比特币的区块链分叉情况很多，但大致可以分为两大类：

1. **对区块链状态产生分歧: state fork**

前面学过，如果两个节点差不多同时挖到一个区块，这两个区块都是挂在当前的区块上的，不同节点先收到的区块不同，就会各自沿着先收到的区块往下扩展，这种时候就会出现临时性的分叉，称为state fork，即由于对区块链当前的状态有意见分歧而产生的分叉。

分叉攻击（forking attack）也属于state fork，只不过这种意见分歧是人为造成的，这种情况也称为deliberate fork。

1. **比特币的协议发生了改变：protocol fork**

要修改比特币协议需要软件升级，在去中心化的系统中，没办法要求所有的结点都升级软件（因为它们根本不需要哪个中心化的节点提供服务）。

假设大部分节点升级了软件，少部分节点没有升级（可能是没来得及升级，也可能是不同意协议的修改），这种分叉称为protocol fork，即对比特币协议产生了分歧，使用不同版本的协议而产生的分叉。

在protocol fork中，根据对协议修改的内容的不同，又可以分为硬分叉和软分叉。

1. 硬分叉

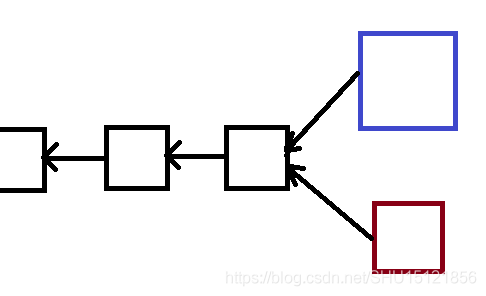
如果对比特币协议增加一些新的特性，扩展一些新的功能，这时候没有升级协议的那些结点是不认可这些特性的（认为它们是非法的）。

硬分叉的一个例子就是比特币中的区块大小限制，比特币限制每个区块不超过1M，这样算下来大约最多能容纳4000笔交易。而平均10分钟产生一个区块，算下来大约平均每秒只能写入7个交易。所以有的人就认为区块太小了，限制了交易上链的速度。

假设协议更新了，将区块大小的限制从1M提高到4M，同时我们假设大多节点更新了软件以支持这个协议。

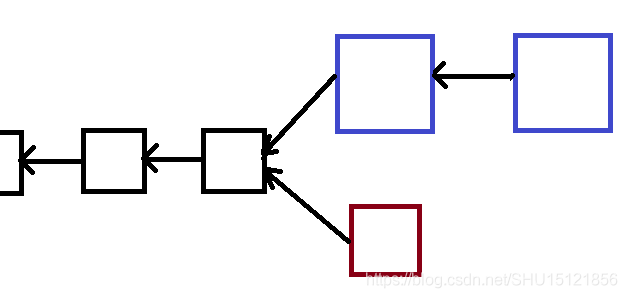
|  |
| --- |
| 注意，节点的“多数”和“少数”不是按照账户数目来算的，而是根据算力来算的，上面那句话是假设系统中拥有大多哈希算力的节点都更新了软件。 |

这样会有什么后果？假设新节点挖出一个区块（下图中蓝色），这个区块是比较大的（因为新的协议只要不小于4M），但旧节点是不认可这个区块的，不会沿着这个区块继续往下挖，而是继续沿着之前的区块往下挖下一个区块（图中红色部分）：

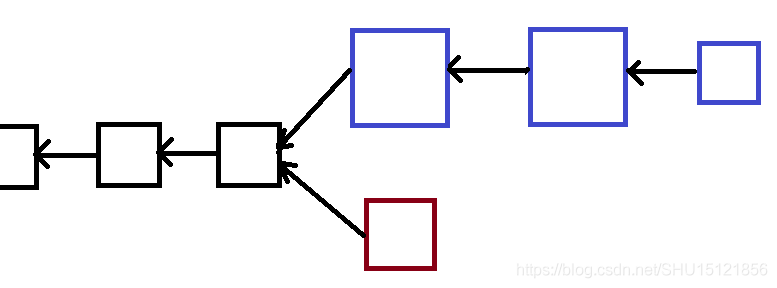


在这个分叉状态下，上面那个蓝色区块的分叉新节点认可，旧节点不认可；下面这个红色区块的分叉新旧节点都认可（因为新旧协议都符合，不超过1M一定不超过4M）。

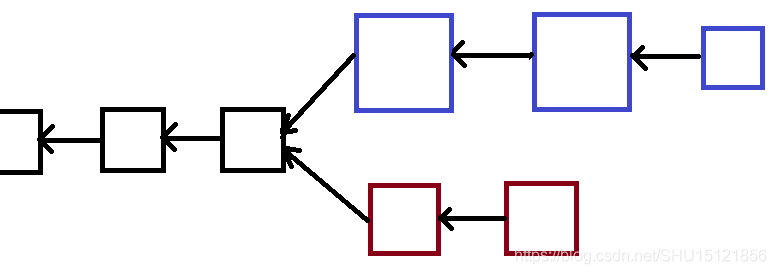
然而，假定大多节点都是新节点，即更新了软件支持新的协议，因为“大多数”即是其算力更强，新节点的新区块的分叉很快就比旧节点的分叉长了：



对新节点而言，上下两条链都是合法链，但因为只会去扩展最长合法链，所以还是会沿着上面的链往下挖。因为只是约束了大小不到4M就可以，新节点也可能挖出一些大小不到1M的区块：



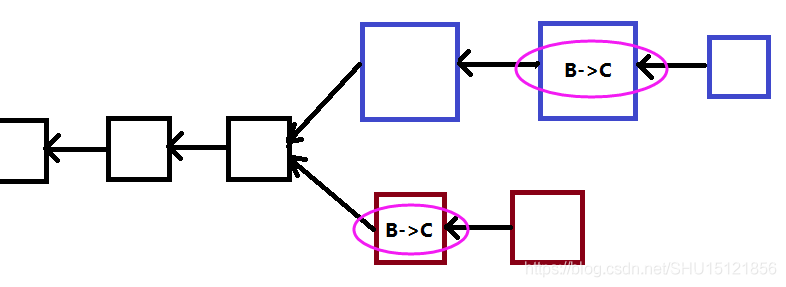
这样的区块是新旧节点都承认的，但上面这条链上有旧节点认为不合法的区块，所以旧节点始终不会去扩展这条链，还是继续沿着下面这条链往下挖：



因此，**这样的分叉是永久性的**，只要这些旧节点不更新软件，这样的分叉就不会消失。比特币网络中，会有部分很保守的人，像这样的协议更新势必会有一些节点不同意，产生硬分叉。

出现硬分叉之后，就相当于社区分裂了，出现了两条平行运行的链，两条链上的BTC也是不相干的，各挖各的矿。在某条链上的出块奖励，对于认可这条链为最长合法链的节点而言是有效的，对认可另一条链的则是无效的，而分裂之前产生的BTC则是在两条链上都认可的。从这个意义上来看，硬分叉可以认为是产生了新的一种加密货币。

硬分叉之后，如果不采取任何措施，会产生一些问题。因为两条链上的账户的私钥、公钥等都是一样的，仅仅是运行的协议不同。**按理来说两条链上的账户余额应该不一样才对，但不采取措施时就不行**。例如，在新链上有一笔交易B->C，那么对旧链而言这笔交易也是完全合法的，所以C可以把这笔交易发布出去，旧节点挖矿时就会把这个交易**回放**了：



这样就相当于B从一个钱包中拿出一定数量的币给C，导致B的**另一个钱包**中也转给了C同样数额（注意不是价值，如以太币ETC和硬分叉后的ETH）的币到相应的钱包中。

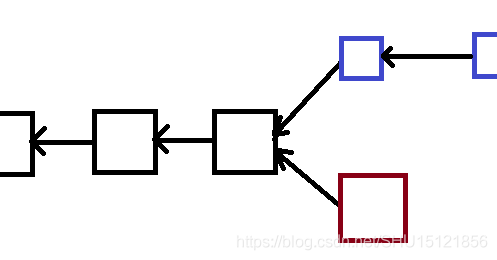
为了解决这个问题，可以在硬分叉后设置**chain ID**，来标识这两条链为**两条独立的链**。

1. 软分叉

如果对比特币协议加了一些限制，使得**原本**某些**合法**的交易或区块，在限制后的新协议中**变得不合法**，那么形成的分叉是软分叉。

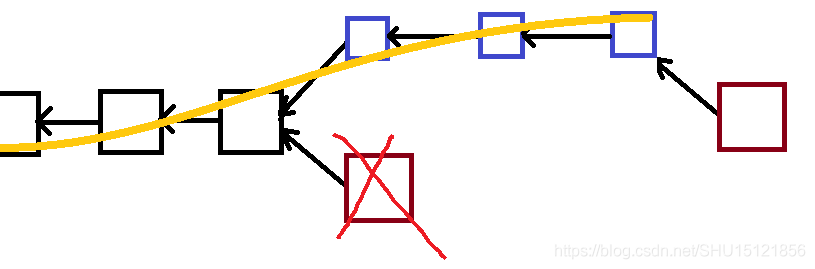
和前面学习硬分叉时候的例子相对应，假设协议更新将区块的限制大小变小了，从1M变成了0.5M（实际肯定不会这样做）。还是假设大多（算力的）节点是新节点，即已经更新了协议，区块限制为0.5M；少部分（算力的）节点是旧节点，仍然认定区块限制为1M。

这时，新节点挖出的区块，旧节点会认为是合法的（因为在1M以内）；但是旧节点挖出的区块，新节点很可能不认为是合法的（因为很可能不在0.5M内）：

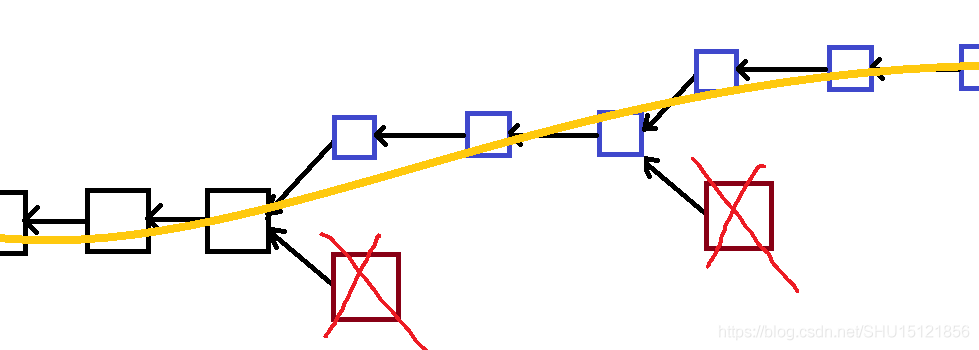


因为新节点占了大部分算力，所以很可能先挖到某个区块，出现上图的情况。这时旧节点观察到上面那条是最长合法链，就会放弃自己的分叉，接着上面的链继续挖。

某个时刻，旧节点先于新节点挖出一个区块，将其上链：



这个区块大于0.5M，新节点不认，会继续扩展上一个合法的区块。这样旧节点刚刚挖的区块又成为orphan block了：



所以在这种情况下，会持续出现软分叉，只要旧节点不更新协议，挖出的区块就一直无法上链。相比硬分叉，软分叉即是非永久存在的分叉，只会临时存在一段时间。

1. 软分叉的使用

协议分叉中，实际可能出现的只有软分叉。比特币系统也经常会利用软分叉来实现协议的更新。

1. **给某些目前协议中没有规定的域增加新的含义**

这种情况下即是当前协议中未限制的一些域，被赋予了新的规则。一个例子就是铸币交易的CoinBase域，没人规定也没人检查。前面学习挖矿难度时，提到这个域可以作为extra nonce来使用，比如拿出前8个字节来和nonce一起调整，以增大挖矿的搜索空间。

CoinBase即便拿出了前8个字节，后面还是有很长的可调整空间。有人就提出可以加入UTXO的根哈希值，因为目前这个UTXO集合只是每个全节点自己在维护，目的就是快速查找，判断交易合法性，这个集合的内容没有写到区块链里。

前面最开始学过Merkle proof可以证明某个交易存在于某个区块中，那么如何证明某个账户A中有多少钱？全节点可以在本地的UTXO集合里算一下，即找到UTXO中所有转账给A的交易的输出，加在一起。

但如果是轻节点呢？例如手机上的比特币钱包。轻节点要去请求全节点，全节点返回结果给它，如何证明全节点返回给轻节点的是正确的呢？轻节点自己没有维护一个UTXO集合，所以是证明不出来的。

因此有人提出将UTXO中的交易也组织成一个Merkle Tree，将其根哈希值写在铸币交易的CoinBase域里面，而铸币交易中的此内容也会随着影响交易的Merkle Tree的根哈希值，这在轻节点里是保存了的。所以在这种方式下就可以像Merkle proof的方式一样证明账户里有多少钱（需要提供UTXO的Merkle Tree对应位置的哈希）。

1. **增加新的功能**

前面学的P2SH（Pay to Script Hash）形式的交易脚本，最开始的比特币系统中是没有的，是后来通过软分叉的方式加进去的。

上节课学了，P2SH即在支付时输出不是收款人公钥的哈希，而是一个赎回脚本（Redeem Script）的哈希。

在赎回（把这些BTC花出去）时分为两阶段，第一阶段验证输入脚本中给出的赎回脚本，和前面交易的输出脚本中的赎回脚本的哈希对得上，第二阶段执行赎回脚本，验证输入脚本中给出的签名是合法的。

对于旧节点而言，并不知道赎回脚本的这些特性，旧节点只会做第一阶段的验证，即验证赎回脚本是不是正确的（和哈希对得上）。只有新节点才会把两阶段的验证都做完。

所以旧节点验证通过的交易，可能是第二阶段验证无法通过的P2SH形式的交易，在新节点上无法验证通过。而新节点验证通过的交易，旧节点也都验证通过。

1. **对protocol fork的总结**
   1. 软分叉的特点

如果是比特币系统协议要求变严格了，那么就会造成软分叉。只要系统中半数以上（算力）的节点更新了软件，就不会出现永久性的分叉。

* 1. 硬分叉的特点

如果是比特币系统协议要求变宽松了，那么就会造成硬分叉。必须系统中所有（算力）的节点都更新了软件，才不会出现永久性的分叉。