How is the Consensus Reached in Ethereum?

@juinc

Outline

- Overview
- Section 10: Blocktree to Blockchain
- Section 11: Block Finalisation
- Appendix J: ETHASH

Review

Please refer to my previous talk:

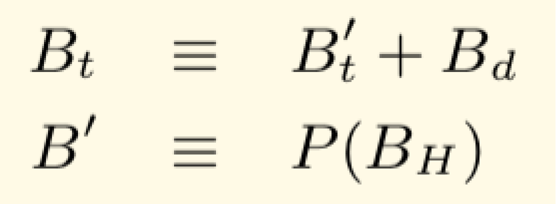
- Let's Make a Blockchain (Basic)
- How is Your Contract Executed in Ethereum? (EVM)

Overview

- Consensus: general agreement on the World State
- The consensus can be reached through different schemes:
  - Proof of Work
  - Proof of Stake
  - ...
- The process of applying or wrapping a block is called finalization

Section 10: Blocktree to Blockchain

- The canonical blockchain is a path from root to leaf through the entire block tree
- Define Total Difficulty:



$$\begin{aligned} B_t &\equiv B'_t + B_d \\ B' &\equiv P(B_H) \end{aligned}$$

- Choose the path that has the most computation done upon it, or, the most difficult path

Section 11: Block Finalization

Peer (Non-miner) v.s. Miner

- Step 1: Validate (Determine) Ommers
- Step 2: Validate (Determine) Transactions
- Step 3: Apply Rewards
- Step 4: Validate (Determine) State and Nonce

Recall: State Transition Function

$$
\begin{aligned}
\boldsymbol{\sigma}_{t+1} &\equiv \Pi(\boldsymbol{\sigma}_t, B) \\
B &\equiv (..., (T_0, T_1, ...)) \\
\Pi(\boldsymbol{\sigma}, B) &\equiv \Omega(B, \Upsilon(\Upsilon(\boldsymbol{\sigma}, T_0), T_1)...)
\end{aligned}
$$

where

$$\boldsymbol{\sigma}' = \Upsilon(\boldsymbol{\sigma}, T)$$

(explained in my last talk)

Step 1: Validate (Determine) Ommers

$$\|B_{\mathbf{U}}\| \leqslant 2 \bigwedge_{U \in B_{\mathbf{U}}} V(U) \;\wedge\; k(U, P(B_H)_H, 6)$$

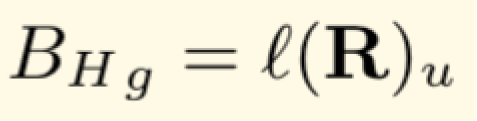
where

$$k(U, H, n) \equiv \begin{cases} false & \text{if} \quad n = 0 \\ s(U, H) & \\ \quad \vee\, k(U, P(H)_H, n-1) & \text{otherwise} \end{cases}$$

and

$$s(U, H) \equiv (P(H) = P(U) \;\wedge\; H \neq U \;\wedge\; U \notin B(H)_{\mathbf{U}})$$

Step 2: Validate (Determine) Transactions



$$B_{H_g} = \ell(\mathbf{R})_u$$

Step 3: Apply Rewards

$$\begin{aligned}
\Omega(B, \boldsymbol{\sigma}) &\equiv \boldsymbol{\sigma}' : \boldsymbol{\sigma}' = \boldsymbol{\sigma} \quad \text{except:} \\
\boldsymbol{\sigma}'[{B_{H}}_{c}]_b &= \boldsymbol{\sigma}[{B_{H}}_{c}]_b + (1 + \frac{\|B_{\mathbf{U}}\|}{32})R_b \\
\forall_{U \in B_{\mathbf{U}}} &: \\
\boldsymbol{\sigma}'[U_c]_b &= \boldsymbol{\sigma}[U_c]_b + (1 + \frac{1}{8}(U_i - {B_{H}}_{i}))R_b
\end{aligned}$$

where

$$R_b = 5 \times 10^{18}$$

Step 4: Validate (Determine) State and Nonce

A. Validate

$$\Pi(\boldsymbol{\sigma}, B) \equiv \Omega(B, \ell(\mathbf{R})_{\boldsymbol{\sigma}})$$

where

$$\mathbf{R}[n]_{\boldsymbol{\sigma}} = \begin{cases} \Gamma(B) & \text{if} \quad n < 0 \\ \Upsilon(\mathbf{R}[n-1]_{\boldsymbol{\sigma}}, B_{\mathbf{T}}[n]) & \text{otherwise} \end{cases}$$

Step 4: Validate (Determine) State and Nonce

A. Validate (Cont.)

and

$$\mathbf{R}[n]_u = \begin{cases} 0 & \text{if} \quad n < 0 \\ \Upsilon^g(\mathbf{R}[n-1]_{\boldsymbol{\sigma}}, B_{\mathbf{T}}[n]) & \\ \quad +\mathbf{R}[n-1]_u & \text{otherwise} \end{cases}$$

and

$$\mathbf{R}[n]_{\mathbf{l}} = \Upsilon^{\mathbf{l}}(\mathbf{R}[n-1]_{\boldsymbol{\sigma}}, B_{\mathbf{T}}[n])$$

Step 4: Validate (Determine) State and Nonce

A. Validate (Cont.)

and

$$\Gamma(B) \equiv \begin{cases} \boldsymbol{\sigma}_0 & \text{if} \quad P(B_H) = \varnothing \\ \boldsymbol{\sigma}_i : \texttt{TRIE}(L_S(\boldsymbol{\sigma}_i)) = P(B_H)_{H_r} & \text{otherwise} \end{cases}$$

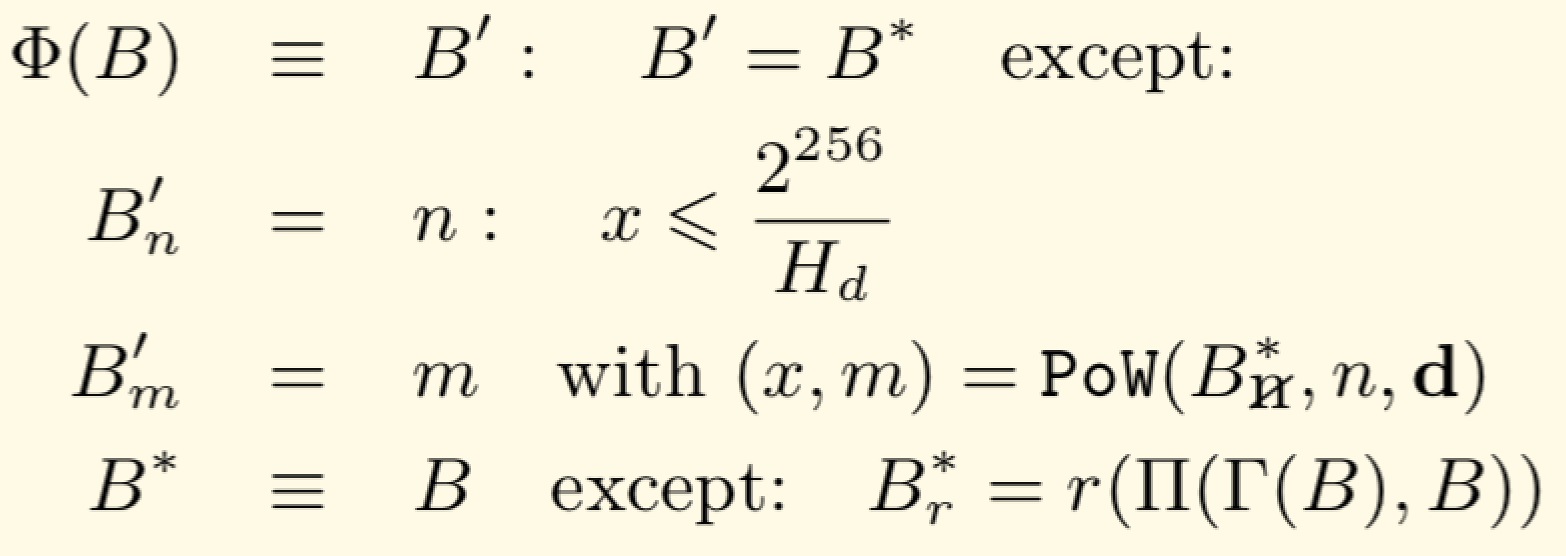
and

$$\boldsymbol{\sigma}' = \Upsilon(\boldsymbol{\sigma}, T)$$

Step 4: Validate (Determine) State and Nonce

B. Determine

Block Transition Function

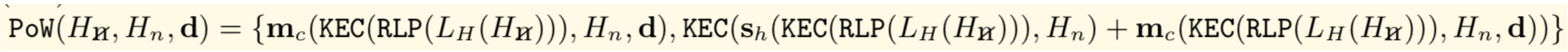


$$
\begin{aligned}
\Phi(B) &\equiv B' : \quad B' = B^* \quad \text{except:} \\
B'_n &= n : \quad x \leqslant \frac{2^{256}}{H_d} \\
B'_m &= m \quad \text{with } (x, m) = \mathtt{PoW}(B^*_{\not{n}}, n, \mathbf{d}) \\
B^* &\equiv B \quad \text{except:} \quad B^*_r = r(\Pi(\Gamma(B), B))
\end{aligned}
$$

Step 4: Validate (Determine) State and Nonce

B. Determine (Cont.)

Proof-of-work Function (Ethash)



$$\mathtt{PoW}(H_{\cancel{n}}, H_n, \mathbf{d}) = \{\mathbf{m}_c(\mathtt{KEC}(\mathtt{RLP}(L_H(H_{\cancel{n}}))), H_n, \mathbf{d}), \mathtt{KEC}(\mathbf{s}_h(\mathtt{KEC}(\mathtt{RLP}(L_H(H_{\cancel{n}}))), H_n) + \mathbf{m}_c(\mathtt{KEC}(\mathtt{RLP}(L_H(H_{\cancel{n}}))), H_n, \mathbf{d}))\}$$

Appendix J: Ethash

Why Ethash?

- ASIC-resistance
- Light Client Verifiability

ref: Ethash

Appendix J: Ethash

1. Determine Cache Size and Dataset (DAG) Size
2. Determine Seed Hash
3. Determine Cache
    - Calculate Initial Cache from Seed and Cache Size
    - Calculate Cache from RandMemoHash (RMH) Algorithm
4. Determine Dataset (DAG) from Cache
5. Determine Mix from Dataset (DAG), Header, and Nonce
6. Compare Mix to Target, repeat 5 until Mix is smaller than Target (PoW)

Conclusion

- By looking into block finalization process, we can understand more on the whole picture of ethereum
- Implementation details might be different from what yellow paper describes