

When you send ERC20 tokens

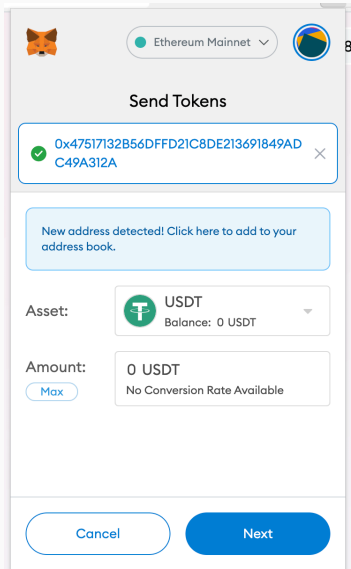
in-house seminar

Jeongho Jeon

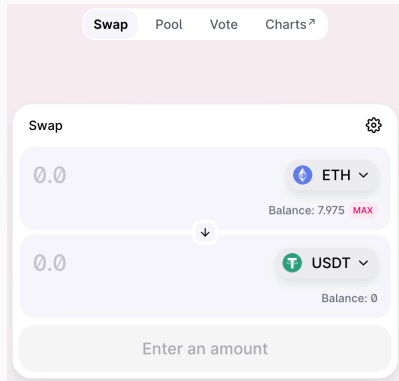
2022-05-03

DSRV

Metamask wallet & Uniswap



The image shows the Metamask wallet interface on the 'Send Tokens' screen. At the top, there is a fox icon, a network selector set to 'Ethereum Mainnet', and a profile icon. The title 'Send Tokens' is centered. Below it, a recipient address '0x47517132B56DFFD21C8DE213691849ADC49A312A' is entered with a green checkmark and a close button. A blue notification box states: 'New address detected! Click here to add to your address book.' The 'Asset' section shows 'USDT' with a balance of '0 USDT'. The 'Amount' section shows '0 USDT' with a 'Max' button and a note 'No Conversion Rate Available'. At the bottom are 'Cancel' and 'Next' buttons.



The image shows the Uniswap interface on the 'Swap' screen. At the top are tabs for 'Swap', 'Pool', 'Vote', and 'Charts'. The 'Swap' section has a settings gear icon. It features two input fields: the top one for 'ETH' with a balance of '7.975 MAX' and the bottom one for 'USDT' with a balance of '0'. A double-headed arrow indicates the swap direction. At the bottom is a large button labeled 'Enter an amount'.

sign tx that DApp has built

build tx and sign it

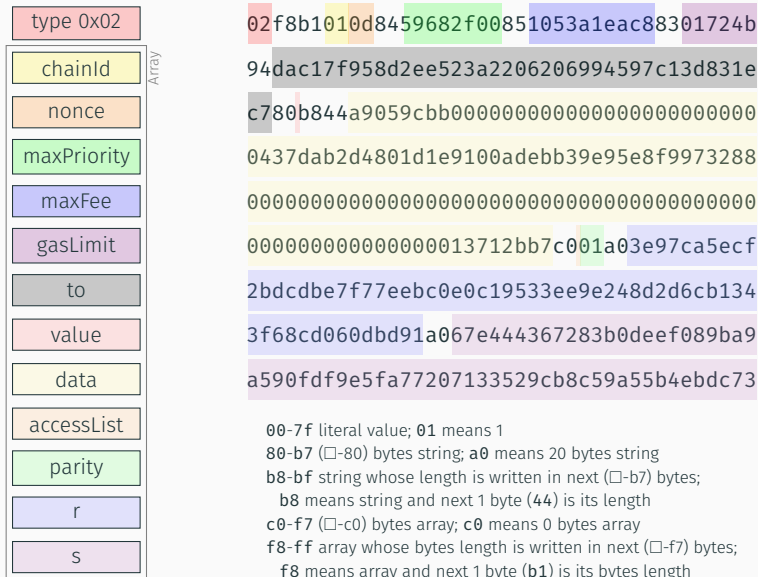
Transaction (tx)

type 0x02	0x02 = EIP-1559 tx
chainId	0x01 = mainnet
nonce	the number of txs that this account has sent. No gap between serial numbers
maxPriority	maximum gas price tip that tx is willing to pay (for miners, unit wei)
maxFee	maximum gas price (tip included, unit wei)
gasLimit	maximum gas amount that tx can use
to	token contract address. not token receipt. empty when creating contracts
value	transferred ether (unit wei = 10^{-18} ether). zero (empty \emptyset) for ERC20 transfers
data	smart contract function selector. see next slide
accessList	for future optimizations
parity	for public key recovery from ECDSA signature, $r^{-1}(sR - zG)$
r	ECDSA signature, kG
s	ECDSA signature, $k^{-1}(z + rd_A)$

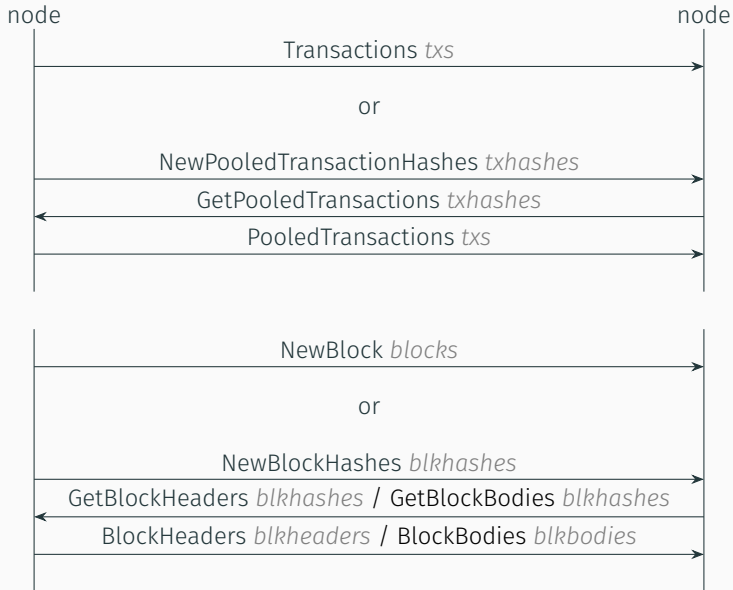
Transaction data field

- [illegible]

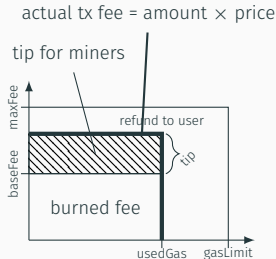
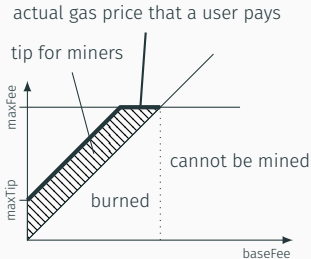
Transaction serialization by RLP (Recursive Length Prefix)



p2p protocol, devp2p

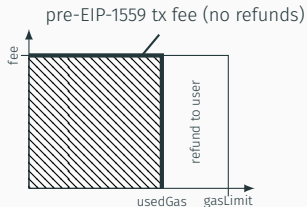


EIP-1559 gas fee



baseFee is determined from used gas ratio each block. If maximum block gas limit is used, **baseFee** increases by +12.5%. If half of gas limit is used, it is not changed. If no gas is used, it decreases by 12.5%.

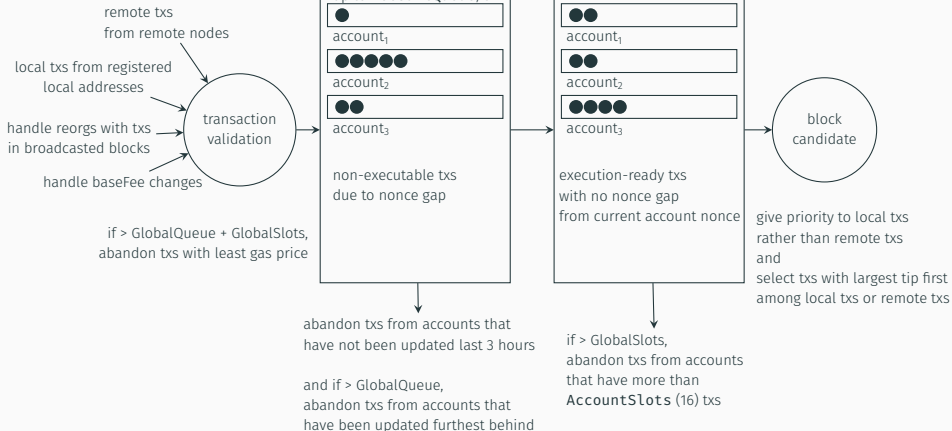
The maximum block gas limit is decided by miners within $\pm \frac{1}{1024}$ (about 0.1%) difference from the previous block.



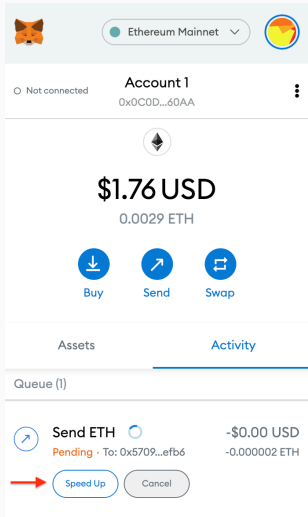
mempool, a.k.a. Dark Forest

simplified Geth mempool (not effective mining pool mempool)

may replace the same nonce tx
with higher gas price



Metamask Speed Up & Cancel



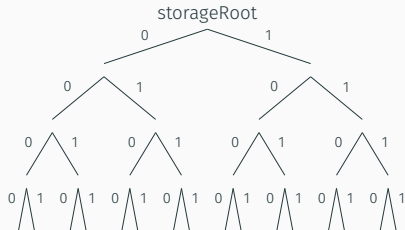
- You may “speed up” or “cancel” a pending transaction. Both require higher gas price.
- “Speed Up” resends the same transaction except the gas price.
- “Cancel” resends “transfer zero ethers to self” transaction with the same nonce and higher gas price.
- Metamask advises to use at least 10% higher **maxPriority** and at least 30% higher **maxFee**.
- Geth replaces a transaction with the same nonce if both **maxPriority** and **maxFee** are at least 10% (**PriceBump** parameter) higher.
- But transaction replacement is not guaranteed all times.

EVM Execution

First, jump to transfer function by a selector,

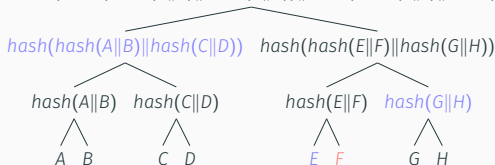
```
...
dup1
0x95d89b41
eq
tag_22
jumpi
dup1
0xa9059cbb
eq
tag_23
jumpi
dup1
0xc0324c77
eq
...
address public owner; // at storage 0x00
bool public paused; // at 0x00 next to owner
uint public _totalSupply; // at 0x01
mapping(address => uint) public balances; // at keccak(address||2)
uint public basisPointsRate; // at 0x03
uint public maximumFee; // at 0x04
mapping (address, => mapping (address_2 => uint)) public allowed; // at keccak(address_2||keccak(address,||5))
mapping (address => bool) public isBlackListed; at keccak(address||6)
string public name; // at 0x07
string public symbol; // at 0x08
uint public decimals; // at 0x09
address public upgradedAddress; // at 0x0a
bool public deprecated; // at 0x0a next to upgradedAddress
...
balances[msg.sender] = balances[msg.sender].sub(_value);
balances[_to] = balances[_to].add(sendAmount);
...
sub(exp(0x2, 0xa0), 0x1)
caller /* = msg.sender */
and
0x0
swap1
dup2
mstore
0x2
0x20
mstore
0x40
swap1
keccak256
sload /* storage load */
...
```

storage in tree style
(not consecutive space)



Merkle Tree

$$\text{root} = \text{hash}(\text{hash}(\text{hash}(A\|B)\|\text{hash}(C\|D))\|\text{hash}(\text{hash}(E\|F)\|\text{hash}(G\|H)))$$



Integrity

root is changed if any leaf is changed.

You can compare roots only.

Inclusion Proof (Merkle Proof)

If Alice knows a root already, you can convince her of the existence of any leaf, F .

You send her a Merkle proof: F , E (left), $\text{hash}(G\|H)$ (right), $\text{hash}(\text{hash}(A\|B)\|\text{hash}(C\|D))$ (left).

Alice may compute a root with a Merkle proof only, and compare two roots.

You cannot fool her with nonexistent leaves.

Receipt (execution output)

type 0x02	0x02 = EIP-1559 tx
statusCode	tx failure 0, tx success 1
cumulativeGas	used gas from the start of block up to the end of this tx
bloomFilter	Bloom filter for quick searches on contract address and log topics
logs	array of logs, log = (contract address, array of log topics (max 4), log data)

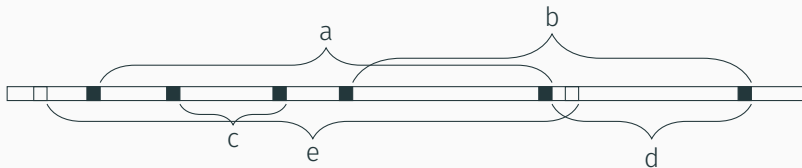
Consensus requires that all nodes make the same receipts after transaction execution. Receipts are not block data. Only Merkle tree root of receipts (**receiptRoot**) is stored in block header.

`event Transfer(address indexed from, address indexed to, uint value);`
writes a log:

<code>0xdac17f958d2ee523a2206206994597c13d831ec7</code>	contract address
<code>0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef</code>	first topic, <i>keccak("Transfer(address,address,uint)")</i>
<code>0x00</code>	second topic (first indexed) from
<code>0x00</code>	third topic (second indexed) to
<code>0x00</code>	cheaper log data (non-indexed fields) value

You may use web3.js to monitor log events: `web3.eth.getPastLogs()`,
`TetherContract.getPastEvents()`, `TetherContract.events.Transfer()`,
`TetherContract.events.allEvents()`, and `web3.eth.subscribe('logs')`

Bloom filter



Set bits from many hash functions of values. Then we can test bits from hash functions of new values. Bloom filter may tell that not-included-values (“d”) are included by small chance (false positive). But Bloom filter **never** tell that included-values are not included (no false negative). When we want to know whether any data is included without large data transfer, we use it.

Ethereum uses three bit ranges of keccak instead of three hash functions. It makes 2048 (2^{11}) bits Bloom filter with 0-10th, 16-26th, and 32-42nd bits (11 bits from 0-1st, 2-3rd, and 4-5th bytes, respectively) from LSB of *keccak()* results.

Account

balance

ethers that this account holds (unit wei)

storageRoot

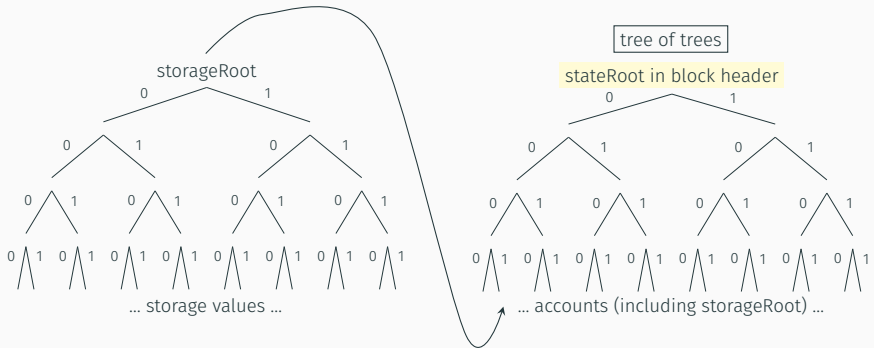
root of storage tree

nonce

the number of txs that this account has sent

codeHash

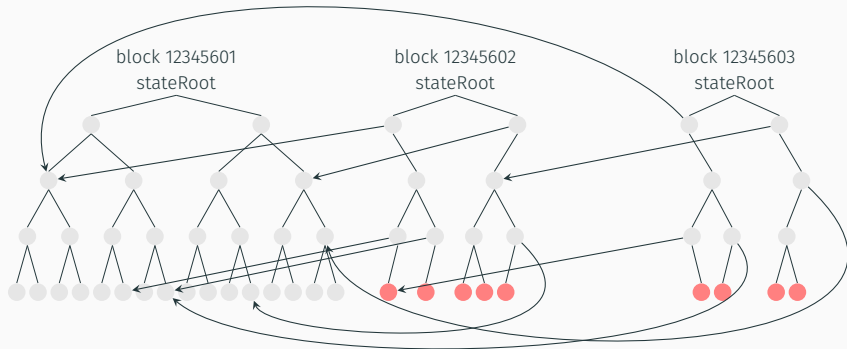
hash of smart contract code. empty in the case of user account (EOA)



Block Header

parentHash	hash of parent block header
unclesHash	hash of uncle block headers
beneficiary	miner address (coinbase)
stateRoot	Merkle tree root of all accounts
txRoot	Merkle tree root of all transactions. (key: sequence, value: transaction)
receiptRoot	Merkle tree root of all receipts (execution outputs). (key: sequence, value: receipt)
logsBloom	Bloom filter for quick searches on logs in all receipts
difficulty	PoW (Ethash) difficulty
number	block number
gasLimit	maximum gas amount that all transactions in this block can use
gasUsed	gas amount that all transactions in this block used actually
timestamp	when this block was mined
extraData	miner identity in many cases
mixHash	calculated PoW target from nonce
nonce	PoW value that miner found. cf. account nonce
baseFee	EIP-1559 base fee that is burnt

Archive Node



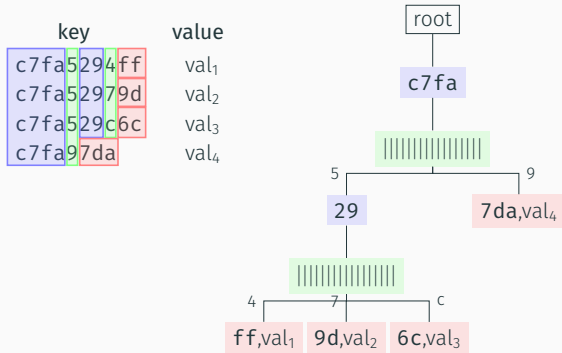
Red circles are changed. State (and storage) trees share unchanged parts. Archive nodes store whole states from the genesis block. It takes about 10 Tb. Normal (Geth) nodes need the latest state only, but store recent 128 blocks (about 25 minutes, over 600 Gb) for handling reorg.

In fact, state is not in a binary tree. It is in a Patricia tree.

2^{256} keys are more than the number of atoms in universe. Ethereum state is sparse (almost empty).

We can compress common parts of keys into single tree nodes. We use series of nibbles, 4 bits key fragments.

The cons is complex code that splits and combines nodes when nodes are inserted and deleted.



Extension node represents common part of keys.

Branch node has up to 16 (nibble) children.

Also it may have its own value if there is a key that ends at this node.

Leaf node contains both remaining part of a key and a value.

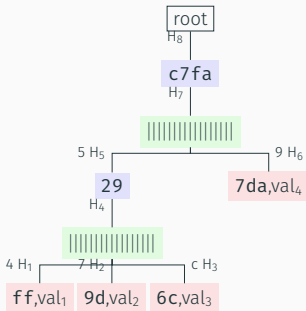
And state is stored in a disk (not a memory).

RLP serializes nodes from bottom to up.

Results up to 32 bytes are used intact, but keccak results if they are longer than 32 bytes.

i.e., key: keccak value (longer than 32 bytes) or value (otherwise) & value: RLP-serialized node
(We assume that val_1 - val_4 are 32 bytes here.)

The odd and even number of nibbles in extension nodes and leaf nodes determines a prefix.



$H_1 = \text{keccak}(20, ff, val_1)$, 20 = leaf node with even nibbles

$H_2 = \text{keccak}(20, 9d, val_2)$

$H_3 = \text{keccak}(20, 6c, val_3)$

$H_4 = \text{keccak}(\emptyset, \emptyset, \emptyset, \emptyset, H_1, \emptyset, \emptyset, H_2, \emptyset, \emptyset, \emptyset, \emptyset, H_3, \emptyset, \emptyset, \emptyset, \emptyset)$

$H_5 = \text{keccak}(00, 29, H_4)$, 00 = extension node with even nibbles

$H_6 = \text{keccak}(37, da, H_4)$, 37 = leaf node with odd nibbles

$H_7 = \text{keccak}(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, H_5, \emptyset, \emptyset, \emptyset, H_6, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$

$H_8 = \text{keccak}(00, cf, fa, H_7)$, (17 = extension node with odd nibbles)

Nodes and other objects are stored in physical LevelDB key-value store.

Geth uses one byte key-prefix to avoid collisions. For example,

“h” + blockNumber + “n” (for block header hash) and “c” + codeHash.

But no prefix is required to find tree nodes (tries).

Geth may store data older than 90000 blocks (about 2 weeks) that is considered immutable

in slow but less expensive disk. This “freezer” (or “ancient”) database uses append-only flat files.

Closing

In future, we will use SSZ (Simple Serialize, see Jiyun's `cpp_ssz`) instead of RLP and more efficient accumulator (such as Verkle) instead of Patricia tree.

Special thanks to Korea Ethereum Research Group colleagues and our captain, Jay Park!