• 比特币前端

    ○ 钱包：钱包保存用户私钥，管理用户余额，提供比特币交易(支付，转账)功能

        § 分类

            □ 决定性钱包：决定性指私钥是否由种子生成

            □ 非决定性钱包：直接保存私钥，私钥都保存在DB上面

                ® 非决定性钱包不够安全

                ® 管理不方便

            □ 决定性钱包：所有私钥都由一个私钥种子通过单向的哈希算法生成

                ® 备份方便，只要备份私钥种子，就可以一次性恢复所有的私钥

                ® 分类

                    ◊ 普通决定性钱包：由私钥种子直接生成所有的私钥

                    ◊ 层级决定性钱包：由私钥种子生成父私钥，父私钥生成子私钥

        § 部署场景分类

            □ 移动钱包：运行在手机，移动终端的轻量级钱包。移动钱包不会下载整条区块链，采用“简化支付验证(SPV)”方法验证交易，也叫SPV钱包。

                ® SPV验证：依靠网络上的可信任节点 查询所有区块的区块头，以及按照交易的确认数，再有就是能否在相应的区块中找到该笔交易来验证支付的真伪。

            □ 桌面钱包

                ® 分类

                    ◊ 厚钱包：下载整条区块链，进行完整交易

                        } bitcoinCore:提供完整钱包功能，包括签名，钱包加密，备份，密钥导入，导出等

                        } 厚钱包主要优势是安全，但有交易验证的开销，适合于安全性要求很高的场景

                    ◊ 薄钱包：不会下载整条区块链，而是采用SPV等方式来验证与用户相关支付交易

                        } 薄钱包主要优势是灵活性高，但是安全性相对较差，适合于小额支付场景

            □ 互联网钱包

                ® 互联网钱包也不下载整条区块链，它依托于第三方的平台为用户提供隐私保护

                    ◊ 其优势是可以在任何地方，任何设备上管理钱包，但安全性不高

                    ◊ 产品：coinCorner,Blockchain.info等

            □ 纸钱包

                ® 将私钥进行冷备份，可以用于防范电脑或者USB介质损坏所造成的私钥丢失，而且由于其离线存在，可以有效防范黑客通过网络攻击盗窃私钥。需要防止物理偷窃

        § HTTP/JSON RPC API

            □ 比特币提供的接口，供外部通过接口控制比特币节点。

            □ 比特币提供了一个HTTP/JSON RPC的服务端，外部程序可以通过JSON RPC API来访问比特币节点。

        § 命令行工具 bitcoin-cli

            □ 控制比特币的节点。通过JSON RPC API接口访问比特币后台bitcoind。用户可以通过发命令来守完成比特币的各项操作，比如查询余额，支付，转账等

        § 比特币浏览器

            □ 比特币提供一个跨平台的C++ Libbitcoin库，该库支持比特币全节点服务端和浏览器作为客户端命令行工具。比特币浏览器命令提供与bitcoin-cli基本一样的功能。

            □ 额外功能：密钥管理功能，处理工具。

        § 图形开发工具(QT)

            □ 比特币核心是比特币使用最广泛的客户端，采用QT开发完成

• 节点后台

    ○ 功能：负责参与比特币网络的通信互联，维护区块链，验证区块，交易，广播，转播传递区块交易信息。

    ○ 比特币的后台程序主要由bitcoind与挖矿节点程序构成。

    ○ 区块链管理：涉及初始区块链下载，连接区块，断开区块，校验区块和保存区块，以及发现最长链条的区块。

        § 下载区块链

            □ 在比特币全节点第一次加入网络运行时，先要下载并验证整条区块链。当区块链容量不断增大时，要花费时间就越来越长。所以在新版本中，比特币发布了新的初始区块的下载方式。叫“区块报头等待”。该方式可以大大提高初始区块的下载速度。在此模式下，新的节点 先从邻节点下载所有的区块报头(一个报头占80个字节，而一个区块有1M，因此，只下载报头可以极大的提速)。当所有报头完成下载之后，节点就可以并行的从多个邻节点下载不同区间的区块，从而大大提高整条区块链的下载速度。

        § 接收区块链

            □ 在现有的节点 开户时，会先将整个区块的索引从LevelDB调进内存。需要注意的问题在于该区块链的索引不是单条的链，而可能是一个树，也就是说每个区块只有一个父区块，但可能有多个子区块，因此可能会导致分叉，这就需要逐渐发现哪个子区块属于最长的区块链。

            □ 当一个节点接收到到一个新的区块，如果该区块包含的指向前区块的区块头哈希值与节点当前顶端哈希值相同时，该节点就会尝试接收新的区块，并将其作为当前链条的最顶端的区块，从而 延伸节点所拥有的区块链。

        § 区块链的验证

            □ 在区块链管理中，连接区块函数是一个检测双花交易的关键。原理：该函数对新接收的区块中所有交易进行检测，验证是否每个交易的比特币来都能在当前的“尚未花费比特币”(Unspend Transaction Output-UTXO)记录中找到匹配。如果遇到网络延迟的情况，节点接收的区块可能 不按顺序到达，在这种情况下，有些交易的比特币来源可能在UTXO记录中暂时找不到，但当后面收到延迟到来的区块后，UTXO记录会被更新，区块链就能连接起来。

        § 重组区块链

            □ 当节点发现网络中有一条不基于它当前区块链的一条更长的区块链时，它需要断开现有区块并对区块链进行重组。这种情况主要发生在不同矿工几乎同时挖到合法的区块时，就会产生分叉。

            □ 断开区块，重组区块链，这些涉及到UTXO更改，被断开的区块中交易会回退到交易内存池，此时“回滚”记录就可以用来 回滚断开区块中的交易。

    ○ 区块验证(验证内容)

        § 交易的格式必须正确

        § 交易的输入和输出不能为空

        § 交易大小不能超过定义的区块最大值MAX\_BLOCK\_SIZE

        § 每个交易的输出以及所有输出的合计必须在一定范围内：0~2100万比特币

        § 输入的哈希值不能为零，不应该转播挖矿交易

        § nLockTime不能大于INT\_MAX

        § 交易的字节大小必须要大于等于100

        § 交易的签名操作数不能超过签名操作的上限

        § 解锁脚本只能把数字放入堆栈，锁定脚本必须是标准格式

        § 和收到的交易相匹配的交易必须能在当前交易池或者主链上的某个区块中找到

        § 对交易的每个输入，如果其对应的UTXO输出能在当前的交易池中找到，该交易必须拒绝(双花交易)。因为当前交易池是未经记录在区块链中的交易，而交易的每个输入应该来自于确认的UTXO，如果在当前交易池中找到，就是双花交易。

        § 对交易的每个输入，如果其对应的UTXO输出不能在主链或当前交易池中找到，则其是一个孤儿交易，应该放入孤儿交易池中

        § 对交易中的每个输入，如果其对应的UTXO输出是一个挖矿初始交易，该初始应该获得100个确认区块的确认

        § 对交易中的每个输入，其对应的输出必须是UTXO

        § 用对应的输出交易来获得输入值，检查每个输入的值及其合计，应该在允许的区间(0~2100W比特币)

        § 如果一个交易的输入合计小于输出总计，则必须拒绝该交易

        § 如果交易的费用太低，则拒绝该交易

        § 每个输入的解锁脚本必须和相应的输出的锁定脚本共同验证交易的合规性

    ○ 内存池管理

        § 内存池的管理就是交易池管理

        § 节点将通过验证的交易放在一个交易池中，准备放到一个挖到的区块中。

        § 当矿工挖到一个合格的区块后，他将按照一定的优先级次序中交易池中选出交易放到区块中。

        § 优先级是按照交易中的输入 对应的UTXO的“链龄”和交易额的大小来划分的。越老UTXO交易以及交易额越大的交易优先级越高。

        § 优先级计算公式：Priority = Sum(Value Of input \* Input Age) / Transaction Size

            □ UTXO链龄：按已在链上记录该交易的区块为起点，按后面有多少个区块来计算，也就是计量该区块在区块链上的“深度”。

            □ 交易的大小以字节为单位。

            □ 要成为高优先级，一般来说高优先级的值高于57600000，这个优先级相当于1个比特币交易额的优先级，已经有了一天的“链龄”(相当于已经有144个区块的确认)，交易长度是250字节。比特币区块中的前50K字节都会保留给高优先级的交易。

        § 当区块被填满后，剩下的交易都会留在内存池，等待下一个区块的到来。随着等待时间越长，“链龄”会逐渐增加，它们以后被选中的机率会越来越大。内存池中的交易不会过期，但是需要注意，内存池的交易不会被保存在硬盘上，一旦挖矿节点重启，内存池的交易会被清空。

    ○ 邻节点管理

        § 当一个新比特币节点做初始启动的时候，它需要发现网络中的其它节点，并至少与一个节点相连。

        § 一般情况 下是与一个已知节点在8333商品建立 TCP连接。

        § 新节点是如何发现邻节点的？

            □ 方法一：用一些"DNS种子"查询DNS,"DNS种子"是提供比特币节点地址的DNS服务器。比特币核心带有5个不同的"DNS种子"，可以提供稳定的比特币节点地址。

            □ 方法二：直接把一个已知的邻节点作为种子节点，通过它发现更多的邻节点。当发现新的邻节点后，新节点一般会断开和种子节点的连接。将节点将其信息发送给邻节点，邻节点会继续将新节点的地址转发给它们的邻节点，这样新的节点就能够被网络上其它节点知道，并保持其在网络上连接的畅通。新的节点一般会维持与8个邻节点的链接。新节点启动结束后，它会记住最近连上的邻节点的地址。当它重新启动的时候，它能够很快完成和已知邻节点的连接。如果以前的节点都连不上，就会重新开始初始启动流程。

            □ 节点维护

                ® 如果一个连接在一段时间内没有交互，节点就会定期发一些信息去维护连接，如果一个节点和邻节点的连接在超过90分钟里没有联系，该邻节点会被认为已经下线，节点会寻找一个新的邻节点来进行连接。

                ® 这么做的好处在于无需中心控制，网络节点可以自由加入或者离开网络，即比特币网络能动态的调节节点连接，以保证比特币网络的正常运行。

    ○ 共识管理

        § 比特币里主要的共识管理包括挖矿，区块验证，交易验证规则

        § 比特币的关键是在陌生的P2P环境建立共识机制，因此共识管理至关重要。

        § 比特币的共识管理必须支持前身兼容，即使过去的版本有缺陷也要保持，否则会导致比特币网络出现分叉

    ○ 规则管理

        § 比特币的共识规则是所有节点都必须遵守的规则。而每个节点都可以采用一些共识规则以外的修改化规则，这部分规则由规则管理模块实现。

        § 实例：交易存放内存池的规则。比如一个节点可以拒绝保存、中转大于200KB的交易。

    ○ 密码模块

        § 密码模块主要是处理比特币的地址，采用RIPEMD和SHA-256算法以及Base-58编码来生成比特币地址。

        § 比特币的公钥是通过私钥产生的，然后通过SHA算法SHA256和RIPEMD算法RIPEMD160对公钥进行处理，最后通过Base58编码形成比特币地址

        § Base58类似于Base64，就是过滤了一些容易混淆的字符，比如 数字“0”，O(大写o)，l(小写L)，I(大写i)，“+”，“/"

        § Base58Check的校验码对地址信息进行双重SHA256哈希处理，并取前4位做校验码，加在比特币地址后面，因此，比特币地址带有校验信息，可以防止人为错误

    ○ 签名模块

        § 比特币采用椭圆曲线数字签名算法(ECDSA)来实现数字签名以及生成公钥。

    ○ 脚本引擎

        § 比特币的脚本语言是一种专门设计的，与”Forth“类似的，基于堆栈的编程脚本语言。

        § 基于堆栈的语言的指令只按顺序执行一次，也就是说，没有循环或者跳转指令。

        § 脚本指令数会给一个程序运行时间的上限和所需内存的上限。

        § 设计原因：防止矿工提交可能有死循环的脚本。

        § 作为一个虚拟货币系统，比特币的这种设计足以满足需求，还有很大的空间去扩展

        § 比特币的脚本语言非常小，只能有256个指令，每个指令是一个字节长。

        § 256个指令中，有75个是保留指令，15个已经废弃。

        § 很多指令在比特币网络上不能执行，因为每个节点可以自己的脚本白名单，只允许运行白名单上的指令。

        § 常见指令如下：

            OP\_DUP  将堆栈头上的内容复制一份，并且压入堆栈

            OP\_HASH160  弹出堆栈头内容，先用SHA256对其做哈希处理，再用RIPEMD-160对结果做第二次哈希处理，结果压入堆栈

            OP\_EQUALVERIFY  弹出堆栈头上的两项内容，如果两个内容一样，返回”真“值，否则返回”假“值

            OP\_CHECKSIG 用输入的公钥检查输入的签名，如果签名符合，返回真，否则返回假

            OP\_CHECKMULTISIG    用提供的多个公钥检查多重签名的正确性

    ○ 挖矿

        § 比特币核心不带挖矿功能

        § 设备进化史：CPU-->GPU-->FPGA-->ASIC

        § 挖矿进化史：个人矿机-->矿池

        § 挖矿算法 SHA256算法

        § 挖矿原理：不断对区块报头进行哈希处理，每次尝试改变一个随机数，直到区块的报头的哈希值符合指定的条件：比如起始必须有多少个零，才算挖到一个合格的区块。

        § 由于哈希处理不可逆，也就是不能通过哈希值反推出输入值，因此不能预测输入的参数，只能随机测试。

        § 比特币挖矿的难度目标决定网络大约多长时间能够挖出一个区块。

        § 平均10分钟左右能挖出一个区块，可调整

        § 比特通过调整挖矿难度来调整出块速度。具体而言，每隔2016个区块，所有节点都要重新更新比特币的挖矿难度目标，所以其到2016个区块所需要的时间应该是20160分钟，如果实际时间小于20160，说明难度太低，需要加大难度，如果实际时间大于20160，说明难度高，需要降低难度

    ○ HTTP/JSON RPC服务器

        § 对外提供HTTP和JSON RPC的接口，外部程序可以通过JSON RPC接口来调用比特币的API，从而达到控制比特币节点的功能。

    ○ Berkeley DB和LevelDB数据库

        § Berkeley DB

            □ 比特币使用Berkeley DB作为钱包数据库，这是一个开源的文件数据库，这是一个轻巧而又性能高的嵌入式数据库。

            □ Berkeley DB可以保存任意类型的键值对，而且可以一个键保存多个数据。

            □ Berkeley DB可以支持数千个并发线程同时操作数据库，支持最大256TB数据

            □ 比特币的区块原始数据不是存放在数据库中，而只是作为文件类型存储在硬盘 上。

        § Level DB

            □ Level DB主要用于存储区块的索引和UTXO记录

            □ 它是一个非常高效的键值数据库

            □ Level DB的数据是冗余数据，可以用原始区块数据来重建。但如果没有Level DB的数据，比特币的校验和其它操作都会变得很慢。

    ○ P2P网络管理

        § 功能：在P2P网络上实现和其他邻节点的通信功能。

        § 通信功能包括：发现邻节点，连接并管理与邻节点的Socket连接，与邻节点交换不同的P2P消息(区块和交易)，还可以在特殊情况下禁止异常行为的邻节点连接。

        § 比特币的缺省配置是主动连接8个邻节点，同时允许最多125个其它邻节点发起连接请求

        § 防止DoS攻击：禁止异常行为的邻节点连接

        § 比特币节点分类

            □ 全功能节点

                ® 全功能节点带有钱包，RPC服务端，具有挖矿功能和进行节点校验区块和交易，并把区块和交易跳转给与之相连接的邻接点。

            □ 基础全节点

                ® 基础全节点也做区块和交易的交易和中转，但不挖矿，不带有钱包和RPC服务端

            □ SPV节点

                ® 校验过程

                    ◊ SPV节点一般会在与邻节点的连接中设置过滤器，该过滤器只接收包含钱包里的公钥地址的交易。

                    ◊ 当一个邻节点看到一个交易与SPV节点的过滤器设置的条件符合，它就会用merkelblock消息来给该SPV节点发送一个区块。该消息包含了一个区块头和一个连接该交易到Merkle树根的一个merkle路径。

                    ◊ SPV节点可以利用该merkle路径把交易和包含交易的区块联系起来，并验证交易包含在区块中。

                    ◊ SPV节点还可以用区块头来验证包含交易的区块和区块链中的其它区块能连上。

                    ◊ 通过以上两个验证，可以证明交易记录在区块连上。

        § ZMO队列管理

            □ 比特币采用ZMQ作为消息队列管理和消息分发工具。

            □ ZMQ是一个比较简单而且易用的传输层，提供一个像框架一样的socket库，使得Socket编程更加简单、简洁，性能更高。

        § ZMQ类似于一个底层的网络通信库，在Socket API之上做了一层封装，将网络通信、进程通信和线程通信抽象为统一的API接口。

---------------------

作者：水木·圳烜

来源：CSDN

原文：https://blog.csdn.net/u010986776/article/details/84143404

版权声明：本文为博主原创文章，转载请附上博文链接！