gas can be sponsored

GAS IN CONFLUX -- LIMITED GAS REFUND

- the cost of a contract call might change, depending on when you call it. consider the following example:

example.sol

```
contract MaybeExpensive {
    uint256 num = 0;
    event E(uint8 n);

    function increment() public { num += 1; }

    function foo() public {
        if (num == 0) {
            for (uint8 ii = 0; ii < 1000; ii++) {
                emit E(ii);
            }
        }
    }
}
```

GAS IN CONFLUX -- LIMITED GAS REFUND

- triggering `foo()` might do almost nothing, or it might also emit 1000 events, depending on the current value of **`num`**

example.sol

```
contract MaybeExpensive {
    uint256 num = 0;
    event E(uint8 n);

    function increment() public { num += 1; }

    function foo() public {
        if (num == 0) {
            for (uint8 ii = 0; ii < 1000; ii++) {
                emit E(ii);
            }
        }
    }
}
```

GAS IN CONFLUX -- LIMITED GAS REFUND

- in Conflux, miners cannot know for sure the state on which the transaction is executed
- this makes it impossible to accurately estimate the profits of including certain transactions
- to make this estimation easier, at most 1/4 of the whole gas is refunded
- e.g. if gas limit is 10000 and the transaction execution used 6000 gas, the account will still be charged for 7500

GAS IN CONFLUX -- PAY SEPARATELY FOR STORAGE

- in Ethereum paying for storage is a one-time fee (gas for SSTORE), even though miners might have to store it forever
- consider this example: that 13 has to be stored on each machine maintaining the blockchain.

storage.sol

```
contract Storage {
    mapping (address => uint256) data;

    constructor() {
        data[msg.sender] = 13;
    }
}
```

GAS IN CONFLUX -- PAY SEPARATELY FOR STORAGE

- in Conflux, you need to deposit CFX for storage entries. this deposit is called storage collateral.
- miners earn interest on the collateralized storage
- if someone deletes or overwrites your entry, you get back your collateral

GAS IN CONFLUX -- PAY SEPARATELY FOR STORAGE

- so what you will pay is

(gas_used x gas_price) + *(1/16 CFX x storage_collateralized)*

… minus the refund of at most *1/4 x gas_used x gas_price*

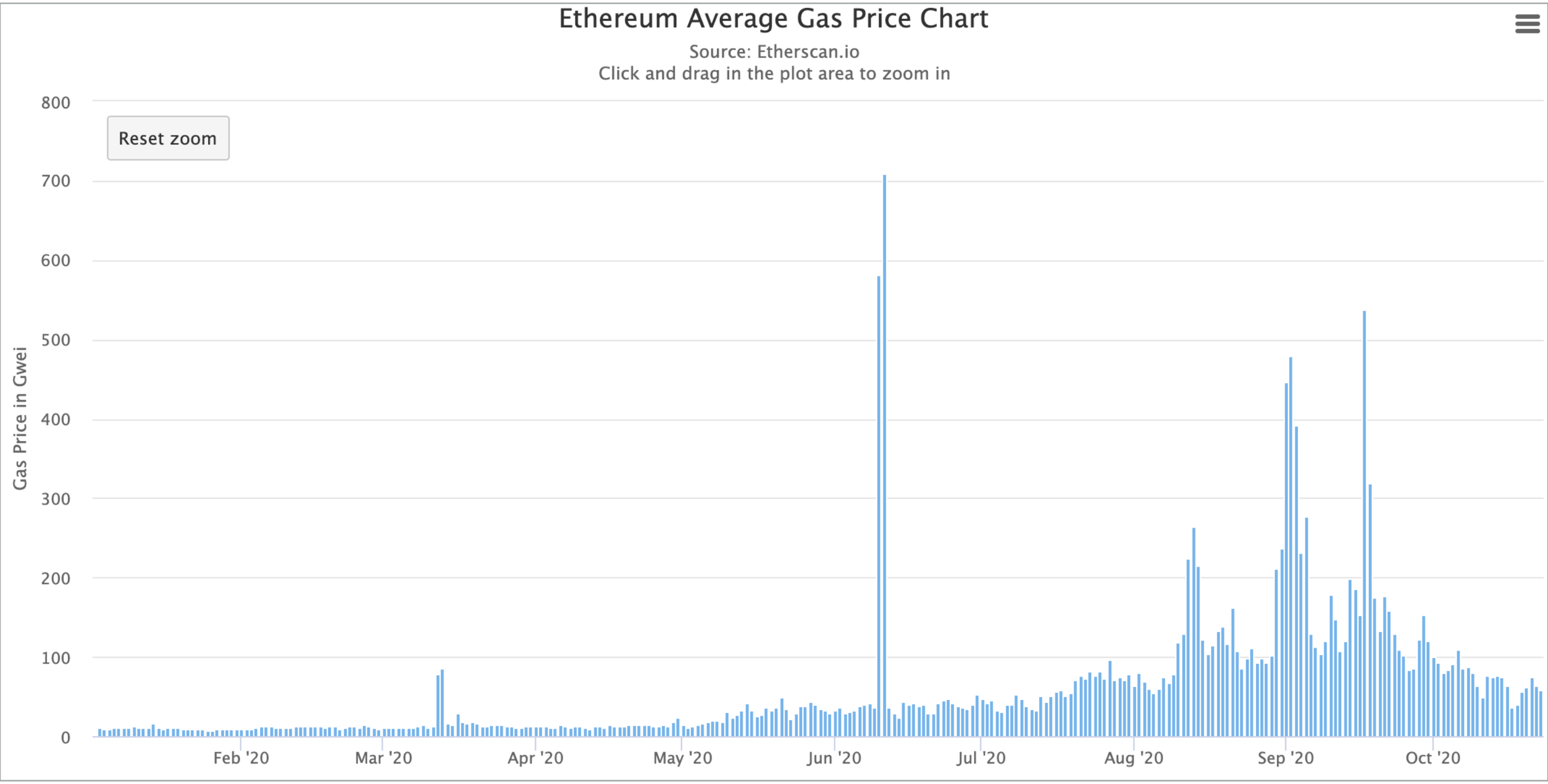
TOPICS WE WILL COVER TODAY

- transaction fee in Bitcoin
- transaction fee in Ethereum
- transaction fee in Conflux
- **transaction sponsorship**
- blockchain applications and tools

THE PROBLEMS WITH GAS -- USER EXPERIENCE

- Ethereum dapps: high entry barrier
  - get a wallet, acquire some ETH, …
- are people willing to pay for each transaction?
  - web 2.0: you have to pay for your server’s uptime but you don’t directly push this onto your customers; you have other revenue streams

THE PROBLEMS WITH GAS -- PRICES ON ETHEREUM



SPONSORSHIP MECHANISM IN CONFLUX

- someone can choose to sponsor transactions to a smart contract, i.e. pay on the sender's behalf
- example use case: MoonSwap
  - UniSwap: if you swap ETH for USDT, fees are around $4.1 at the time of writing
  - MoonSwap: would cost $0.4 at current price but in practice it's $0 because the transaction is sponsored

SPONSORSHIP MECHANISM IN CONFLUX

- there is a built-in contract called *SponsorWhitelistControl* address: `0x0888000000000000000000000000000000000001`
- **sponsor** calls `setSponsorForGas(...)` to become the sponsor of a smart contract. they will also send some CFX along that will be used to cover the gas costs.
- **contract** calls `addPrivilege(...)` to *"whitelist"* accounts (or use `0x0000000000000000000000000000000000000000`)

SPONSORSHIP MECHANISM IN CONFLUX

- let's say Alice deploys the following contract for her dapp

dapp.sol

```
contract MyCoolDapp {
    constructor() {
        address addr = 0x0888000000000000000000000000000000000001;
        SponsorWhitelistControl swc = SponsorWhitelistControl(addr);

        address[] memory a = new address[](1);
        a[0] = 0x0000000000000000000000000000000000000000;
        swc.add_privilege(a);
    }

    function doSomething() public {
        // ...
    }
}
```

SPONSORSHIP MECHANISM IN CONFLUX

- when Bob calls `doSomething()`, he has to cover gas and storage

dapp.sol

```
contract MyCoolDapp {
    constructor() {
        address addr = 0x0888000000000000000000000000000000000001;
        SponsorWhitelistControl swc = SponsorWhitelistControl(addr);

        address[] memory a = new address[](1);
        a[0] = 0x0000000000000000000000000000000000000000;
        swc.add_privilege(a);
    }

    function doSomething() public {
        // ...
    }
}
```

SPONSORSHIP MECHANISM IN CONFLUX

- at some point, Alice sends 10 CFX to the SWC contract through this call:

set_sponsor

```
swc.setSponsorForGas(
    0x8caddbc9f0360b6b96e32fc0a37ffa275ee139f6, // contract
    10000000000000000, // upper limit per transaction: 0.1 CFX
)
```

- she will most likely do this using the JavaScript SDK, or directly from Conflux Studio
- this means that at least 100 transactions will be sponsored

SPONSORSHIP MECHANISM IN CONFLUX

- when Bob calls `doSomething()` again, he does not need to pay anymore

dapp.sol

```
contract MyCoolDapp {
    constructor() {
        address addr = 0x0888000000000000000000000000000000000001;
        SponsorWhitelistControl swc = SponsorWhitelistControl(addr);

        address[] memory a = new address[](1);
        a[0] = 0x0000000000000000000000000000000000000000;
        swc.add_privilege(a);
    }

    function doSomething() public {
        // ...
    }
}
```

TRANSACTION SPONSORSHIP -- DEMO

TOPICS WE WILL COVER TODAY

- transaction fee in Bitcoin
- transaction fee in Ethereum
- transaction fee in Conflux
- transaction sponsorship
- **blockchain applications and tools**

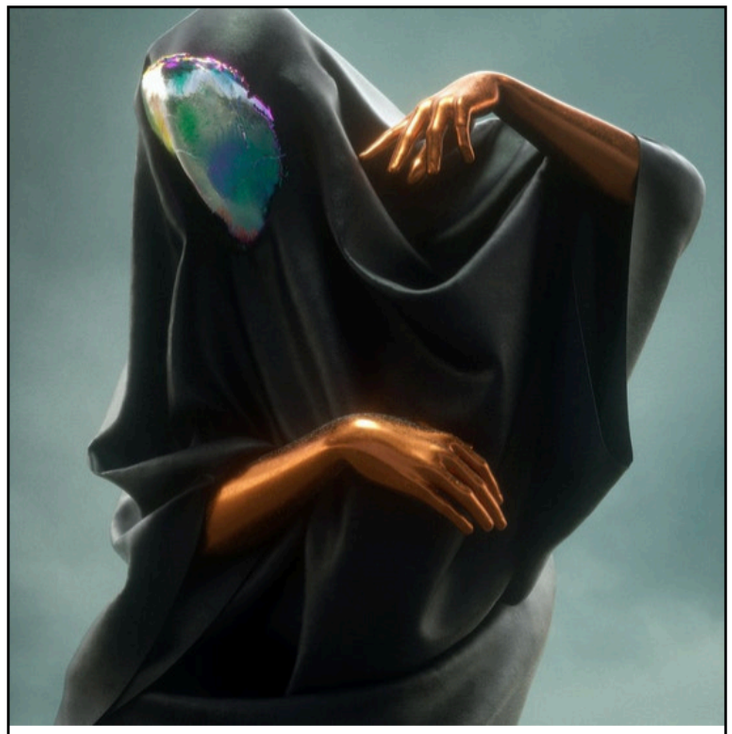
BLOCKCHAIN APPLICATIONS AND TOOLS

- tokens (ERC-20, NFT)
  - stablecoin
  - tokenized art
  - tokenized real-world assets

Skip Tracer

52.53Ξ ($20,97...
List price

ARTIST aeforia
OWNER cryptojack2



Ivy

–
List price

6Ξ ($2,467)
Current bid by @fafafofo

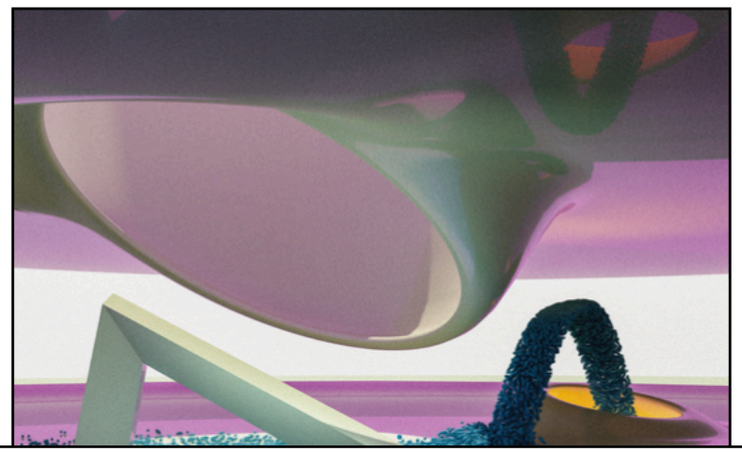
ARTIST aeforia
OWNER aeforia

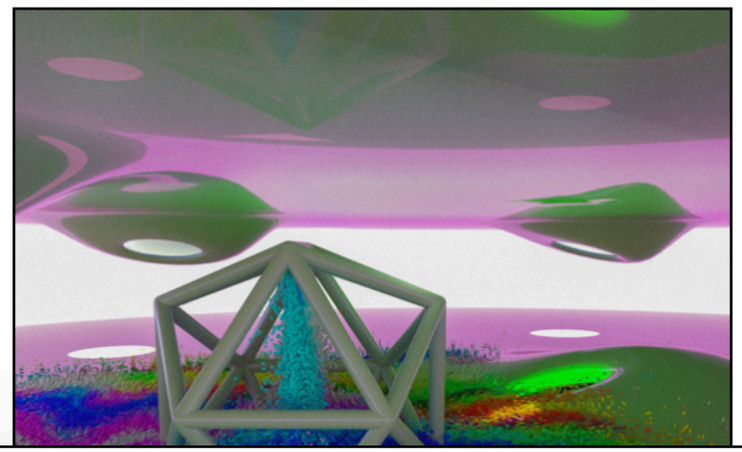
Paranoid

–
List price

17Ξ ($6,991)
Current bid by @doodl...

ARTIST etienecrau...
OWNER fafafofo





BLOCKCHAIN APPLICATIONS AND TOOLS

- DeFi
  - MoonDex
  - MoonSwap
- ShuttleFlow
  - a building block for DeFi applications
  - move tokens from other chains to Conflux
  - e.g. ETH -> cETH, BTC -> cBTC

MOONSWAP

0x1dBD...7636

Swap Pool Wallet

From Balance: 0

0.0 MAX cETH

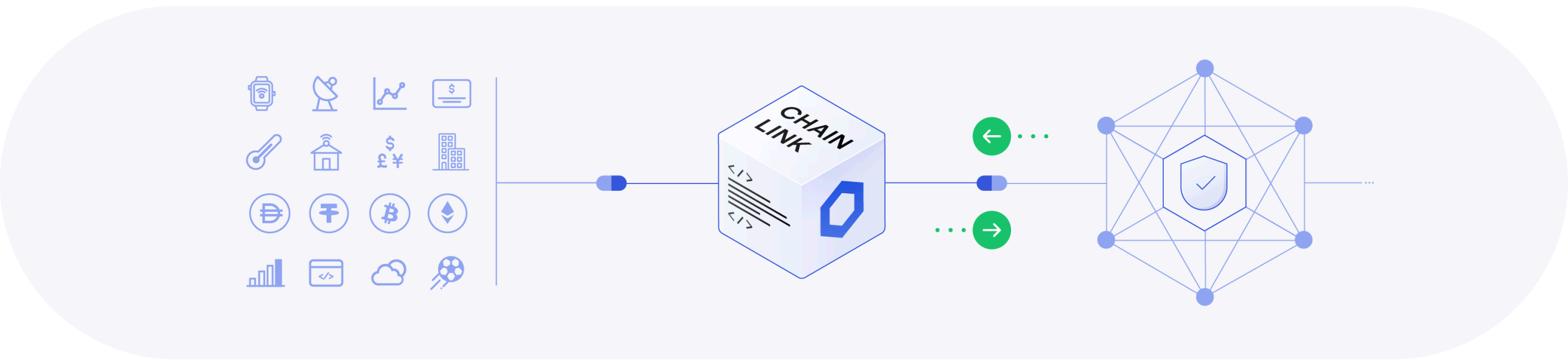
To Balance: 0

0.0 cUSDT

Enter an amount

BLOCKCHAIN APPLICATIONS AND TOOLS

- Oracles: allow your smart contract to interact with the world



TOPICS WE WILL COVER TODAY

- transaction fee in Bitcoin
- transaction fee in Ethereum
- transaction fee in Conflux
- transaction sponsorship
- blockchain applications and tools

RESOURCES

- ETH Gas Station
  ethgasstation.info
- Bitcoin's Underlying Incentives
  queue.acm.org/detail.cfm?id=3168362
- Zohar et al. Redesigning bitcoin’s fee market. 2019.
  arxiv.org/abs/1709.08881
- Eyal, Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable. 2013.
  www.cs.cornell.edu/~ie53/publications/btcProcFC.pdf

RESOURCES

- What is Ethereum Gas?
  blockgeeks.com/guides/ethereum-gas
- Staking & Collateral For Storage On Conflux Network
  medium.com/conflux-network/conflux-economic-model-staking-collateral-for-storage-on-conflux-network-cb4c8c150e3
- Block Reward Components Under the Network's Mining Incentive Model
  medium.com/conflux-network/conflux-networks-economic-model-block-reward-components-under-the-network-s-mining-incentive-ceadd8f8408f