Ethereum Blocks

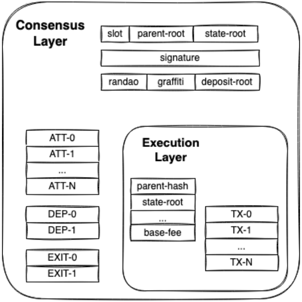
Saravanan Vijayakumaran

Department of Electrical Engineering
Indian Institute of Technology Bombay

February 19, 2024

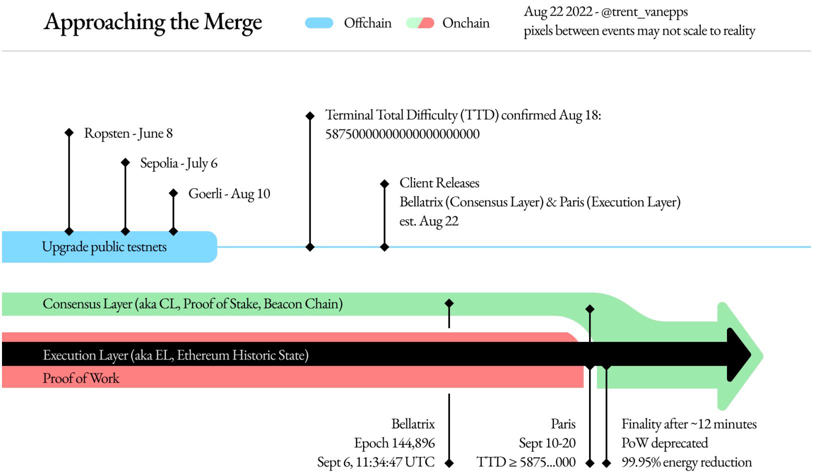
Ethereum Blocks

- Ethereum launched as a PoW chain in July 2015
- In Sept 2022, it transitioned to proof-of-stake (the Merge)
- Ethereum node components
  - **Execution client**: Executes transactions and updates world state
  - **Beacon chain client:** Implements the PoS algorithm to achieve consensus on the execution client blocks
- Ethereum blocks



Source: Ethereum Blog

The Merge



Approaching the Merge

Offchain Onchain

Aug 22 2022 - @trent_vanepps
pixels between events may not scale to reality

Ropsten - June 8

Sepolia - July 6

Goerli - Aug 10

Upgrade public testnets

Terminal Total Difficulty (TTD) confirmed Aug 18:
58750000000000000000000

Client Releases
Bellatrix (Consensus Layer) & Paris (Execution Layer)
est. Aug 22

Consensus Layer (aka CL, Proof of Stake, Beacon Chain)

Execution Layer (aka EL, Ethereum Historic State)

Proof of Work

Bellatrix
Epoch 144,896
Sept 6, 11:34:47 UTC

Paris
Sept 10-20
TTD ≥ 5875...000

Finality after ~12 minutes
PoW deprecated
99.95% energy reduction

Source: Ethereum Blog

Ethereum 1.0 Block Header

Block = (Header, Transactions, Uncle Headers)

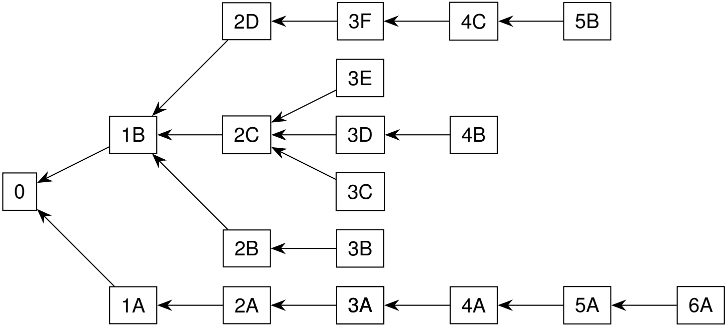
Block Header

`parentHash`	32	bytes
`ommersHash`	32	bytes
`beneficiary`	20	bytes
`stateRoot`	32	bytes
`transactionsRoot`	32	bytes
`receiptsRoot`	32	bytes
`logsBloom`	256	bytes
`difficulty`	≥ 1	byte
`number`	≥ 1	byte
`gasLimit`	≥ 1	byte
`gasUsed`	≥ 1	byte
`timestamp`	≤ 32	bytes
`extraData`	≤ 32	bytes
`mixHash`	32	bytes
`nonce`	8	bytes

Uncle Blocks in Ethereum 1.0

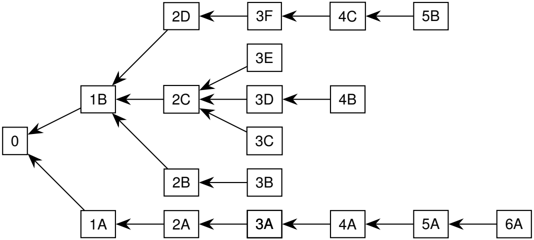
- Block = (Header, Transactions, Uncle Header List)
- `ommersHash` in block header is hash of uncle header list
- Ommer = Gender-neutral term that means “sibling of parent”
- **Problem:** Low inter-block time leads to high stale rate
  - Stale blocks do not contribute to network security
- **Solution:** Reward stale block miners and also miners who include stale block headers
- Rewarded stale blocks are called uncles or ommers
  - Transactions in uncle blocks are invalid
  - Only a fraction of block reward goes to uncle creator; no transaction fees
- How to resolve forks in the presence of uncle blocks?
  - Greedy Heaviest Observed Subtree (GHOST) protocol proposed by Sompolinsky and Zohar in December 2013
  - Ethereum 1.0 used a simpler version of GHOST
- Ethereum 2.0 also uses a version of GHOST called LMD GHOST

GHOST Protocol



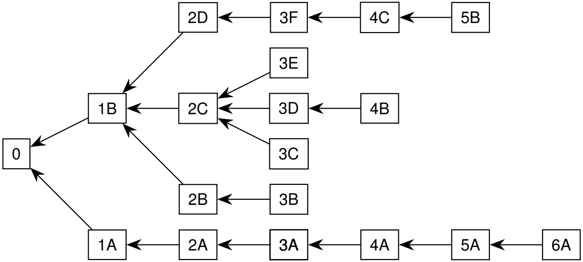
- A policy for choosing the main chain in case of forks
- Given a block tree T, the protocol specifies GHOST(T) as the block representing the main chain
- Mining nodes calculated GHOST(T) locally and mine on top of it
- Heaviest subtree rooted at fork is chosen

GHOST Protocol



```
function $\textsc{Children}_T(B)$
    return Set of blocks with $B$ as immediate parent
end function
function $\textsc{Subtree}_T(B)$
    return Subtree rooted at $B$
end function
function GHOST($T$)
  $B \leftarrow$ Genesis Block
  while True do
    if $\textsc{Children}_T(B) = \emptyset$ then return $B$ and exit
    else$B \leftarrow \mathrm{argmax}_{C \in \textsc{Children}_T(B)} |\textsc{Subtree}_T(C)|$
    end if
  end while
end function
```

GHOST Protocol Example



- Suppose an attacker secretly constructs the chain 1A, 2A,. . . , 6A
- All other blocks are mined by honest miners
- Honest miners' efforts are spread over multiple forks
- Longest chain rule gives 0,1B,2D,3F,4C,5B as main chain
    - Shorter than attacker's chain
- GHOST rule gives 0,1B,2C,3D,4B as main chain

Eth2 Execution Client Block Header

Block = (Header, Transactions, Uncle Headers)

Block Header		
`parentHash`	32	bytes
`ommersHash`	32	bytes
`beneficiary`	20	bytes
`stateRoot`	32	bytes
`transactionsRoot`	32	bytes
`receiptsRoot`	32	bytes
`logsBloom`	256	bytes
`difficulty`	1	byte
`number`	≥ 1	byte
`gasLimit`	≥ 1	byte
`gasUsed`	≥ 1	byte
`timestamp`	≤ 32	bytes
`extraData`	≤ 32	bytes
`prevRandao`	32	bytes
`nonce`	8	bytes
`baseFeePerGas`	≥ 1	byte

Block Header Fields Deprecated in Eth2

Block Header

`parentHash`	32	bytes
`ommersHash`	32	bytes
`beneficiary`	20	bytes
`stateRoot`	32	bytes
`transactionsRoot`	32	bytes
`receiptsRoot`	32	bytes
`logsBloom`	256	bytes
`difficulty`	1	byte
`number`	≥ 1	byte
`gasLimit`	≥ 1	byte
`gasUsed`	≥ 1	byte
`timestamp`	≤ 32	bytes
`extraData`	≤ 32	bytes
`prevRandao`	32	bytes
`nonce`	8	bytes
`baseFeePerGas`	≥ 1	byte

- `ommersHash` = Hash of an empty list
- `difficulty` = Set to zero
- `nonce` = Set to 8 zero bytes
- `mixHash` is replaced with `prevRandao`
    - RANDAO is a pseudorandom value generated by validators in the PoS consensus algorithm

Fields in the Execution Client Header

Block Header

parentHash	32	bytes
ommersHash	32	bytes
beneficiary	20	bytes
stateRoot	32	bytes
transactionsRoot	32	bytes
receiptsRoot	32	bytes
logsBloom	256	bytes
difficulty	1	byte
number	≥ 1	byte
gasLimit	≥ 1	byte
gasUsed	≥ 1	byte
timestamp	≤ 32	bytes
extraData	≤ 32	bytes
prevRandao	32	bytes
nonce	8	bytes
baseFeePerGas	≥ 1	byte

- `parentHash` = Keccak-256 hash of parent block header
- `beneficiary` = Destination address of block reward and transaction fees
- `stateRoot` = Root hash of world state trie after all transactions are applied
- `transactionsRoot` = Root hash of trie populated with all transactions in the block
- `number` = Number of ancestor blocks
- `timestamp` = Unix time at block creation
- `extraData` = Arbitrary data; validators identify themselves in this field

receiptsRoot

Block Header		
parentHash	32	bytes
ommersHash	32	bytes
beneficiary	20	bytes
stateRoot	32	bytes
transactionsRoot	32	bytes
receiptsRoot	32	bytes
logsBloom	256	bytes
difficulty	1	byte
number	≥ 1	byte
gasLimit	≥ 1	byte
gasUsed	≥ 1	byte
timestamp	≤ 32	bytes
extraData	≤ 32	bytes
prevRandao	32	bytes
nonce	8	bytes
baseFeePerGas	≥ 1	byte

- `receiptsRoot` is the root hash of transaction receipts trie
- A transaction receipt contains logs emitted by smart contracts
- Smart contracts can write to logs using events

```
event Transfer(address indexed from, address indexed to,
    uint256 value);
```

logsBloom

Block Header

parentHash	32	bytes
ommersHash	32	bytes
beneficiary	20	bytes
stateRoot	32	bytes
transactionsRoot	32	bytes
receiptsRoot	32	bytes
logsBloom	256	bytes
difficulty	1	byte
number	≥ 1	byte
gasLimit	≥ 1	byte
gasUsed	≥ 1	byte
timestamp	≤ 32	bytes
extraData	≤ 32	bytes
prevRandao	32	bytes
nonce	8	bytes
baseFeePerGas	≥ 1	byte

- Bloom filter = Probabilistic data structure for set membership queries
- Each transaction receipt contains Bloom filter of addresses and “topics”
- `logsBloom` is the OR of all transaction receipt Bloom filters
- KECCAK-256 of the logger’s address and indexed topics are used to set 3 bits out of 2048
- Light clients can efficiently retrieve only transactions of interest

Fee-related Fields

Block Header		
`parentHash`	32	bytes
`ommersHash`	32	bytes
`beneficiary`	20	bytes
`stateRoot`	32	bytes
`transactionsRoot`	32	bytes
`receiptsRoot`	32	bytes
`logsBloom`	256	bytes
`difficulty`	1	byte
`number`	≥ 1	byte
`gasLimit`	≥ 1	byte
`gasUsed`	≥ 1	byte
`timestamp`	≤ 32	bytes
`extraData`	≤ 32	bytes
`prevRandao`	32	bytes
`nonce`	8	bytes
`baseFeePerGas`	≥ 1	byte

- `gasUsed` is the total gas used by all transactions in the block
- `gasLimit` is the maximum gas which can be used (currently 30 million)
- `baseFeePerGas` is the minimum required transaction fees per unit of gas
  - Burned by the protocol
  - Updated every block depending of how far gasUsed is from a target limit of 15 million

Base Fee Calculation

- Proposed in EIP 1559; included in London hard fork (Aug 2021)

$$\begin{aligned}
\text{gasTarget} &= \frac{\text{gasLimit}}{2} \\
\delta &= \frac{\text{gasUsed} - \text{gasTarget}}{4 \times \text{gasLimit}} \times \text{baseFeePerGas} \\
\text{baseFeePerGas}_{\text{new}} &= \text{baseFeePerGas} + \delta
\end{aligned}$$

- Previously gas prices were a first-price auction
- Users had to guess the gas price which would result in block inclusion of their transactions
- Base fees gives an indication of blockspace demand
- Users can pay a tip to miners via priority fee

References

- Yellow paper `https://ethereum.github.io/yellowpaper/paper.pdf`
- GHOST paper `https://eprint.iacr.org/2013/881`
- Ethereum blog post `https://blog.ethereum.org/2021/11/29/how-the-merge-impacts-app-layer`
- Solidity events and logs
- Upgrading Ethereum book `https://eth2book.info/`
- An Economic Analysis of EIP-1559 `https://timroughgarden.org/papers/eip1559.pdf`