Role of Geth

Components of Ethereum

- **P2P network**
  - connecting participants and propagating transactions and blocks of verified transactions, based on standardized "gossip" protocol
  - TCP port 30303, runs a protocol called *ÐΞVp2p*.
- **Consensus rules**
  - governing what constitutes a transaction and what makes for a valid state transition.
- **Transactions**
  - network messages that include (among other things) a sender, recipient, value, and data payload.
- **State machine**
  - processing transactions according to the consensus rules.
- **Data structures**
  - Ethereum's state is stored locally on each node as a database (usually Google's LevelDB), which contains the transactions and system state in a serialized hashed data structure called a *Merkle Patricia Tree*.
- **Consensus algorithm**
  - decentralizing control over the blockchain, by forcing participants to cooperate in the enforcement of the consensus rules.
- **Economic security**
  - PoW → PoS
- **Clients**

- Geth, Nethermind, Besu, Erigon(Execution clients)
- Prysm, Lighthouse, Teku, Nimbus, Lodestar(Consensus clients)

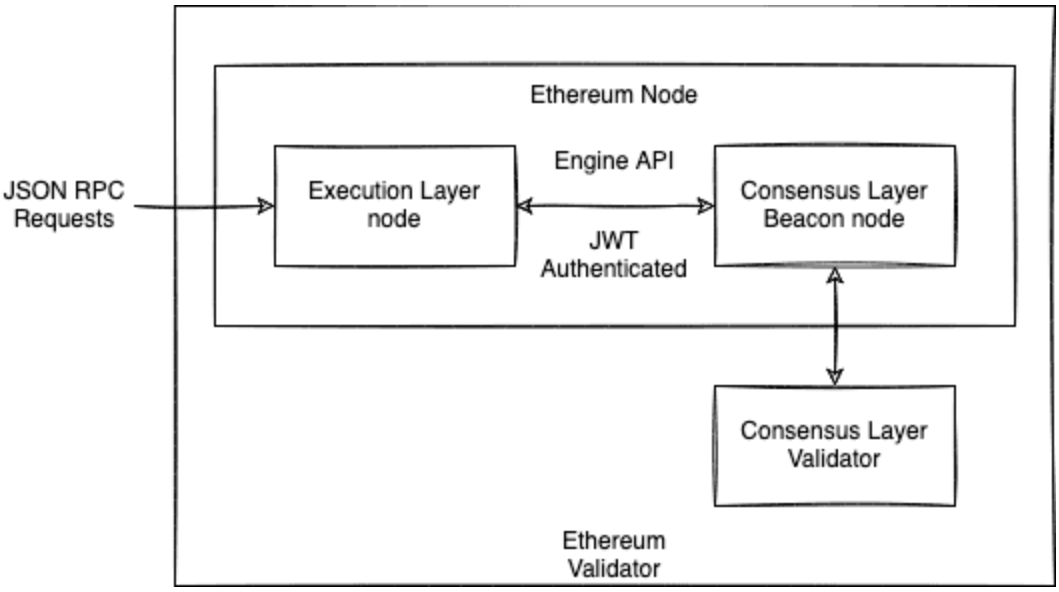
> **consensus rules vs. consensus algorithm?**
> → Implementing consensus rules on Ethereum with certain logic is consensus algorithm

What are nodes and clients?

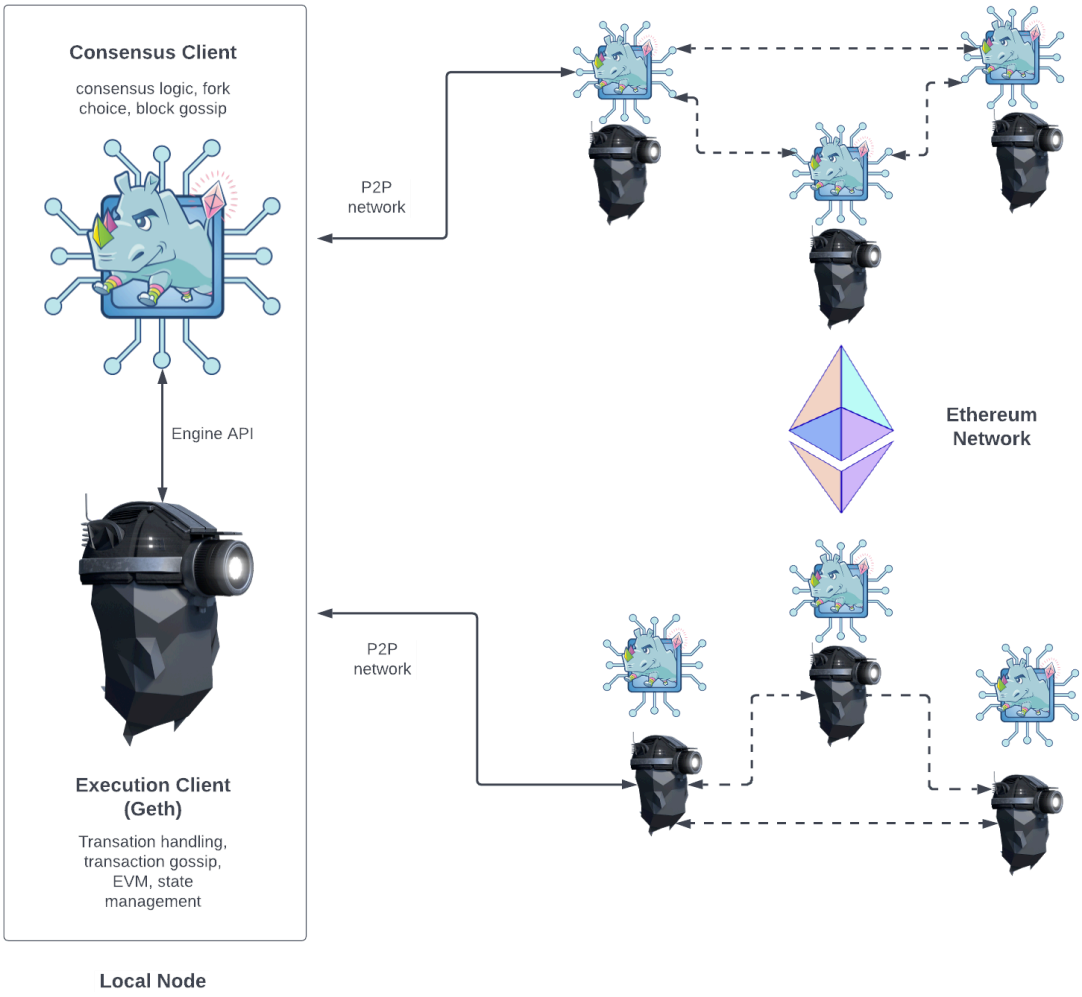
A **"node"** is any instance of Ethereum client software that is connected to other computers also running Ethereum software, forming a network. A client is an implementation of Ethereum that **verifies data** against the protocol rules and **keeps the network secure**. A node has to run two clients: a **consensus client** and an **execution client**.

- The execution client **listens to new transactions** broadcasted in the network, **executes** them in EVM, and **holds the latest state and database** of all current Ethereum data.
- The consensus client **implements the proof-of-stake consensus algorithm**, which enables the network to achieve agreement based on **validated data from the execution client**. There is also a third piece of software, known as a **'validator'** that can be added to the consensus client, allowing a node to **participate in securing network**.

These clients work together to **keep track of the head of the Ethereum chain** and **allow users to interact with Ethereum network**. The modular design with multiple pieces of software woking together is called encapsulated complexity. This approach made it easier to execute "The Merge" seamlessly, makes client software easier to maintain and develop, and enables the reuse of individual clients, for example, in the layer 2 ecosystem.



Node Architecture



Execution client is responsible for **transaction handling**, **transaction gossip**, **state management** and **supporting the EVM**. However, Geth is not responsible for **block building, block gossiping or handling consensus logic**. These are in the remit of the consensus client.

The two clients each connect to their own perspective peer-to-peer networks. This is because the **execution clients gossip transactions over their P2P network enabling them to manage their local transaction pool**. The **consensus clients gossip blocks over their P2P network, enabling consensus and chain growth**.

For this two-client structure to work, **consensus clients must be able to pass bundles of transactions to Geth to be executed**. Executing the transactions locally is how the client validates that the transactions do not violate any Ethereum

rules and that the proposed update to Ethereum's state is correct. Likewise, **when the node is selected to be a block producer the consensus client must be able to request bundles of transactions from Geth to include in the new block**. This inter-client communication is handled by a local RPC connection using the Engine API.

What does Geth do?

As an execution client, Geth is responsible for **creating the execution payloads** - the list of transactions, updated state trie plus other execution related data - that consensus clients include in their blocks. Geth is also responsible for **re-executing transactions** that arrive in new blocks to ensure they are valid. Executing transactions is done on Geth's embeded computer, known as the EVM.

What does the consensus client do?

The consensus client deals with **all the logic that enables a node to stay in sync with the Ethereum network.** This includes receiving blocks from peers and running a fork choice algorithm to ensure the node always follows the chain with the greatest accumulation of attestations.

The consensus client has its **own P2P network**, separate from the network that connects execution clients to each other. The consensus client itself doesn't participate in attesting to or proposing blocks - this is done by a validator which is an optional add-on to a consensus client. A **consensus client without a validator only keeps up with the head of the chain**, allowing the node to stay synced. This enables a user to transact with Ethereum using their execution client, confident that they are on the right chain.

Validators

Validators can be added to consensus clients if 32 ETH have been sent to the deposit contract. The validator client comes bundled with the consensus client and can be added to a node at any time. The validator handles **attestations and**

block proposals. They enable a node accrue rewards or lose ETH via penalties or slashing. Running the validator software also makes a node eligible to be selected to propose a new block.

Decoding Geth

WIP

Reference

- https://github.com/ethereum/go-ethereum
- https://github.com/ethereumbook/ethereumbook/blob/develop/01what-is.asciidoc#references
- https://ethereum.org/en/developers/docs/nodes-and-clients/#client-diversity
- https://ethereum.org/en/developers/docs/nodes-and-clients/client-diversity/
- https://hackmd.io/@danielrachi/engine_api