一、区块链网络层

1.1、什么是P2P技术

答：p2p是一种对等的网络节点技术。具有可扩展性、健壮性、负载均衡等特定。
比较流行的有三种结构：

- DHT结构（分布式哈希表），是一个环形的拓扑结构
- 树形结构
- 网状结构，没有稳定关系，提供了最大的容忍度和动态适应性，目前以太坊、比特币是这种结构

更多区块链资料详见主页： https://learnblockchain.cn/people/436

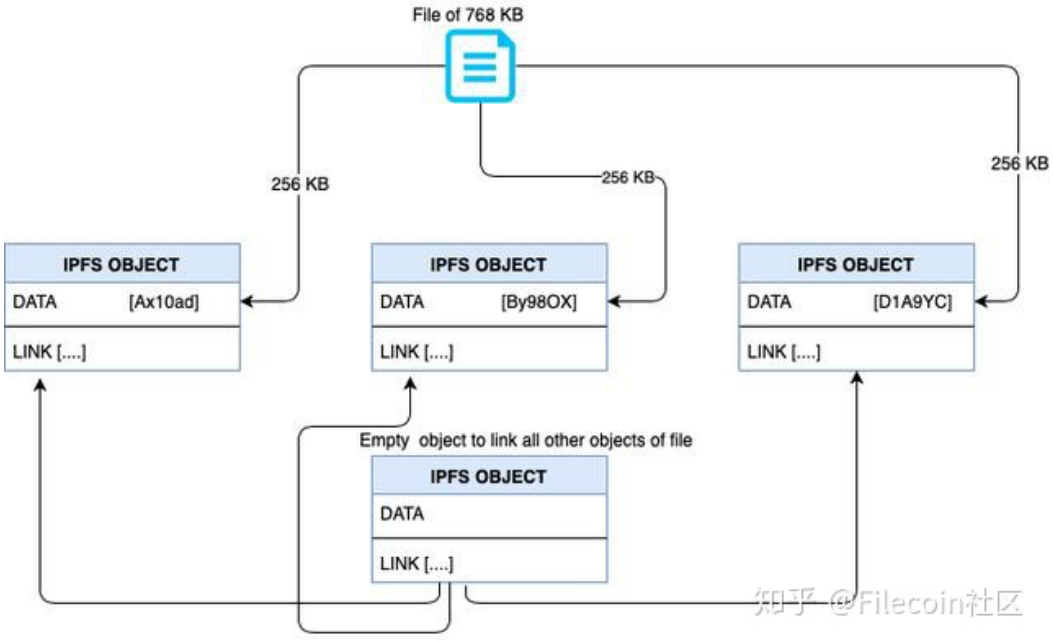
二、数据存储

2.1、IPFS如何存储文件

答：IPFS的数据存储结构是一种Merkle DAG（有向无环图）。
数据存储到IPFS的流程为：

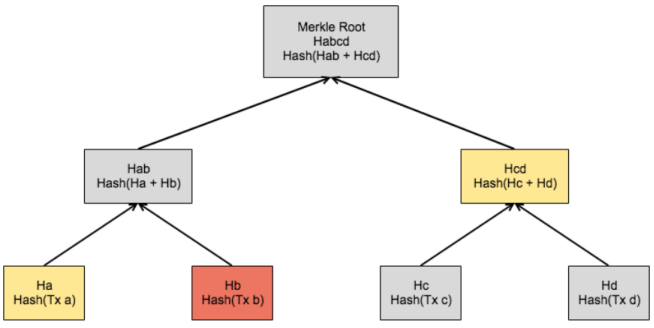
- 如果文件大于256kb，则将数据切分为多个块
- 系统为每一个块计算hash，并将所有hash拼凑成一个数组
- 计算数组hash，得到最终的文件hash

https://www.jianshu.com/p/f5a352be8c02



2.2、Merkle 树

答：比特币、以太坊等区块链数据结构都是采用Merkle树的数据结构。
树的叶子是数据库的hash值，非叶子节点对应子树hash串联字符串的hash值，结构如下图所示。



Makle Proof 是证明某个区块是否存在，是从树中的某个节点向上遍历，算出Merkle Root所需要经过的路径节点。如下图，计算Txb，只需要从全节点获取Txa和Hc+Hd两个节点的hash即可。

三、共识算法

共识解决的核心问题是分布式系统一致性问题。节点从相同的初始状态，开始接收相同的顺序指令，最终保持相同的结果。
在分布式系统中，通常把会出现故障但是不会伪造信息的情况称为“非拜占庭错误(Non-Byzantine Fault)”或“**故障错误**(Crash Fault)”；伪造信息的恶意响应情况成为“**拜占庭错误**”(Byzantine Fault)，对应节点为拜占庭节点。拜占庭错误场景中因为存在捣乱者更难达成共识。
根据解决的场景是否允许拜占庭错误情况，共识算法可以分为**崩溃容错CFT**(CrashFault Tolerance)和**拜占庭容错BFT**(Byzantine Fault Tolerance)两类。

3.1 崩溃容错CFT算法

CFT类容错算法通常性能比较好，处理较快，**容忍不超过一半的故障节点**。

3.1.1 Paxos

Paxos是一个两阶段提交算法，包含三种角色：**Proposer（提议者）**，**Acceptor（决策者）**，**Learner（决策学习者）**。
算法的流程为：
第一阶段：
A、Proposer生成一个提案，具有全局唯一编号N，然后向首页Acceptor发送Prepare请求。
B、Acceptor收到编号为N的提案后，如果N未处理过，且比自己之前处理的提案编号都要大，则向Proposer响应，并不再接受任何比N小的提案。
第二阶段：
A、Proposer收到半数以上的Acceptor的回应，那么就发送一个针对[N,V]提案的Accept请求给Acceptor。
B、Acceptor收到编号为N的提案的Accept请求后，只要Acceptor没有接收过比N大的提案，则接受改提案。
C、Acceptor接受提案后，有三种方式将提案发送给learner。第一种接收一个就发送给每一个Learner；第二种，接收后发送给主Learner；第三种，接收后发送给Learner集合。

Paxos算法只有两种情况下出现服务不可用：
1、超过半数的Proposer异常。通过增加节点个数来避免。
2、出现活锁。如果两个提案者恰好依次提出更新的提案，则导致活锁，系统会永远无法达成共识(实际发生概率很小)。活锁没有产生阻塞，但是一直无法达成一致。活锁有三种解决方案：
A、在被打回第一阶段再次发起PrepareRequest请求前加入随机等待时间。
B、设置一个超时时间，到达超时时间后，不再接收PrepareRequest请求。
C、在Proposer中选举出一个leader，通过leader统一发出PrepareRequest和AcceptRequest。

3.1.2 Raft

Raft算法是对Paxos算法的重新简化设计和实现，将一致性问题分解为**领导选举（leader election）**、**日志复制（log replication）**、**安全性（safety）**三部分。
Raft算法角色包括：

- 领导者(Leader)，集群中只有一个处于Leader状态的服务器，负责响应所有客户端的请求。
- 候选者(Candidate)，Follower状态服务器准备发起新的Leader选举前需要转换到的状态，是Follower和Leader的中间状态。
- 跟随者(Follower)，刚启动时所有节点为Follower状态，响应Leader的日志同步请求，响应Candidate的请求。

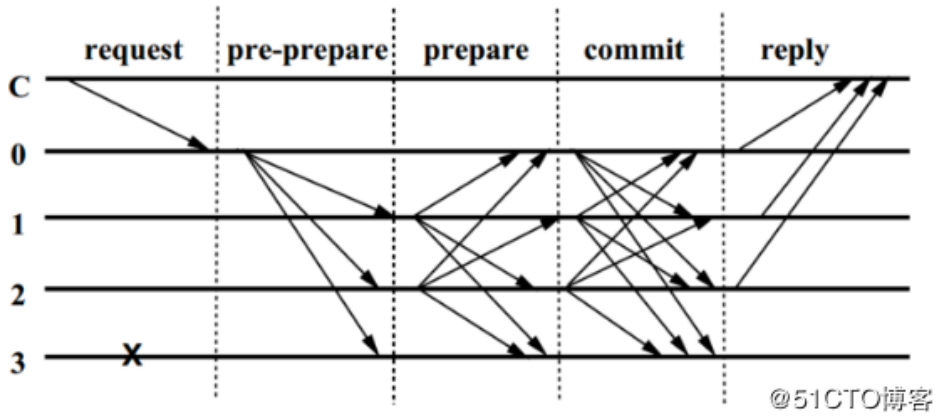
Raft算法在集群的初始情况下，都是Follower，Follower在规定时间内没有收到Leader的信息就会自动转成Candidate，并发起竞选Leader的投票，获得超过半数的支持则当选为Leader。其他未当选的候选者转换为Follower。
在一个任期（Term）中，没有或只有一个Leader。
Raft算法日志复制流程：
当Leader选举出来后就开始接受客户端的请求，所有请求都必须先经过Leader的处理。Leader接受到请求后，将其追加到日志（Log）的尾部，然后向其他Follower发出Append的请求，当大多数Follower复制完成后，Leader将请求应用到内部状态机，并将执行结果返回给客户端。

安全性
安全性是用来保证每个节点都执行相同序列的安全机制，如当某个Follower在当前Leader提交命令时不可用，稍后可能该Follower又会被选举为Leader，这时新Leader可能会用新的Log覆盖先前已提交的Log，这就是导致节点执行不同序列；安全性就是用于保证选举出来的Leader一定包含先前已经提交Log的机制。
为了达到安全性，Raft算法增加了两个约束条件：
A、要求只有其Log包含了所有已经提交的操作命令的那些服务器才有权被选为Leader。
B、对于新Leader来说，只有它自己已经提交过当前Term的操作命令才被认为是真正的提交。

3.2 拜占庭容错BFT算法

3.2.1 PBFT

确定性一致算法。一旦达成共识即不可逆转。容忍不超过1/3的故障节点。
PBFT（Practical Byzantine Fault Tolerance），即实用拜占庭容错算法。
PBFT算法实现流程如下：



首先，客户端向主节点发起请求，主节点 0 收到客户端请求，会向其它节点发送 pre-prepare 消息，其它节点就收到了pre-prepare 消息，就开始了这个核心三阶段共识过程了。
Pre-prepare 阶段：节点收到 pre-prepare 消息后，会有两种选择，一种是接受，一种是不接受。什么时候才不接受主节点发来的 pre-prepare 消息呢？一种典型的情况就是如果一个节点接受到了一条 pre-pre 消息，消息里的 v 和 n 在之前收到里的消息是曾经出现过的，但是 d 和 m 却和之前的消息不一致，或者请求编号不在高低水位之间（高低水位的概念在下文会进行解释），这时候就会拒绝请求。拒绝的逻辑就是主节点不会发送两条具有相同的 v 和 n ，但 d 和 m 却不同的消息。
Prepare 阶段：节点同意请求后会向其它节点发送 prepare 消息。这里要注意一点，同一时刻不是只有一个节点在进行这个过程，可能有 n 个节点也在进行这个过程。因此节点是有可能收到其它节点发送的 prepare 消息的。在一定时间范围内，如果收到超过 2f 个不同节点的 prepare 消息，就代表 prepare 阶段已经完成。
Commit 阶段：于是进入 commit 阶段。向其它节点广播 commit 消息，同理，这个过程可能是有 n 个节点也在进行的。因此可能会收到其它节点发过来的 commit 消息
REPLY：当收到 2f+1 个 commit 消息后（包括自己），代表大多数节点已经进入 commit 阶段，这一阶段已经达成共识，于是节点就会执行请求，写入数据。
PBFT算法的优点:
PBFT算法具有高交易通量和吞吐量，高可用性，易于理解。
PBFT算法的缺点:
A、计算效率依赖于参与协议的节点数量，由于每个副本节点都需要和其它节点进行P2P的共识同步，因此随着节点的增多，性能会下降的很快，但在较少节点的情况下可以有不错的性能，并且分叉的几率很低，不适用于节点数量过大的区块链，扩展性差。
B、系统节点是固定的，无法应对公有链的开放环境，只适用于联盟链或私
有链环境。
C、PBFT算法要求总节点数n>=3f+1(其中，f代表作恶节点数)。系统的失效节点数量不得超过全网节点的1/3，容错率相对较低。

3.2.2 POW

概率性一致算法，达成的共识结果是临时的，但是随着时间的推移，结果越不可逆转。具有51%的算力攻击风险。
POW（Proof of Work），即工作量证明，也称挖矿。工作量证明通过计算来猜测一个数值(nonce)，使得拼凑上交易数据后内容的Hash值满足规定的上限。由于Hash难题在目前计算模型下需要大量的计算，可以保证在一段时间内，系统中只能出现少数合法提案。如果能够提出合法提案，证明提案者确实已经付出了一定的工作量。
工作量证明的主要特征是客户端需要做一定难度的工作得出一个结果，验证方却很容易通过结果来检查出客户端是不是做了相应的工作。
比特币区块由区块头和该区块所包含的交易列表组成。区块头大小为80字节，其构成包括：

4字节：版本号
32字节：上一个区块的哈希值
32字节：交易列表的Merkle根哈希值

4字节：当前时间戳
4字节：当前难度值
4字节：随机数Nonce值
80字节长度的区块头，即为比特币Pow算法的输入字符串。
交易列表附加在区块头之后，其中第一笔交易为矿工获得奖励和手续费的特殊交易。
工作量证明过程，即为不断调整Nonce值，对区块头做双重SHA256哈希运算，使得结果满足给定数量前导0的哈希值的过程，其中前导0的个数，取决于挖矿难度，前导0的个数越多，挖矿难度越大。
POW算法的优点：
A、完全去中心化
B、节点自由进出
C、安全性高
POW算法的缺点：
A、记账权向资本集中
B、挖矿造成大量的资源浪费。
C、网络性能太低，需要等待多个确认，容易产生分叉，区块的确认共识达成的周期较长，不适合商业应用。

3.2.3 POS

POS（Proof of Stake），即权益证明，是POW的一种升级共识机制，根据每个节点所占代币的比例和时间，等比例的降低挖矿难度，从而加快找随机数的速度。
POS原理的核心概念为币龄，即持有货币的时间。例如有10个币、持有90天，即拥有900币天的币龄。
POS算法的优点：
A、不消耗大量算力挖矿，节省能耗。
B、在一定程度上缩短了共识达成的时间
C、防作弊。
POS算法的缺点：
A、本质仍然需要挖矿，未解决商业应用的痛点
B、极端的情况下会带来中心化的结果
更多区块链资料详见主页： https://learnblockchain.cn/people/436

3.2.3 DPOS

DPOS（Delegated Proof of Stake），即股份授权证明算法，是在POW及POS基础上诞生的一种新型共识算法。DPOS既能解决POW在挖矿过程中产生的大量能源过耗的问题，也能避免POS权益分配下可能产生的信任天平偏颇的问题。
DPOS是由被社区选举的可信账户（超级节点，比如得票数前101位可以成为）来创建区块。比如选出101个超级节点，即101个矿池，超级节点之间的权利是完全相等的。普通的持币者可以随时通过投票更换超级节点（矿池），DPOS的去中心化不是每个持币者就有直接的股份权益，而是需要间接的投票权力，来保证被推选出来的超级节点不作恶，同时也可以自己拉选票成为超级节点或者备用超级节点。

DPOS算法的优点：
A、能耗优势，网络运行成本低。
B、理论上更加去中心化。
C、较快的确认速度。出块速度秒级，交易确认分钟级。
DPOS算法的缺点：
A、坏节点不能被及时处理，只有经过选举才能清除坏节点。
B、小散投票积极性不高。
C、依赖代币

3.3 CAP原理

- 一致性，Consistency。任何事务应该都是原子的，所有副本上的状态都是事务成功提交后的结果，并保持强一致。
- 可用性，Availability。系统(非失败节点)能在有限时间内完成对操作请求的应答。
- 分区容忍性，Partition。系统中的网络可能发生分区故障(成为多个子网，甚至出现节点上线和下线)，即节点之间的通信无法保障。而网络故障不应该影响到系统正常服务。

CAP原理认为，分布式系统最多只能保证三项特性中的两项特性。当网络可能出现分区时，系统是无法同时保证一致性和可用性的。要么，节点收到请求后因为没有得到其它节点的确认而不应答(牺牲可用性)，要么节点只能应答非一致的结果(牺牲一致性)。
由于大部分时候网络被认为是可靠的，因此系统可以提供一致可靠的服务；当网络不可靠时，系统要么牺牲掉一致性(多数场景下)，要么牺牲掉可用性。

四、密码学

4.1 PKI

PKI体系提供了一套完整的证书管理的框架，包括生成、颁发、撤销过程等，解决了证书生命周期相关的认证和管理问题。
PKI至少包括如下核心组件:
CA(Certification Authority)：负责证书的颁发和吊销(Revoke)，接收来自 RA 的请求，是最核心的部分，主要完成对证书信息的维护；
RA(Registration Authority)：对用户身份进行验证，校验数据合法性，负责登记，审核过就发给CA；
证书数据库：存放证书，多采用X.500系列标准格式。可以配合LDAP 目录服务管理用户信息。

常见的操作流程为，用户通过RA登记申请证书，提供身份和认证信息等；CA 审核后完成证书的制造，颁发给用户。用户如果需要撤销证书则需要再次向CA 发出申请。
CA证书的签发实质上是CA机构使用 自己的私钥对用户公钥进行签名，任何人都可以用CA机构的公钥对证书进行校验。

4.2 同态加密

同态加密(Homomorphic Encryption)是一种特殊的加密方法，允许对密文进行处理得到仍然是加密的结果。即对密文直接进行处理，跟对明文进行处理后再对处理结果加密，得到的结果相同。从抽象代数的角度讲，保持了同态性。

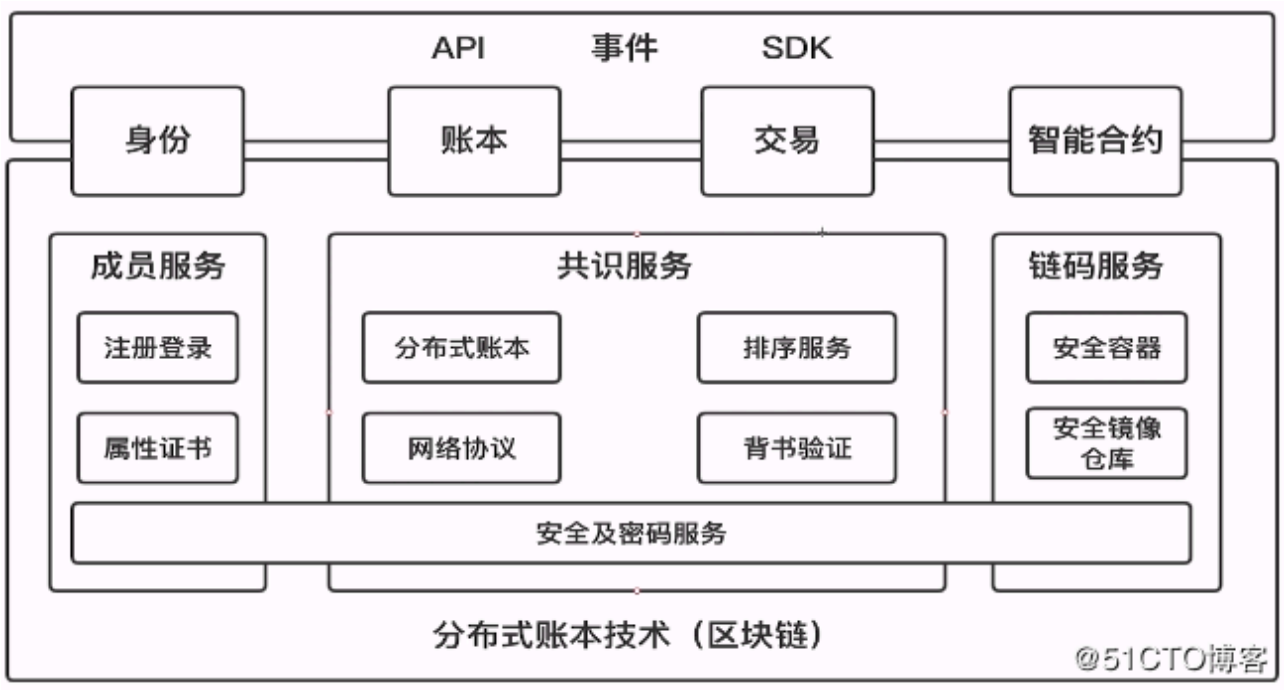
4.3 零知识证明

零知识证明(Zero Knowledge Proof)是证明者在不向验证者提供任何额外信息的前提下，使验证者相信某个论断是正确的过程。
证明过程包括交互式和非交互式两种。
zk-SNARK 它是一种非常适合区块链的零知识验证技术，可以让别人在不知道具体交易内容的情况下验证交易（或者是智能合约函数调用）的有效性。有了zk-SNARK，我们既保留了区块链互相不信任个体间的共识达成问题，又保护了交易隐私，简直就是在众目睽睽下原地隐身。

五、Fabric

5.1 Fabric逻辑架构

Fabric逻辑架构根据不同角度进行划分：
上层基于应用程序角度进行设计，包括SDK、API、事件，通过SDK、API、事件来对底层区块链进行操作：包括身份管理、账本管理、交易管理、智能合约的部署和调用；
下层基于底层区块链进行设计，对外提供成员管理服务、共识服务、链码服务、安全和密码服务。
Fabric通过将各个部分分离成不同的模块，做到可插拔性、灵活扩展性。



（1）成员服务管理
MSP（Member Service Provider）成员服务模块对成员管理进行了抽象，提供包括会员注册，身份保护、内容保密、交易审计等功能，可以使用可插拔的Fabric-CA模块或第三方的CA来代替。
（2）共识服务
共识服务负责节点间共识管理、账本的分布式计算、账本的存储及节点间的P2P协议功能的实现，是区块链的核心组成部分，为区块链的主体功能提供了底层技术支撑
（3）链码服务

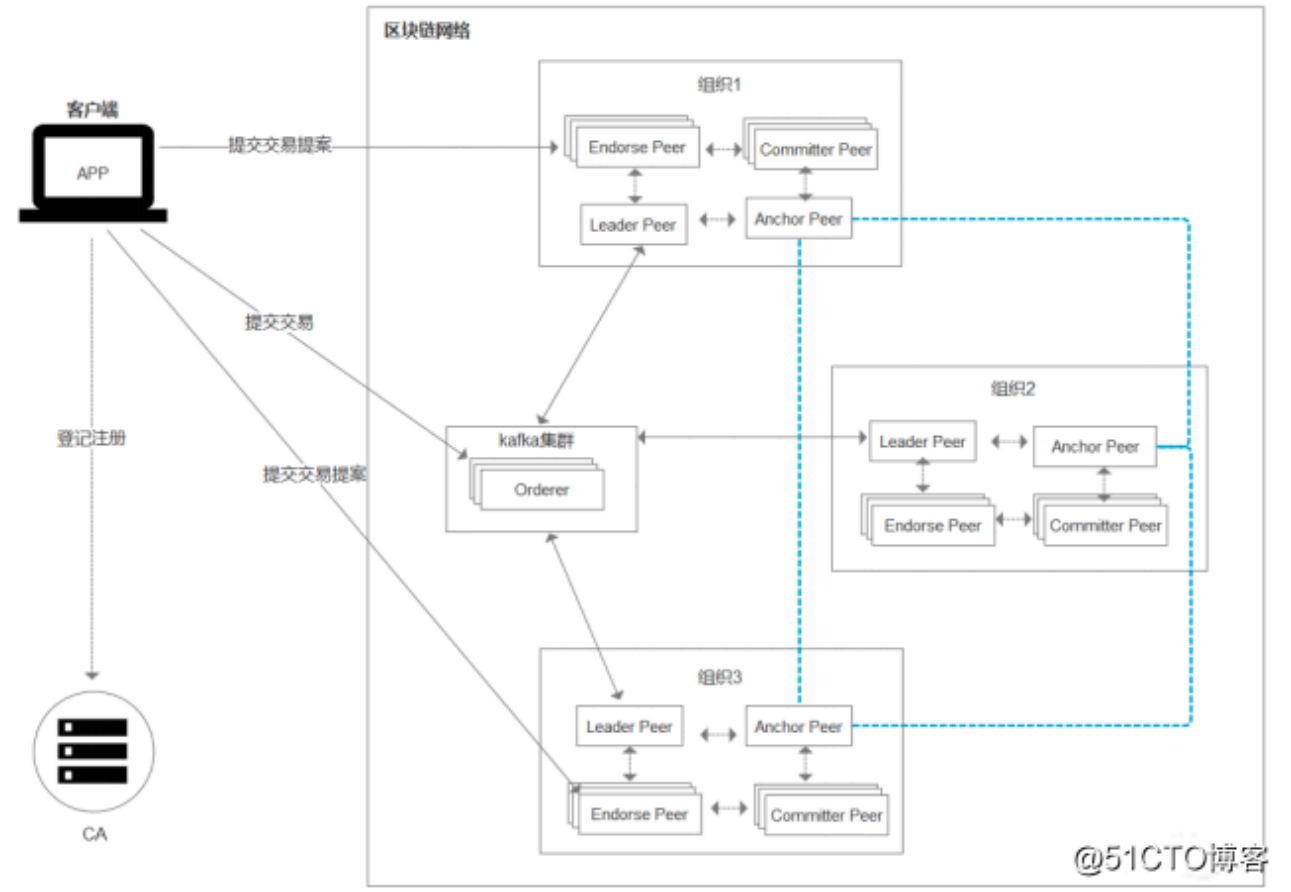
链码服务为智能合约实现提供了一系列接口，并为链码的安装、运行、部署提供了环境。智能合约的实现依赖于安全的执行环境，确保安全的执行过程和用户数据的隔离。Fabric采用Docker管理普通的链码，提供安全的沙箱环境和镜像文件仓库，可支持多种语言的链码。

（4）安全和密码服务

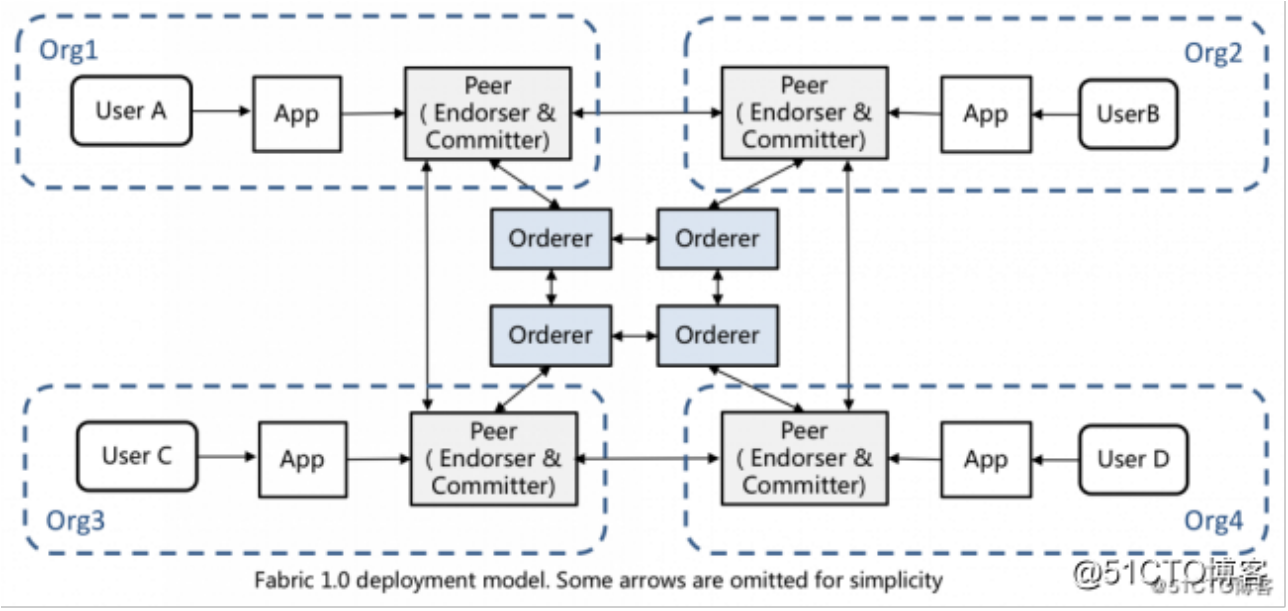
安全问题是企业级区块链关心的问题，Hyperledger Fabric专门定义了一个BCCSP（BlockChain Cryptographic Service Provider）模块，实现密钥生成、哈希运算、签名验签、加密解密等基础功能。

5.2 Fabric网络架构

Fabric网络是通过组织来划分的，每个组织内都包含承担不同功能的Peer 节点，每个Peer节点又可以担任多种角色。所有的组织共用一个统一的Orderer排序服务集群。



每个组织通常拥有自己的客户端、Peer节点和CA节点，并且可以根据需要创建一个或多个不同的类型节点。Orderer节点不属于某个组织的实体，属于组织共同维护的节点。



1、 客户端节点

客户端或应用程序代表由终端用户操作的实体，必须连接到某一个Peer节点或者排序服务节点上与区块链网络进行通信。客户端向背书节点（Endorser Peer）提交交易提案(Proposal)，当收集到足够背书后，向排序服务节点广播交易，进行排序，生成区块。

2、CA节点

CA节点主要给Fabric网络中的成员提供基于数字证书的身份信息，可以生成或取消成员的×××书（certificate）。CA节点是Fabric网络的证书颁发节点(Certificate Authority)，由服务器(fabric-ca-server)和客户端(fabric-ca-client)组成。

3、Peer节点

Fabric区块链网络中的每个组织可以拥有一到多个Peer节点。每个Peer节点可以担任多种角色：Endorser Peer（背书节点）、Leader Peer（主节点）、Committer Peer（记账节点）、Anchor Peer（锚节点）。

每个Peer节点必定是一个记账节点，除记账节点外，也可以担任其它一到多种角色，即某个Peer节点可以同时是记账节点和背书节点，也可以同时是记账节点、背书节点、主节点，锚节点。

（1）Endorser Peer（背书节点）

部分Peer节点会执行交易并对结果进行签名背书，充当背书节点的角色 。

背书(Endorsement)是指特定Peer节点执行交易并向生成交易提案(proposal)的客户端应用程序返回YES/NO响应的过程。

背书节点是动态的角色，是与具体链码绑定的。每个链码在实例化的时候都会设置背书策略(Endorsement policy)，指定哪些节点对交易背书才有效。

也只有在应用程序向节点发起交易背书请求时才成为背书节点，其它时候是普通的记账节点，只负责验证交易并记账。

（2）Leader Peer（主节点）

主节点负责和Orderer排序服务节点通信，从排序服务节点处获取最新的区块并在组织内部同步。可以强制设置，也可以选举产生。

（3）Committer Peer（记账节点）

负责验证从排序服务节点接收的区块里的交易，然后将区块提交（写入/追加）到其通道账本的副本。记账节点还将每个块中的每个交易标记为有效或无效。

（4）Anchor Peer（锚节点）

在一个通道上可以被所有其它Peer节点发现的Peer节点，通道上的每个成员都有一个Anchor Peer(或多个Anchor Peer来防止单点故障)，允许属于不同成员的Peer节点发现通道上的所有现有Peer节点。

4、Order排序节点

排序服务节点接收包含背书签名的交易，对未打包的交易进行排序生成区块，广播给Peer节点。排序服务提供的是原子广播，保证同一个链上的节点为接收到相同的消息，并且有相同的逻辑顺序。

排序服务节点按照一定规则确定交易顺序后，发给各个记账节点，把交易持久化到区块链的账本中。排序服务节点支持互相隔离的多个通道，使得交易只发送给相关的记账节点（Peer节点）。

更多区块链资料详见主页： https://learnblockchain.cn/people/436

5.3 Fabric交易流程

1、客户端构造交易提案

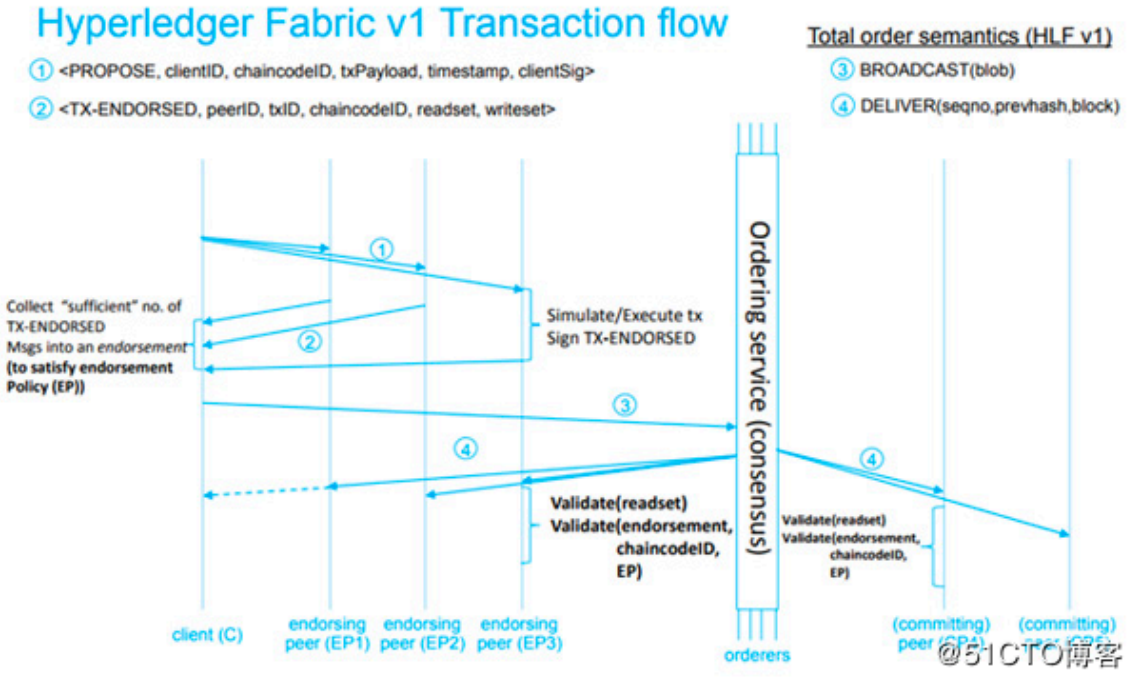
客户端应用程序利用SDK构造交易提案（Proposal）。交易提案是一个调用智能合约功能函数的请求，用来确认哪些数据可以读取或写入账本。

客户端把交易提案发送给一个或多个背书节点，交易提案中包含本次交易要调用的合约标识、合约方法和参数信息以及客户端签名等。

SDK将交易提案打包为可识别的格式（如gRPC上的protobuf），并使用用户的加密凭证为该交易提案生成唯一的签名。

2、背书节点模拟交易执行

背书节点（endorser）收到交易提案后，验证签名并确定提交者是否有权执行操作。背书节点将交易提案的参数作为输入，在当前状态KV数据库上执行交易，生成包含执行返回值、读操作集和写操作集的交易结果（此时不会更新账本），交易结果集、背书节点的签名和背书结果（YES/NO）作为提案的结果返回给客户端SDK，SDK解析信息判断是否应用于后续的交易。



3、客户端把交易发送给排序节点

客户端应用程序（SDK）验证背书节点签名，并比较各节点返回的提案结果，判断提案结果是否一致以及是否参照指定的背书策略执行。

客户端收到各个背书节点的应答后，打包到一起组成一个交易并签名，发送给排序服务节点。

4、共识排序，生成新区块

排序服务节点对接收到的交易进行共识排序，然后按照区块生成策略，将一批交易打包到一起，生成新的区块，调用deliver API投递消息，发送给确认节点。

5、交易校验

确认节点收到区块后，会对区块中的每笔交易进行校验，检查交易依赖的输入输出是否符合当前区块链的状态，完成后将区块追加到本地的区块链，并修改K-V状态数据库。

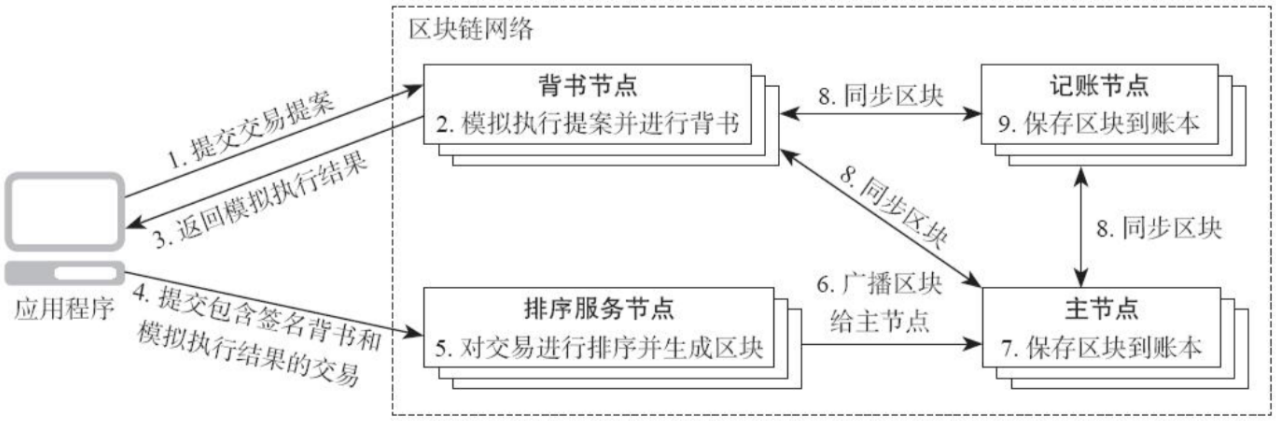


图3-4 交易流程总图

5.4 Fabric账本

Fabric中每个通道有一个唯一的账本，有通道成员维护，每个记账节点都保存了其所属通道的账本的副本。

账本由**区块链**和**状态数据库**两部分组成。

区块链是一组不可更改的有序的区块（数据块），记录着全部交易的日志。每个区块中包含若干个交易的数据，不同区块所包含的交易数量可以不同。区块之间用哈希链（Hashed-link）关联：每个区块头包含该区块所有交易的哈希值以及上一个区块头的哈希值。链式结构可以确保每个区块的数据不可更改以及每个区块之间的顺序关系不可更改。因此，区块链的区块只可以添加在链的尾部。

状态数据库记录了账本中所有键值对的当前值，相当于对当前账本的交易日志做了索引。链码执行交易的时候需要读取账本的当前状态，从状态数据库可以迅速获取键值的最新状态。

区块链的数据块以文件形式保存在各个节点中。状态数据库可以是各种键值数据库，Fabric缺省使用LevelDB，也支持CouchDB的选项。CouchDB除了支持键值数据外，也支持JSON格式的文档模型，能够做复杂的查询。

更多区块链资料详见主页： https://learnblockchain.cn/people/436

5.5 Fabric链码

1、链码简介

在Fabric中，智能合约也称为链码（chaincode），分为用户链码和系统链码。

链码被编译成一个独立的应用程序，运行于隔离的Docker容器中，在链码部署的时候会自动生成链码的Docker镜像。

2、背书策略

背书策略是背书节点如何决策交易是否合法的条件。链码实例化时可指定背书策略，当记账节点接收到交易时，会获知相关链码信息，然后检查链码的背书策略，判断交易是否满足背书策略，若满足则标注交易为合法。

3、系统链码

常见系统链码如下：

生命周期系统链码（LSCC ）：处理生命周期管理。

配置系统链码（CSCC）：处理在Peer节点上的通道配置。

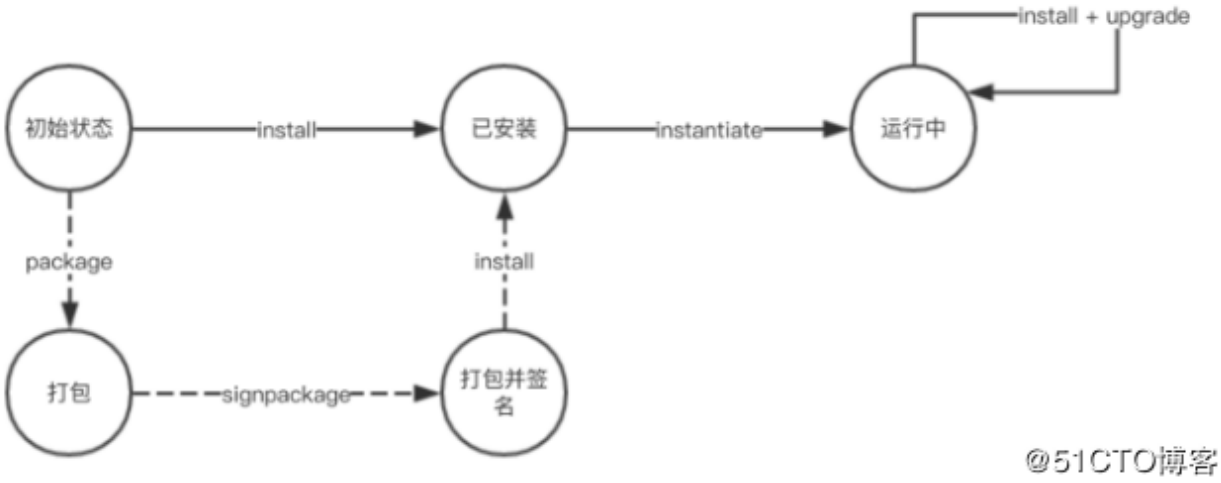
查询系统链码（QSCC）：提供账本的查询API，例如获取区块以及交易。

背书系统链码（ESCC）：通过对交易提案响应进行签名来处理背书过程。

验证系统链码（VSCC）：处理交易验证，包括检查背书策略以及多进程并发控制。

4、链码生命周期

Fabric提供了 package，install，instantiate和upgrade 4个命令管理链码生命周期。



通过install安装链码，通过instantiate实例化链码，然后可以通过invoke、query调用链码和查询链码。

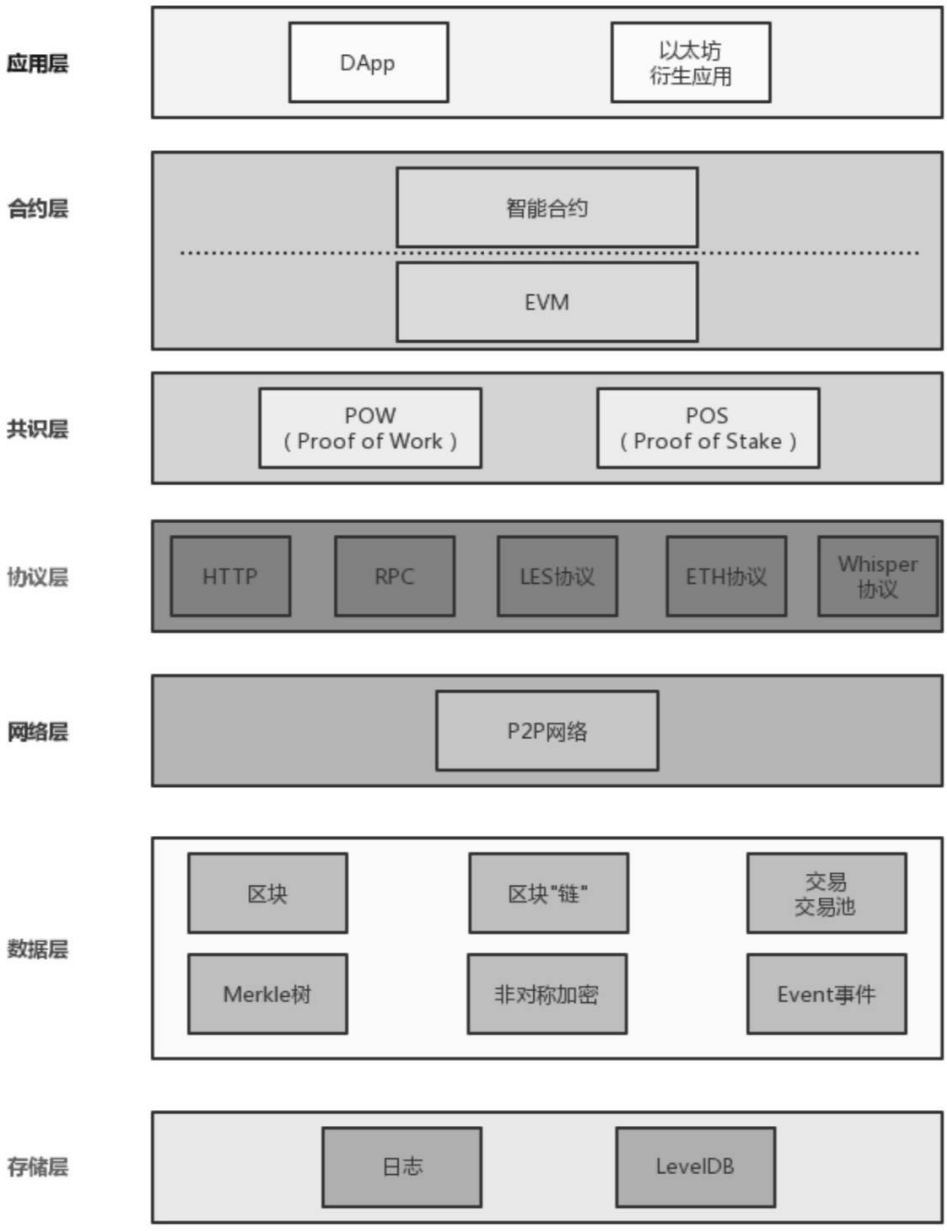
如果需要升级链码，则需要先install安装新版本的链码，通过upgrade升级链码。

在install安装链码前，可以通过package打包并签名生成打包文件，然后在通过install安装。

链码在成功install以及instantiate后，链码处于运行状态，能够通过Invoke方法来处理交易，后续能够对链码进行升级。

六、以太坊

6.1 以太坊架构图



6.2 以太坊交易流程

1、用户A发起转账交易到以太坊网络节点A中，交易信息包括：from、to、value、nonce、gas、gasprice、data

2、节点A检查交易有效后，将交易利用p2p网络同步给其他对等节点。检查有效包括交易格式是否正确，签名是否正确，是否有足够的余额等

3、有效交易被放到交易池中，获取到记账权的节点从交易池中将该交易打包到区块中，（如果是合约调用，矿工会在本地EVM中执行合约代码，直到代码运行结束或者gas消耗完毕）

4、记账节点广播区块信息，其他节点收到区块后执行并验证交易。

6.3 如何构造签名交易

1、对合约方法及参数进行ABI编码。编码结果包含两部分，方法的函数标识码（方法的SHA256取前四位）；参数的byte32编码。

2、构造Transaction交易对象。包括nonce、address、value、gasLimit、gasPrice、input

3、交易签名。

- 对交易对象进行RLP编码，计算rlpHash
- 对rlpHash使用私钥进行签名
- 填充交易对象中的V,R,S字段

4、发送已签名的交易信息
更多区块链资料详见主页： https://learnblockchain.cn/people/436