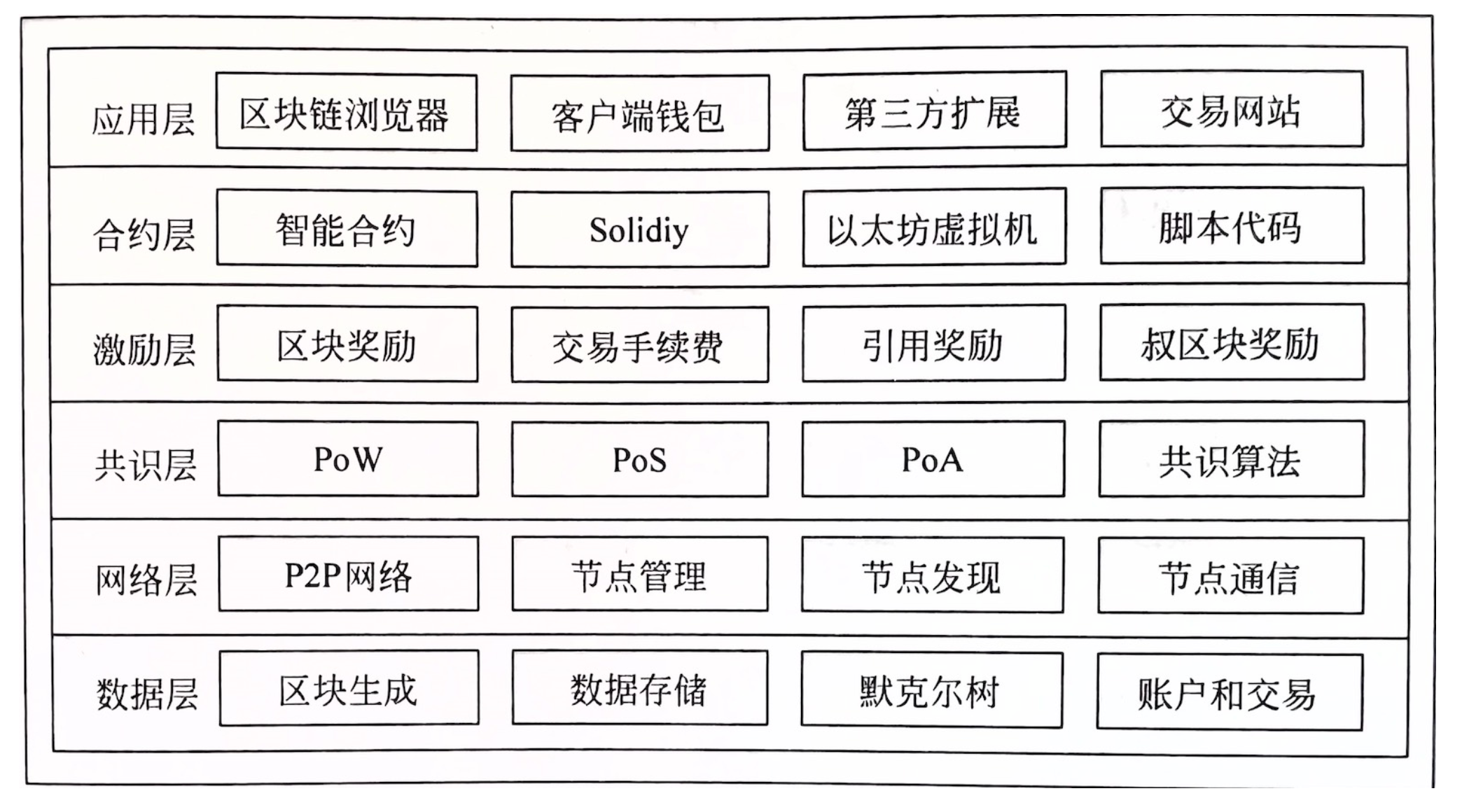
Ethereum技术细节

1. 架构图



二 数据层

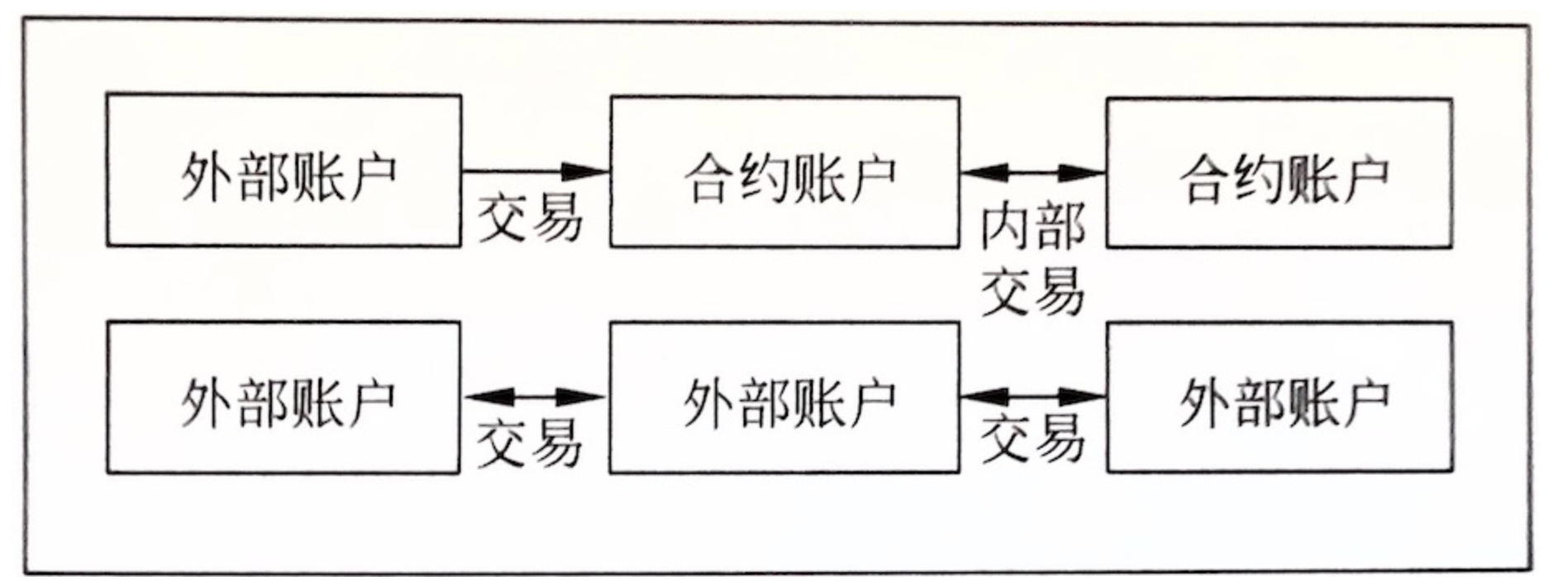
位于区块链架构的最底层，以太坊的数据层使用 LevelDB 数据库以键值对的形式存储数据，并采用 Merkle Patricia Tree（MPT）作为数据结构来组织和管理。每个区块构成区块链的基本单位，包含了一系列交易列表。这些交易的执行将改变账户状态，从而更新以太坊的全局状态。

1. 账户原理

1.1 账户类型

在以太坊中，账户分为两种：外部账户和合约账户。外部账户由用户创建，并可以通过私钥签名来发送交易至另一个外部账户或合约账户。交易可以是简单的价值转移，或者是触发合约账户中代码的执行。这些代码的执行可能包括代币转移、状态修改、逻辑运算执行、新合约创建或其他合约的调用等。

合约账户由合约代码控制，并且不能自主发起交易；它们仅在接收到交易后被动执行。这些交易可以来自外部账户的直接调用或其他合约账户的代码调用。合约账户之间还可以通过“消息”或“内部交易”进行通信，这些消息或内部交易虽类似于交易，但它们不由外部账户生成，也不会被序列化记录到区块链中，仅存在于以太坊的执行环境中。当一个合约向另一个合约发送内部交易时，接收合约的代码将被执行，从而触发相应的逻辑。以下图示展示了他们之间的调用逻辑：



1.2账户状态

nonce: 对于外部账户，这个字段表示该账户发出的交易数量，用以防止重复交易（双花问题）。对于合约账户，nonce 表示该账户创建的合约数量。

balance: 账户的余额，单位为 wei。1 ether 等于 1018 wei。

storageRoot: 这是一个指向 Merkle Patricia Tree（MPT）根节点的哈希值，树中存储了账户的状态信息。

codeHash: 在外部账户中，这个字段通常是一个空字符串的哈希值；而在合约账户中，它表示账户中 EVM 代码（编译后的合约字节码）的哈希。当合约账户接收到一个消息调用时，相关代码将被执行。不同于其他字段，codeHash 是不可变的，它用作从状态数据库检索相应 EVM 代码的索引。

1.3账户生成

外部账户生成 外部账户是通过一对公私钥生成的。公钥用作账户地址，而私钥则存储在一个加密的 JSON 文件中，该文件可以用任何文本编辑器打开和查看。在创建账户时，私钥会被设置密码加密，因此，只有同时拥有私钥及其密码，才能证明对账户的所有权。这些钥匙文件存放在以太坊节点的 data/keystore 目录下。在同一个节点的 data/ 目录下的所有账户构成一个钱包，而一个以太坊节点可以通过设置不同的 data/ 目录来创建和管理多个钱包。

合约账户生成

新的合约账户是通过部署智能合约来创建的。不同于外部账户，合约账户不由私钥管理，而是受到合约部署者的控制。合约内可以编写特定的函数来实现所有权的转移或继承。合约的拥有者也有权销毁合约；销毁后，合约地址上的剩余以太币将被发送到指定的地址，同时合约账户上的所有状态也会被清除。需要注意的是，尽管合约本身被销毁，其代码仍然保留在区块链的历史记录中，不会被移除。因此，虽然历史记录不受影响，但一旦合约被销毁，其所有功能将无法调用，任何后续向该合约地址转入的以太币也将无法被转出。

1.4账户管理

在以太坊客户端中，负责管理账户信息的模块能够处理多个钱包，而每个钱包又可以管理多个账户。这里所说的“钱包”指的是管理账户的功能模块，它主要作为私钥的存放容器。

通常所指的钱包是一种具有用户界面的应用程序，用户可以通过它访问自己的资金、管理密钥和地址、创建和签署交易、与智能合约进行互动，以及作为去中心化应用（DApps）的接入点。这种钱包为用户提供了一个直观和便捷的方式来进行日常的区块链交互和管理任务。

1. 交易原理

交易的定义：在以太坊中，交易是一个被签名的数据包，主要用途包括从一个账户向另一个账户转移以太币、调用合约方法或部署新的合约。交易分为两大类：消息通信和合约创建，这两种类型的交易均由外部账户发起并提交到以太坊网络，之后它们通过网络传输，并最终在以太坊区块链上记录下来。

2.1创建交易

创建交易时，客户端需要填充以下详细信息：

nonce: 发送者账户的 nonce，表示该账户发出的交易总数。

GasPrice: 发送者愿意为每单位 Gas 支付的价格。

GasLimit: 发送者为这次交易设定的最大 Gas 消耗量。

to: 接收者的地址。在创建合约的交易中，此字段为空，采用默认的零地址。

value: 以 wei 为单位，从发送者向接收者转移的金额，或者是创建合约时赋予合约账户的初始金额。

v, r, s: 这些是交易签名组件，用于验证交易的发送者。

init: 仅在创建合约类型的交易中使用，用于初始化新合约账户的 EVM 代码片段。init 代码在首次执行时将返回一个永久与合约账户关联的代码体，执行完成后，init 值被丢弃。

data: 一个可以包含任意大小的字节数组，用作交易的附加信息，或者包含合约代码及其参数。

这些组件共同构成了一次交易，确保了交易的功能性和安全性。

2.2 交易签署

以太坊目前采用的签名方案是根据 EIP-155 规定的，该规定引入了交易重放保护机制。EIP-155 在交易数据结构中添加了三个字段：v, r, s，其中 r 和 s 的初始值为 0，v 的初始值为链标识符。这三个字段被添加到交易序列号之前，因此它们会影响交易的 hash 值。v 字段确保了链标识符的不可篡改性，这是因为如果链标识符被篡改，那么签名将变得无效。由于签名的有效性依赖于链标识符，这样就防止了交易在不同的链上被重放。

交易重放保护方案的内容是：在签名前的交易数据中添加了链标识符（chain-id）。确保了在一个链上（如以太坊主网）创建的交易不能再其他链上（如 ETC 或 Rposten 测试网）重放。

可用的链标识符如下表：

区块链名称 链标识符

ETH 主网 1

Expanse 2

Rposten 3

Rinkeby 4

Rootstock 主网 30

Rootstock 测试 31

Kovan 42

ETC 主网 61

ETC 测试 62

Geth private 测试 1337

2.3 交易广播

以太坊利用 P2P 网络架构，其中网络中的任意节点均可创建、发送、接收和验证交易。交易的广播过程始于产生该交易的节点，随后交易信息会被传递至相邻的节点，逐步扩散至整个网络。这种方式确保了交易能够迅速且广泛地在网络中被传播和确认。

2.4 交易提交

当一个节点在以太坊网络中接收到新的交易后，它首先会将这个交易存放到交易内存池中。交易内存池包含两个主要的列表：queue 列表和 pending 列表。新收到的交易最初被放入 queue 列表，随后根据一定的条件（如 gas 价格和 nonce 等）被转移到 pending 列表中，等待进一步的处理和确认。

2.5 交易执行

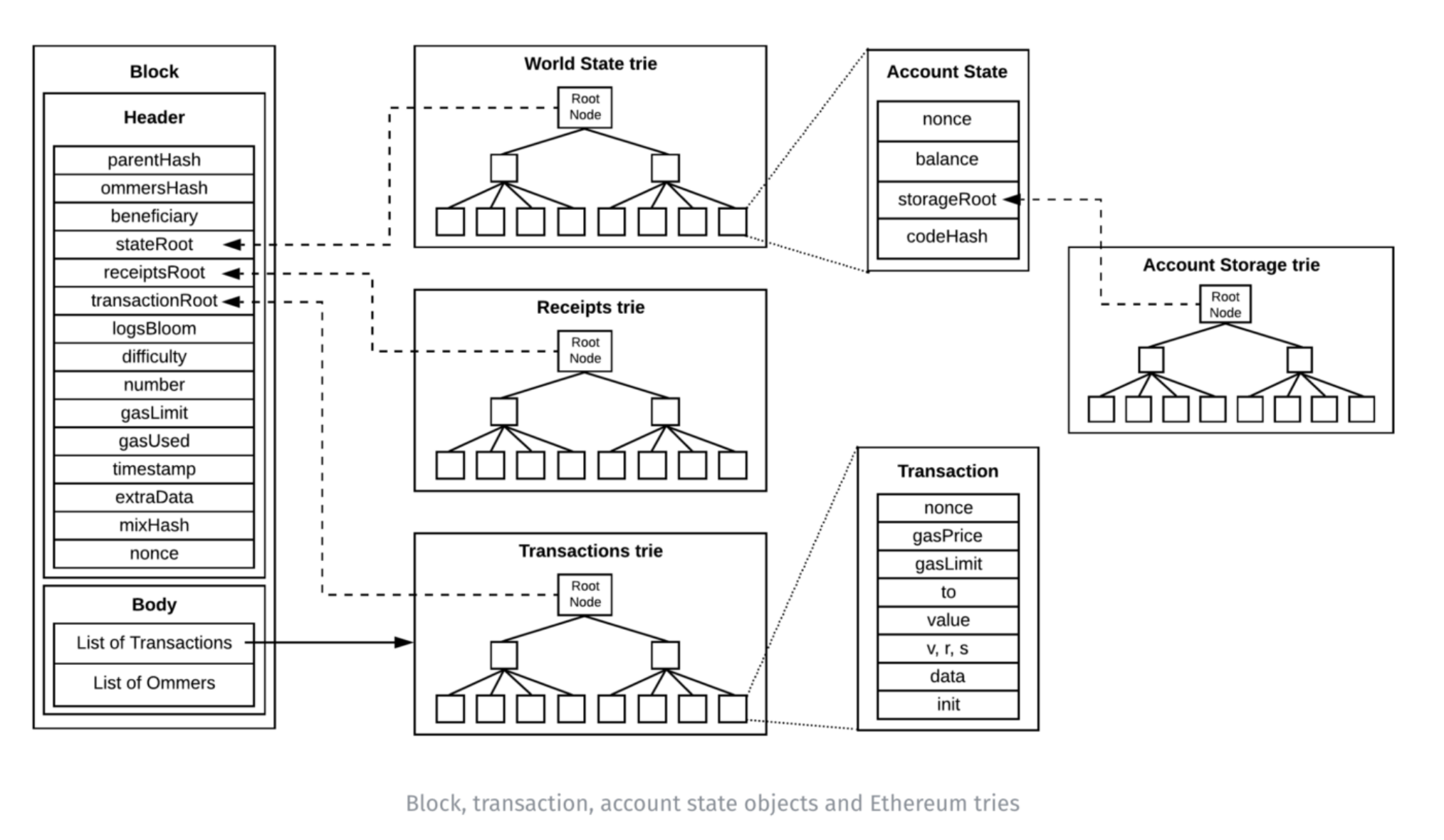
当交易被添加到 pending 列表中，它们成为待处理的交易，准备执行。这一执行过程是由矿工在挖掘新区块时进行的。矿工从 pending 列表中选取交易，交由以太坊虚拟机（EVM）执行，并将其打包进新区块中。重要的是，矿工在打包交易时不会等待执行结果，这样做可以加快整个区块的处理速度。

2.6 交易存储

交易执行后会生成一个收据，该收据记录了交易对账户状态所做的更改。这个收据只有在交易所在的区块被正式添加到区块链上之后，才会被提交并存储到底层数据库中。这标志着我们的交易已经完成了从创建、签署、传播到最终上链的整个过程。

1. 区块从创建到入库

3.1区块结构



区块头字段详解（注意：ommer 和 uncle 为同义词）：

parentHash：前一个区块（父区块）的 256 位哈希值。

ommersHash：与该区块关联的 uncle 区块的 256 位哈希值。

beneficiary：接收交易费的矿工的以太坊地址。

stateRoot：在区块中所有交易执行完成且区块被确认后，状态树根节点的 256 位哈希值。

receiptRoot：区块中所有交易收据所组成的树的根节点的 256 位哈希值。

transactionRoot：区块中所有交易所构成的树的根节点的 256 位哈希值。

logsBloom：区块中所有交易收据的日志所组成的 Bloom 过滤器。

difficulty：区块的难度值，根据父区块的难度值和时间戳计算得到。

number：区块的高度，其中创世区块高度为 0。

GasLimit：区块中可用的最大 Gas 量，根据父区块的 GasUsed 与 GasLimit \* 2/3 计算得到。

GasUsed：区块中所有交易消耗的 Gas 总量。

timestamp：区块生成的时间戳。

extraData：区块的附加数据，最长 32 字节。

mixHash：一个 256 位哈希值，与 nonce 共同用于证明当前区块执行了足够的计算量。

nonce：工作量证明过程中生成的一个 64 位哈希值，与 mixHash 一起用于证明当前区块的计算量。

【区块体】

区块体主要包括交易列表和引用的 uncle 区块列表。交易列表由交易池中选定并验证的交易组成，这些交易最终被包括在区块中。叔区块是区块体的一个特殊组成部分，从业务角度看，它不是必须的，但其存在会增加计算区块哈希的时间。

引用 uncle 区块的主要目的是增强主链的安全性，并减少拥有较大计算力的节点对区块链的潜在影响，从而降低安全风险。通过为 uncle 区块的打包者提供奖励，以太坊网络鼓励节点间计算资源的均匀分布，防止这些节点破坏区块链去中心化的基本原则。

3.2 区块生成

在以太坊网络中，除了初始的创世区块，所有其他区块都是由矿工创建并添加到网络的。区块的创建过程涉及矿工从交易池中按照特定的规则选择一系列经过验证的交易来进行打包，同时生成区块头和区块体。

在这个过程中，矿工在网络里扮演着接收、转发、验证及打包交易的关键角色。在以太坊 1.0 的时代，网络仍采用工作量证明（PoW）算法来竞争记账权。在以太坊中，这一竞争过程被称为区块的封印（Seal）。

3.3 区块封印

所谓的挖矿过程实际上涉及区块的创建和封印两个阶段。在第一阶段，矿工首先创建出一个完整的新区块，这包括填充区块头的部分属性、编排交易列表、添加 uncle 区块等。此时，所有交易已经执行，所有交易的收据已经收集，且相关账户状态也已更新。接下来，只需等待第二阶段的封印结果，才能完成区块头中剩余部分属性的填充，如难度（Difficulty）、随机数（Nonce）和混合哈希（MixHash）等。

一旦封印完成，区块便可以被广播到其他节点。需要注意的是，封印阶段所消耗的计算资源远超过区块创建阶段。

区块的封印是通过共识算法来完成的。目前，以太坊使用的共识算法包括 Ethash（以太坊主网采用）和 Clique（测试网如 ganache 采用）。在以太坊发展的第四阶段，系统将从工作量证明（PoW）转向权益证明（PoS），即采用基于 PoW 的 Casper 算法。封印过程不会改变区块中任何与有效数据相关的部分，其主要目的是通过一系列复杂的步骤来选拔具备本轮记账资格的矿工。

3.4 区块验证

在区块被封印并广播至其他节点后，这些节点负责验证区块是否符合上链条件。区块验证过程主要包括四个关键步骤：

验证区块头和 uncle 区块：检查区块头的完整性和正确性，以及引用的 uncle 区块是否合规。

验证交易：确保区块中包含的所有交易都是有效的，包括交易的结构和签名。

发放奖励：验证矿工和相关参与者（如 uncle 区块的矿工）的奖励是否正确计算并分配。

验证全局状态和工作量：检查区块所述的状态变化是否与执行结果一致，以及工作量证明是否有效。

这个全面的验证流程确保每个新增的区块都严格遵循以太坊网络的规则，维护了区块链的安全和一致性。

3.5 选择主链

在以太坊中，新区块在某个节点通过验证后，还需确认该区块所在的链是否为主链。鉴于以太坊中区块的生成速度较快，常常同时产生多个区块，形成一种区块树的结构。在这种情况下，主链定义为这棵区块树中从根节点到叶子节点最长路径上的链。

为了达成网络的共识，通常选择总难度最大的路径作为主链。主链的识别最直观的方法是比较从根节点到叶子节点的路径长度，即非创世区块的数量。路径越长，意味着该路径上累计的挖矿工作量越大。

在路径长度相等的情况下，需要比较两个叶子节点路径上区块的总难度。每个新区块的总难度由该区块自身的难度值加上其父区块的难度值组成。由于区块头包含了难度值信息，因此可以仅通过区块头来比较各个路径已完成的计算量。

3.6 区块存储

在新区块确认加入主链后，区块中的数据才会被以键值对形式存入节点本地的LevelDB中。存入过程：

handleNumberToKey(HeightNumber) -> BlockHash；（区块高度和hash的映射）

handleNumberAndHashToHeaderKey(num, hash) -> Header{各种hash，difficulty，Nonce，mixDigest，Number，GasUsed，GasLimit} （区块高度和区块hash组成的key和 区块头的映射）

handleNumberAndHashToBodyKey(num, hash) -> Body{txs, Withdrawals, Uncles[]}（区块高度和区块hash组成的key和 区块体的映射）

并将区块中所有收据存储为单个收据数组，用于在区块链重组期间重新安排丢弃的交易。经过以上流程，一个个区块被添加到区块链上，并存入LevelDB数据库中，形成以太坊的这条区块链。

从数据层的角度详细介绍了以太坊中的各个关键过程：从账户发送交易，到交易组成区块，区块构成区块链，以及区块数据最终被存入数据库的整个流程。

1. 网络层

. Kad 网络协议

KAD（Kademlia）是一个基于 DHT（Distributed Hash Table，分布式哈希表）的 P2P（点对点）网络协议。Kademlia 协议通过使用一种高效的算法来实现去中心化网络中的节点查找和资源定位，广泛应用于文件共享网络和分布式存储系统中。

Kademlia 协议的核心概念

1. 节点标识和键空间

节点 ID: 在 Kademlia 网络中，每个节点都有一个唯一的节点 ID，通常通过哈希函数生成。例如，节点 ID 可能是一个 SHA-1 哈希值。

键空间: 节点 ID 和数据键都属于相同的键空间。键空间是一个由 m 位二进制数字构成的空间，常用的长度是 160 位。

2. 距离度量

异或距离: Kademlia 采用异或运算来测量两个节点 ID 之间的距离。距离定义为两个 ID 的异或结果，并且这个结果被视为一个整数。

distance(A𝐴,𝐵B)=𝐴A⊕B𝐵distance(A,B)=A\_⊕\_B

这种度量方式具有对称性和满足三角不等式，这对于网络的高效路由是有益的。

3. 路由表

k-桶: 每个节点维护一个包含其他节点信息的路由表，这个表根据节点间的距离被分成多个桶（称为 k-桶）。第 i 个桶包含距离在 [2𝑖,2𝑖+1)[2\_i\_,2\_i\_+1) 范围内的节点信息，每个桶通常容纳最多 k 个节点（标准值为 20）。

节点选择: 路由表优先选择存活时间长且响应迅速的节点加入 k-桶。失效的或响应慢的节点将被新的节点替换，以保持网络的稳定性和效率。

Kademlia 的操作

节点查找

查找过程： 为了定位目标节点 ID，发起节点会根据其路由表中记录的信息，选择距离目标 ID 最近的 k 个节点并向它们发出查询请求。这些节点将继续检查自己的路由表，进而查询距离目标还更近的节点，直到找到目标节点或无法找到更近的节点为止。

数据存储

存储键生成： 数据（例如文件）的存储键通常由数据内容的哈希值生成，确保数据的唯一标识和检索。

存储操作： 存储数据时，会选取距离存储键最近的 k 个节点，并将数据的副本存储在这些节点上。发起节点通过查找与数据键距离最近的 k 个节点，然后向它们发送数据存储请求。

数据查找

查找过程： 查找数据的过程与查找节点类似。发起节点会基于数据键进行搜索，目标是找到实际存储该数据的节点。

查询操作： 在查询数据时，发起节点首先找到与数据键距离最近的 k 个节点，然后请求这些节点返回所需的数据或指向更接近目标数据的节点的信息。

这些步骤确保了 Kademlia 网络中的高效数据存储和检索，同时提高了网络的可靠性和数据的可访问性。

Kademlia 的优点

高效路由：Kademlia 的查找操作通常在 O(log n)的时间复杂度内完成，其中 n 为网络中的节点数。

容错性和动态性：网络中的节点可以随时加入或离开，Kademlia 通过维护和更新路由表，保证了网络的鲁棒性。

去中心化：没有中心服务器，所有节点都是对等的，增强了网络的去中心化特性。

应用场景

Kademlia 协议因其高效性、容错性和去中心化特性，被广泛用于各种去中心化的文件共享和分布式存储系统中。以下是一些主要的应用示例：

BitTorrent: 利用分布式哈希表（DHT）实现种子查找和文件共享。

IPFS (InterPlanetary File System): 一个分布式文件存储系统，使用 DHT 进行数据的定位和检索。

以太坊 Swarm: 一个去中心化存储平台，主要用于分布式应用（DApp）的数据存储和分发。

这些应用展示了 Kademlia 协议在支持去中心化网络和 P2P 应用中的关键作用。

节点发现流程

系统初始化: 系统启动后，随机生成本机的 NodeID，即 LocalID，并在整个生命周期中保持不变。

连接公共节点: 系统将读取公共节点信息，并通过 PING 和 PONG 命令验证这些节点的活跃状态。一旦验证通过，这些公共节点信息被写入到本机的 K 桶中。

定期刷新: 系统每隔 7200 毫秒执行一次 K 桶的刷新流程，以保持网络的活跃和更新。

以太坊与传统 Kad 网络的区别

与传统的 Kad 网络不同，其中节点发现主要是为了寻找一个固定的目标节点，以太坊的节点发现过程旨在尽可能地发现多个目标节点，而不是寻找特定的固定节点。因此，在以太坊的节点发现过程中，每一轮的目标节点都是新随机生成的，进行 8 次循环，以增强网络的连通性和鲁棒性。

1. 共识层

基于Poa的Clique算法

Clique 大致原理

Clique 是一种基于权威证明（Proof of Authority, PoA）的共识算法，专为需要更高效率和控制力的企业和联盟链设计。Clique 通过预先选定的验证者节点来验证区块和达成共识，大大提高了网络的效率和吞吐量。以下是 Clique 算法的大致原理：

基本原理

验证者节点

权威节点（验证者）： Clique 网络中的区块由一组预先选定的权威节点（validators）创建和验证。这些节点的身份是已知且可信的，通常由网络管理者选定。

有限的节点数量： 与开放的公有链不同，Clique 中的验证者数量是有限的，这有助于提高共识的效率和速度。

区块创建

轮值机制： Clique 使用轮值机制决定哪个验证者节点将创建下一个区块。在每个轮次中，一个预定的验证者负责创建区块，按顺序轮换。

创建间隔： 每个验证者在指定的时间间隔内创建区块，这个间隔通常较短（例如 15 秒），以提高交易处理速度。

区块签名

区块签名： 验证者创建区块后，会使用其私钥对区块进行签名。这个签名是区块有效性的证明，其他节点可以通过验证这个签名来确认区块是由合法的验证者创建的。

共识签名： 为了确认一个区块的有效性，需要得到一定比例的验证者签名。例如，如果有 N 个验证者，可能需要至少 N/2 + 1 个验证者的签名才能认为区块有效。

容错机制

双区块保护： 为了防止恶意行为和双花攻击，Clique 引入了一个限制，即验证者在创建一个新的区块之前，必须等待其上一个区块被其他验证者创建的区块所覆盖。这种机制确保了网络的安全性和一致性。

踢出机制： 如果某个验证者被发现行为不端或长时间不活跃，网络管理者可以将其从验证者列表中移除。

工作流程

区块生成

每个验证者按顺序轮值生成区块。

验证者在自己的轮次上创建区块，并对其签名。

其他验证者接收到区块后，验证签名并签署共识。

共识达成

一个区块必须得到大多数验证者的签名（通常是 N/2 + 1），才能被认为是有效的。

达成共识后，区块被添加到区块链上。

区块传播

验证者将已签名的区块广播给网络中的其他节点。

其他节点在接收到区块后进行验证，并更新各自的区块链副本。

优缺点

优点

高效率： 由于只有少数验证者参与共识，Clique 能够实现较高的交易处理速度和较短的区块生成时间。

低能耗： 与工作量证明（PoW）不同，Clique 不需要大量计算资源来进行区块验证，因此能耗较低。

易于管理： 验证者的身份是已知且受信任的，这使得网络管理和治理更加简单。

缺点

中心化风险： 由于验证者是预先选定的，Clique 在一定程度上依赖于这些验证者的诚信，存在一定的中心化风险。

有限的去中心化： 适用于需要控制参与者的私有链或联盟链，不适用于完全去中心化的公有链。

1.3 应用场景

Clique 算法特别适用于以下场景：

企业区块链： 适合需要高效处理交易且参与方有限的企业级应用。

联盟链： 适用于多个组织共同运营的区块链网络，参与方都是经过认证的。

Clique 通过预先选定的验证者节点来提高网络效率，同时保留了一定的去中心化特性，是一种适合企业和联盟链使用的高效共识算法。

1. 激励层

为了保证系统的长期健康发展，必须有完善的奖惩机制。以下是以太币的发行机制，包括四种形式：初始发行、普通区块奖励、uncle 区块奖励和 uncle 区块引用奖励。

1. 初始发行

以太币在初始发行时，价格锚定比特币，以 1 BTC = 1337~2000 ETH 的价格进行发售。2014 年 7 月至 8 月期间，通过众筹发行了约 7200 万以太币，这些以太币全部发往众筹投资人的以太坊账户地址中。

2. 普通区块奖励

这是奖励给挖出区块的矿工的，具体规则如下：

固定奖励 3 ETH（拜占庭分叉后，将原来的 5 ETH 奖励降为 3 ETH）。

区块内所有交易的交易费。

如果普通区块引用了 uncle 区块，每引用一个 uncle 区块可以得到固定奖励 3 ETH 的 1/32，最多只能引用 2 个 uncle 区块。

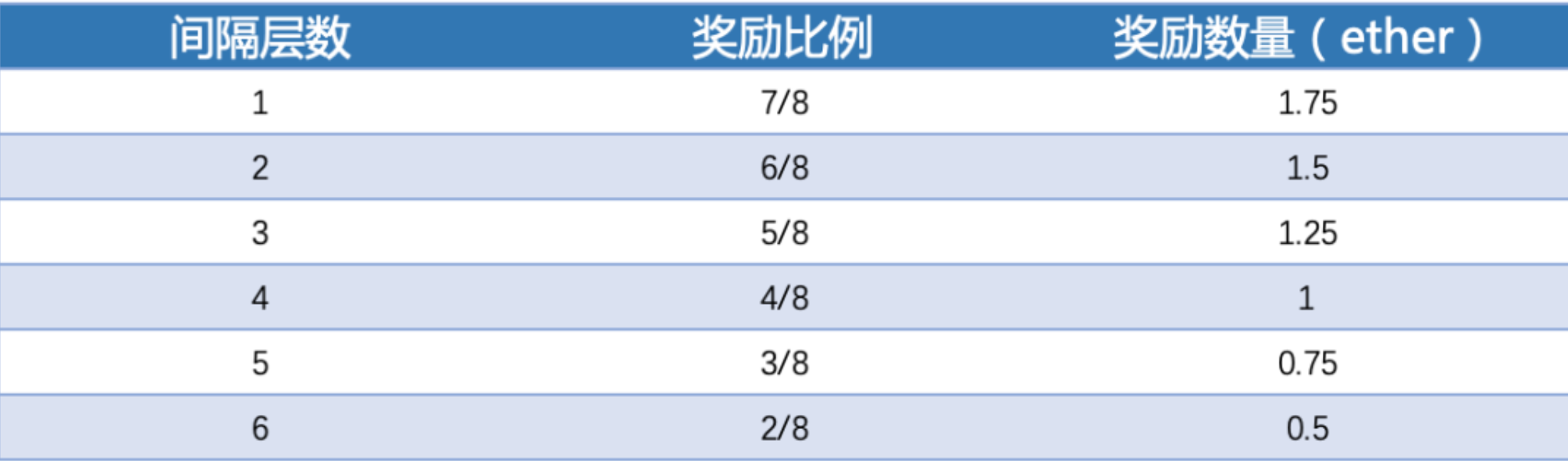
3. uncle 区块奖励

这是给挖出此 uncle 区块矿工的奖励。

uncle 区块奖励=（uncle 区块高度 + 8 - 引用 uncle 区块的普通区块高度）x 普通区块的固定奖励/8

4. uncle 区块被引用奖励

当 uncle 区块被引用时，挖出 uncle 区块矿工收到此奖励，奖励规则如下：



其中间隔层数指的是被引用的 uncle 区块与普通区块之间的高度差，越小则奖励越多。

1. 合约层

智能合约可以看作是将现实中的合约条款用代码形式描述，并记录到区块链中，然后由以太坊虚拟机（EVM）自动执行条款。智能合约从创建到执行需要以下几个步骤：

1、编写合约：使用智能合约语言（如 Solidity）编写合约逻辑。

2、编译合约：使用编译器编译合约，生成 ABI 接口和字节码。

3、部署合约：使用合约部署工具将合约部署到以太坊网络上。

4、执行合约：合约代码被载入 EVM 中，解释为机器语言并执行，执行完后会改变相应账户的状态。

5、调用合约：使用合约交互平台调用合约功能，由 EVM 负责执行。

6、销毁合约：调用合约的自毁函数，清空该合约账户的状态。

通过以上步骤，智能合约实现了从编写到部署、执行再到销毁的完整生命周期，为以太坊平台上的应用提供了强大的支持。

1. 创建合约

当一笔交易的 to 字段为空时，表明该交易是一个合约创建交易，而交易的 data 字段则包含了合约代码的字节码。创建合约的步骤如下：

1、编写合约代码：使用合约语言编写合约代码。目前，以太坊合约语言包括 Solidity、Vyper、Serpent 和 LLL，其中 Solidity 最为常用。

2、编译合约代码：使用编译器将合约代码编译成字节码，并生成 ABI 接口。

3、创建交易：将编译后的字节码附在交易的 data 字段中，并将 to 字段留空。

4、提交交易：将交易提交到以太坊网络，EVM 会读取交易并创建合约。

通过这些步骤，EVM 将解析交易的字节码并创建新的智能合约，完成合约的部署。

2. 执行合约

在执行合约之前，EVM 会首先判断该合约是否为预编译合约。如果是预编译合约，则直接执行；如果不是，则将合约的字节码交给编译器解析成机器码后再执行。

3. 合约调用

有两种类型：合约调用函数、合约调用合约。

3.1 调用合约函数

在调用合约函数时，合约字节码会携带一个 Input 参数进入 EVM 解释器。Input 数据包含两部分：前 4 字节是函数签名的 Keccak 哈希值的前 4 字节，作为该函数的唯一标识，解释器通过这 4 字节知道要调用哪个函数。后面部分是该函数需要的参数，长度不定。

具体流程如下：

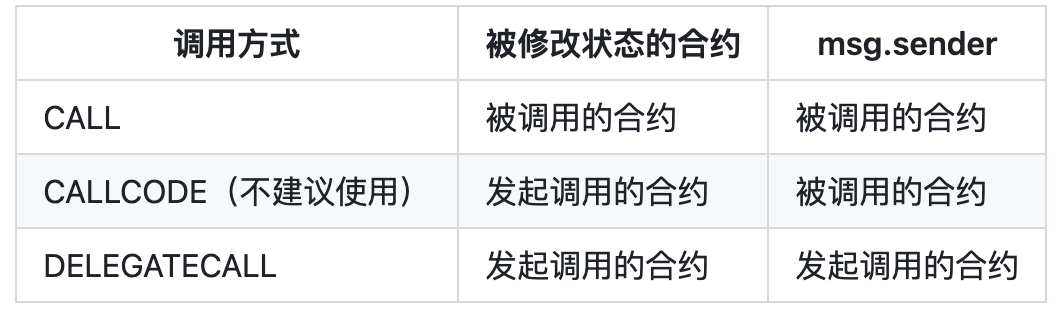
解析函数签名： 通过 CALLDATALOAD 指令将前 4 字节入栈。

匹配函数： 将这 4 字节依次与合约中的函数签名进行对比，匹配到的函数被调用。

执行函数： 通过 JUMPI 指令跳转到匹配的函数代码，继续执行。后续的字节作为参数传递给该函数。

3.2 合约调用合约

这种类型有四种调用方式：CALL、CALLCODE、DELEGATECALL、STATICALL。其中 STATICALL 表示不允许读取和修改状态变量的调用类型，暂未使用，在此不讨论。 三种调用方式比较如下图：



其中，DELEGATECALL 是 CALLCODE 的升级版本。使用 DELEGATECALL 的目的是使合约在不传输自身状态的情况下，能够使用其他合约的代码，这使得合约可以实现“库”的功能，即将可复用的代码放在一个合约中，当被其他合约委托调用时，只使用库的代码而不会更改库中变量的状态。

以下是合约通过 CALL 指令调用其他合约的流程：

1、准备调用参数：调用者将调用参数存储在内存中，然后执行 CALL 指令。

2、执行 CALL 指令：CALL 指令执行时会创建一个新的合约对象，并将内存中的调用参数作为其输入。

3、创建执行环境：解释器为新合约的执行创建新的栈和内存空间，不会破坏原合约的执行环境。

4、返回结果：新合约执行完成后，通过 RETURN 指令将执行结果写入原合约指定的内存地址，然后原合约继续执行。

这种机制确保了调用的隔离性和安全性，使得合约可以安全高效地利用其他合约的代码。

1. Gas和Gas优化

合约的部署和调用都会消耗一定的 Gas 数量，它作为矿工提供计算和存储资源的报酬，也是为了防止以太坊网络遭到 DoS 攻击。

4.1 关于 Gas

Gas（译作汽油）可以形象地比作维持以太坊平台持续运行的燃料。当用户发起交易时，需要支付一定数量的 Gas。例如，为一笔普通转账交易支付 21000 Gas。

此外，由于交易可以通过 Input Data 字段携带合约相关的字节码数据以实现部署合约或调用合约函数的目的，而这些字节码数据的计算、带宽和存储会消耗节点资源，因此用户还需要根据所使用的资源量进行付费。这确保了以太坊网络的资源得到合理分配和使用。

4.2 Gas 计价规则

首先，Gas 计费有几个基本原则：

先扣费，后执行： 每个操作码在执行前先扣费，再执行。这是因为部分操作的 Gas 消耗无法提前得知，因此判断粒度是操作码而不是整个合约。

提前中止不退费： 如果执行过程中用户预设的 Gas 数量不足以支付操作码执行所需的费用，合约执行会中止，且已扣除的 Gas 不会退还。

释放资源可退费： 在执行过程中，如果某些操作码释放了存储资源，例如重置某个状态变量或销毁合约，则会在当次调用结束后退还部分 Gas，但退还量不超过已使用 Gas 的一半。

【GasLimit、GasPrice 和 GasUsed】

在以太坊中，有三个重要的 Gas 相关概念：

GasLimit：用户在发起交易时设置，表示本次交易可消耗的最大 Gas 数量。

GasPrice（有时也称为 GasFees）：用户在发起交易时设置，表示本次交易每单位 Gas 的价格，计价单位是 ETH。由于合约消耗的费用通常不会达到几个 ETH，因此类似于人民币的元分角，ETH 也有更小的计价单位。常用单位是 Gwei，1 ETH = 10^9 Gwei。

GasUsed：实际交易消耗的 Gas 数量，不超过 GasLimit。在区块链浏览器的单个交易详情页，通常可以看到此字段的值旁边有个百分比，比如 GasUsed By Tx: 24876 (41.46%)，这个 41.46% 是 GasUsed/GasLimit，此时页面上显示的 GasLimit 为 60000。

总的交易开销（transaction cost）计算公式为：



【BaseFees 和 PriorityFees】

在 2021 年 8 月 5 日，以太坊完成了伦敦升级，改进的 EIP-1559 用两个新参数（BaseFees 和 PriorityFees）替换了旧的 GasPrice 参数：

BaseFees：在以太坊上发送交易或完成操作所需的最低 Gas 价格。基本费由协议动态调整，并且这些基本费会被销毁，不会支付给矿工。

PriorityFees（Tip）：优先级费用（也叫小费），用户支付给矿工的小费，以便矿工优先处理自己的交易或操作。用户可以自行设置这个费用，默认情况下，小费预计为 2 Gwei。

MaxFees：等于 BaseFees 加上 PriorityFees，相当于之前的 GasPrice，但增加了一个“最高”的定语。实际用到的小费可能不会达到预设的 PriorityFees 这么高，剩余部分会退还。

更新后的总交易开销（transaction cost）计算公式为：



EIP-1559 由以太坊核心开发者在 2019 年 4 月正式提出，目的是优化以太坊网络的 Gas 收费市场，能够一定程度上防止针对以太坊网络的垃圾交易攻击。 它最大的特点是提出了费用销毁机制（BaseFees 不会支付给矿工），减少了攻击动机，提供以太坊安全性。关于更多详细请自行查询。

【GasPrice 预估服务】

有多个网站和工具可以提供 Gas 价格预测的数据服务，例如：

Blocknative：浏览器插件，提供实时的 Gas 价格预测。

ETH Gas Station：网页端工具，提供详细的 Gas 价格数据和预测。

Etherscan：以太坊区块链浏览器，提供 Gas 价格预测和其他区块链数据。

OKLink：区块链数据平台，提供 Gas 价格预测和分析。

这些平台还提供 API 服务，用户可以集成到自己的应用中。不过，各自提供的数据准确度可能有所不同，用户需要在使用后自行评估其可靠性。

注意：预估时会涉及到传统 GasPrice 和 EIP-1559 版本的 GasPrice。对于传统 GasPrice，它是一个字段，而 EIP-1559 是 BaseFees（区块链网络决定，不可修改）和 PriorityFees 两个字段。

浏览器插件在体验上就更具优势，可以在交易时快速输入几个 Gas 字段（如 PriorityFees、GasLimit）。

4.3 区块 Gas 限制

区块 Gas 限制是指一个区块中所有交易总共能消耗的最大 Gas 数量，因此也限制了一个区块能包含的交易数量。

由于矿工会优先打包支付更高 PriorityFee 的交易（包括执行附带的合约字节码），区块 Gas 限制直接影响了交易的打包顺序和速度。区块 Gas 限制是由矿工通过投票定期调整的，系统根据当时的网络需求决定最终结果。

4.4 Gas 优化

目前，一个中大型合约的部署成本可能达到几百上千美元，这里选取主网一笔创建合约的交易来计算，交易链接 。

https://etherscan.io/tx/0xeed6c791744af81bde027c6bcb2ac927b6d7964535edfc434c060fec7d24de2b

交易总费用的美元价值=GasPrice \* GasUsed \* ETH市价= 86.74Gwei \* 3533714 \* 1e-9 Ether \* $3829.61 = $1173.83

其中 1e-9 是把 Gwei 转 Ether 单位，$3829.61 是交易发生时 1ETH 的美元价格。由从可知，创建一个合约的成本不可谓不高！

所以优化合约代码以减少 Gas 费用就成为了一个非常重要的主题。

七、应用层

以太坊应用层是区块链的展示层和业务层，通过调用 web3 等接口，使各类应用与区块链集成。总体上，应用层包括两类应用：

1、工具类应用：这些应用本身并不使用区块链技术，但调用了区块链的接口，为用户提供图形化界面，方便使用区块链服务。例如：

轻节点钱包：提供用户友好的界面来管理钱包和进行交易。

区块链浏览器：用于查看区块、交易和账户信息。

中心化交易所：通过区块链接口实现交易，但本身是中心化的。

2、区块链技术应用：在以太坊中，这类应用称为 DApps（去中心化应用），它们的功能全部或部分由区块链技术实现，能够真正发挥区块链的优势。例如：

去中心化金融（DeFi）应用：如去中心化交易所（DEX）、借贷平台等。

去中心化自治组织（DAO）：通过智能合约实现的自治组织。

游戏和娱乐应用：如区块链游戏、数字收藏品等。

1. 钱包

钱包只包含密钥，通过密钥来控制以太坊区块链中的代币。用户通过钱包密钥来签署交易，以证明他们拥有代币，从而控制以太坊网络上的代币。因此，钱包可以看作是一个包含公私钥的钥匙串。

以太坊客户端实现了软件钱包和硬件钱包。而软件钱包又分为两种：非确定性钱包和确定性钱包，它们通过所包含的密钥是否彼此相关来区分。

非确定性钱包：其中每个密钥都是随机生成的，密钥之间没有关系。

确定性钱包：所有密钥都来自一个叫做种子（seed）的主密钥，其中包含的所有密钥都是互相关联的。只要主密钥还在，就可以重新生成其他子密钥。

在确定性钱包中使用了许多不同的密钥派生方法，常用的是树状结构，由此生成的钱包称为分层确定性钱包或 HD 钱包。为了使种子更容易被记住，通常将其编码为一些英文单词，也叫做助记词。

2. DApp

DApp（去中心化应用）是一个基于开放的、去中心化的 P2P 基础架构服务的 web 应用程序。它可以是一个网站或一个手机应用，只要包含智能合约和 web 用户界面，就可以称为一个 DApp。

2.1 优势

DApp 比智能合约更能体现以太坊作为通用区块链平台的功能。每一个 DApp 都运行在基于以太坊的节点网络上，具有以下几个优点：

1、永不宕机：依赖区块链系统运行，不会因为单点故障而停机。

2、去中心化：DApp 的数据分布式存储在区块链的所有节点上，避免数据集中化。

3、公开透明：DApp 的代码是开源的，由所有人或组织共同维护和监督。修改代码需要大多数用户达成共识，不能由单个人或机构私自修改。

4、激励机制：为了维持 DApp 的长久运行，需要设计激励机制来奖励消耗算力、内存空间等资源的用户，从而确保 DApp 在各个用户节点上正常运行。

架构

DApp 的架构通常有三种模式：

间接模式：Web 网页客户端与中间服务器交互，中间服务器再与区块链交互。

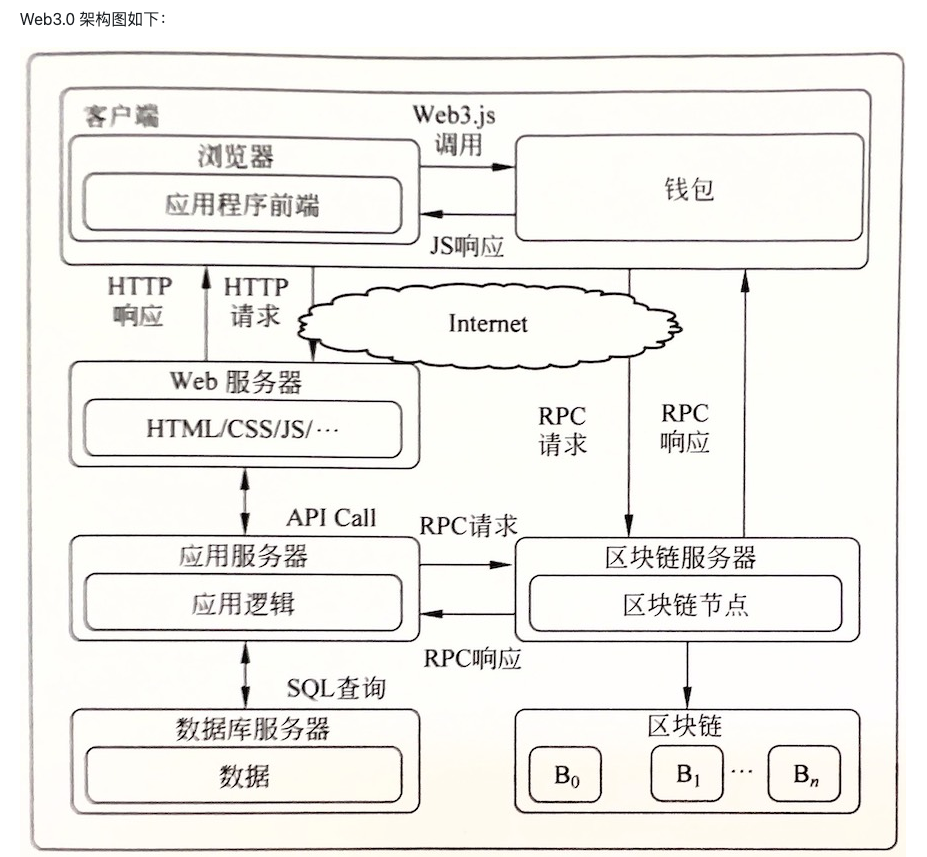
混合模式：链上数据直接与区块链交互，其他内容与中间服务器交互。

直接模式：DApp 直接与区块链交互。

DApp 的实现与传统 Web App 对比如下表：



架构图：



从技术架构来看，Web 3.0 更加复杂，它在传统信息互联网的基础上增加了价值的传递。Web 3.0 引入 Web3.js，通过远程过程调用（RPC）从远程计算机上请求服务器来与区块链节点通信，并且需要借助钱包来连接前端的 DApp 和底层区块链，从而实现信息和价值在整个网络中的传递。

从整个流程来看，底层逻辑很复杂，但对于普通用户来说，使用 DApp 浏览器访问 DApp 的方式与传统 App 无差别。用户接触的仍然是熟悉的前端页面。

可以说，DApp 浏览器是连接用户与 Web 3.0 世界的窗口，为用户提供端对端、匿名、安全和零信任的交互体验。

2.3 DApp 浏览器

目前绝大多数 DApp 仅支持 Web 端使用，这增加了用户的使用成本，不利于 DApp 的普及。在这种背景下，移动端的 DApp 浏览器应运而生。与传统浏览器不同，DApp 浏览器并不是一个独立入口，它需要结合或内嵌在数字钱包中才能使用。因为区块链带来的是价值互联网，现阶段的价值需要借助钱包来传递，钱包也是加密货币持有者的标配，借助钱包的流量入口优势，DApp 更容易触达用户。

常见的 DApp 浏览器包括 Mist、MetaMask、Coinbase Wallet、Trust Wallet、imToken、Status 和 Cipher 等。以太坊的 Mist 是最早的 DApp 浏览器，兼具钱包和 DApp 入口的功能，用户可以通过 Mist 与以太坊网络上的其他 DApp 交互。但 Mist 需要用户下载以太坊完整区块链数据才能运行。由于账本会随着时间越来越庞大，同步一个全节点数据需要非常长的时间和较大硬盘容量，这对普通用户非常不友好。

因此，后来出现的 DApp 浏览器一般通过 RPC 访问全节点来与区块链交互，无需自身作为全节点运行，这大大简化了用户的使用过程，提高了 DApp 的普及率和可用性。

以太坊新技术

Quorum 是基于以太坊的 Golang 版本（go-ethereum）开发而成的，因此也被称为 GoQuorum。它为金融服务行业提供了以太坊许可链方案，能够轻松支持交易与合约的隐私性。Quorum 的设计目标之一是尽可能地复用已有技术，最大限度地减少对现有以太坊协议的改动。

在 go-ethereum 的基础上，Quorum 进行了如下关键改动：

添加了交易与合约的隐私性：只有相关方才能查看交易细节，非相关方无法获取这些信息。

网络/节点的许可管理：只有经过授权的节点才能加入区块链网络。

调整了共识协议为 PoA 类协议：实现了 4 种 PoA 类共识协议，其中包括 IBFT、QBFT、Raft 和 Clique。这些协议不需要挖矿，因此具有远高于 PoW 类协议的交易吞吐量。

区块生成逻辑调整：将区块头中的全局状态根替换为全局公共状态根。

区块验证逻辑调整：将主要逻辑改为处理私有交易，并修改为检查全局公共状态根。

状态树分裂：将 State Patricia trie 分裂为公共和私有两棵树。

交易创建逻辑修改：允许使用加密哈希替换交易数据，以便在需要时保留私有数据。

取消交易费：在 Quorum 链中取消了交易产生的费用，但保留了 Gas 概念。

Quorum 的核心在于使用密码学技术防止交易方以外的人查看敏感数据。该方案结合了单独的共享区块链和智能合约框架与对以太坊原始代码的修改。智能合约框架用于隔离隐私数据，而对 Go-Ethereum 代码库的修改涉及区块提案和验证过程。区块验证过程通过执行交易合约代码进行，所有节点都会验证公开交易和与交易方相关的私有交易，而忽略其他私有交易。

2. 以太坊分片技术

以太坊 2.0 将以太坊网络分为两层：上层为现有的以太坊主链，基本保持不变；下层为分片链，主要处理和验证状态分区中的交易。为了进一步提高每个分片的共识速度，以太坊 2.0 将会将原有的 PoW 共识机制升级为 PoW&PoS 共识机制，即 Casper FFG 共识算法，未来可能升级为完全的 PoS 机制，即 Casper CBC 算法。

以太坊 2.0 的分片网络被分为主链和分片链，各自的职责如下：

主链：生成随机数、保存验证节点相关信息、追踪子链区块。

分片链：处理交易、存储账户与合约的状态。

可以简单地认为，分片中的交易都会被装入校对块。与侧链类似，校对块中只有一小部分被记录到主链。

在分片链上，交易处于自己的独立空间中，分片验证人只需要验证他们所关注的分片。分片链通过 PoS 机制依附于主链，以获得更高层次的共识。

具体过程是，分片中包含多个节点，即分片验证人，他们可以通过 PoS 机制在分片中完成交易验证。验证完成后，会生成一个 Collation，该 Collation 的头部信息被加入主链区块，而具体交易则不存入主链。

为了将分片链加入主链中，主链上提供了一个名为验证人管理员的智能合约，该合约是整个分片机制的核心。它提供了许多函数接口，包括权益证明系统、伪随机采样、Collation 头验证、跨分片通信、链上治理等功能。

3. 以太坊侧链技术

侧链是一种解决目前主要区块链项目（如比特币、以太坊等）的交易性能、隐私保护等方面问题的方案。

侧链本质上是一种跨区块链解决方案，通过这种方案，可以实现数字资产从第一个区块链转移到第二个区块链，又可以在稍后某个时间点转移回来。

其中第一个区块链叫做主链，第二个叫侧链。最初，主链通常是指比特币区块链，而现在可以是任何区块链。通过侧链技术，可以在主链基础上进行交易隐私保护技术、

智能合约等新功能的添加，这样可以让用户访问大量的新型服务，并且对现有主链的工作不造成任何影响。另外，侧链也提供一种更安全的协议升级方式，

当侧链发生灾难性问题时，主链依旧安然无恙。

侧链实现的技术基础是双向锚定（two-way peg），通过双向锚定技术，可以实现暂时的将数字资产在主链中锁定，同时将等价数字资产在侧链中释放。

同样，当等价数字资产在侧链中被锁定时，主链的数字资产也可以被释放。双向锚定实现的最大难点是协议改造需要兼容现有主链，也就是不能对现有主链的工作造成影响，

其具体实现方式可以分为单一托管模式、联盟模式、SPV 模式、驱动链模式和混合模式 5 类。