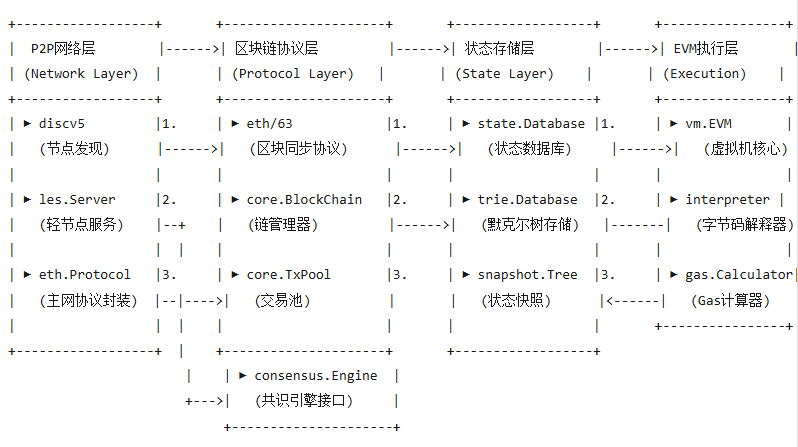
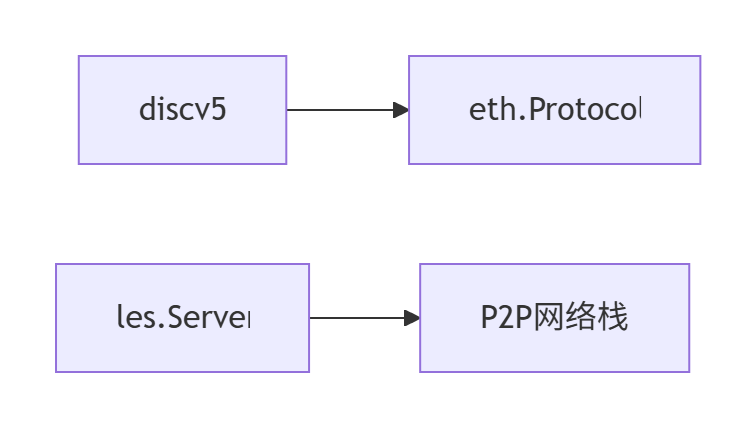
Geth分层架构图及关键模块说明，采用UML风格描述各层关系：



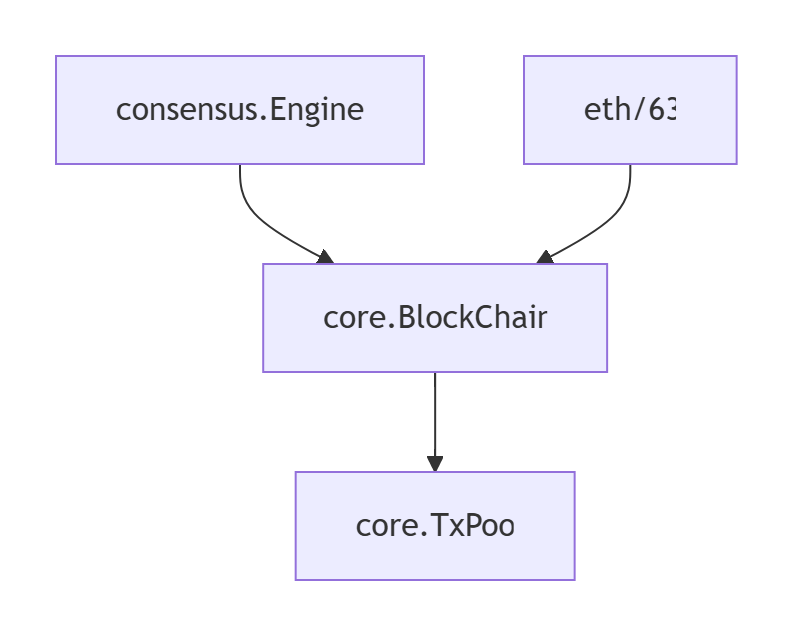
### 各层关键模块解析

#### ​****1. P2P网络层****​

​

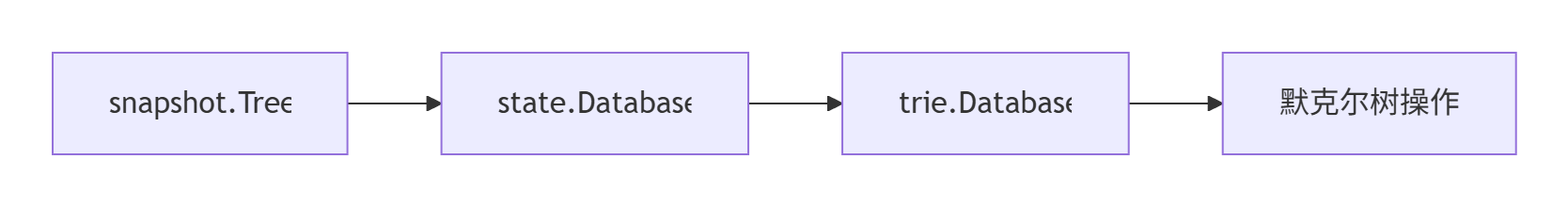
* **discv5**​：分布式节点发现协议，使用Kademlia算法维护节点路由表
* ​**eth.Protocol**​：以太坊主网协议封装，处理：
  + 区块/交易广播
  + 节点能力协商
  + 协议版本管理(eth/62, eth/63)
* ​**les.Server**​ (轻节点协议核心)：
  + 按需提供区块头/交易证明
  + 状态查询服务(通过trie.Proof)
  + 带宽限流机制

#### ​****2. 区块链协议层****​

​

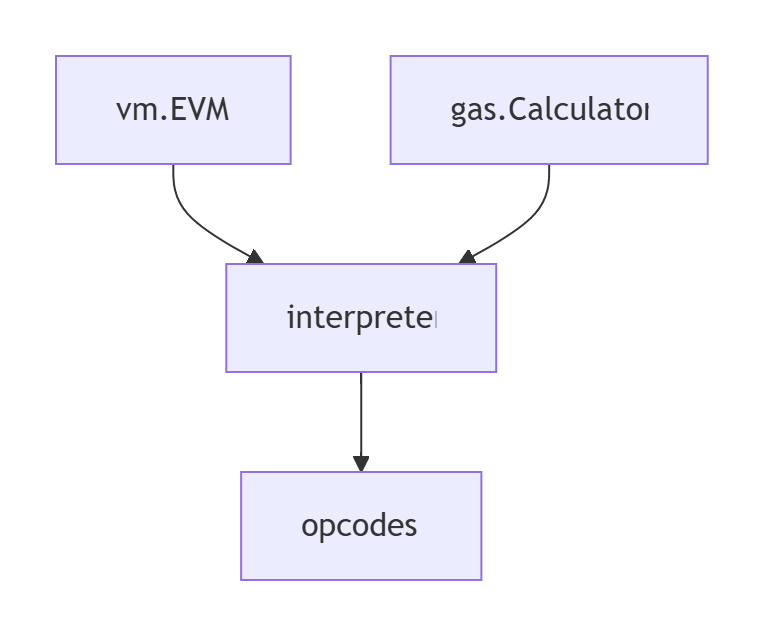
* **core.BlockChain**​：区块链管理中枢
  + 维护链结构(ParentHash链)
  + 处理重组(reorg)
  + 调用EVM执行交易
* ​**core.TxPool**​：交易池管理
  + 交易验证（Nonce/Gas校验）
  + 优先级排序（基于Gas Price）
  + 交易广播（通过P2P层）
* ​**eth/63**​：同步协议实现
  + 区块头快速同步(Header download)
  + 状态快照下载(snap sync)
  + 交易收据验证
* ​**core/types**​：核心数据结构
  + Header：区块头（ParentHash, StateRoot...）
  + Block：区块体（TxHash, Uncles...）
  + Transaction：交易数据

#### ​****3. 状态存储层****​

​ **trie**​：默克尔帕特里夏树实现

* + 树节点哈希计算
  + 状态证明生成（trie.Proof）
  + 增量更新机制
* ​**state.Database**​：状态存储抽象层
  + 账户状态管理（balance/nonce/storage）
  + 合约代码存取
  + 提供EVM状态访问接口
* ​**snapshot.Tree**​：状态快照
  + 加速状态访问
  + 支持快速同步（snap sync）

#### ​****4. EVM执行层****​

​

* **vm.EVM**​：执行环境核心
  + 构建执行上下文（BlockNumber, Timestamp...）
  + 管理内存/堆栈/合约调用栈
* ​**interpreter**​：字节码解释器
  + 逐指令解析操作码
  + 执行状态变更指令
* ​**gas.Calculator**​：Gas机制实现
  + 操作码Gas成本表
  + 动态Gas计算（内存扩展等）
  + 退款机制处理

### 关键跨层交互流程

**1.交易执行**​  
TxPool → 交易 → BlockChain → 调用 EVM → 读写 state.Database → 更新 trie 状态树

​2.**状态同步**​  
eth/63 → 获取状态数据 → 构建 trie 节点 → 写入 state.Database → 生成 StateRoot

​3.**轻节点验证**​  
les.Server → 获取 trie.Proof → 验证状态 → 返回精简数据

​4.**共识交互**​  
consensus.Engine → 最终确认区块 → BlockChain 提交 → state.Database 持久化状态

该架构展现Geth模块化设计：下层为上层提供服务接口，共识机制贯穿各层，状态层作为执行核心枢纽，网络层确保去中心化通信。