比特币分叉

区块链由一条链变为两条链就叫分叉。分叉可能是多种原因造成的，比如挖矿的时候，两个节点差不多同一个时候挖到了矿，就会出现一个临时性的分叉，我们把这个分叉叫作state fork，是由于对比特币区块链当前的状态有意见分歧而导致的分叉。

前面还讲过分叉攻击(forking attack)，它也属于state fork，也是属于对比特币这个区块链当前的状态产生的意见分歧，只不过这个意见分歧是故意造成的，人为造成的，所以我们又叫它deliberate fork。

除了这种state fork 之外，还有一种产生分叉的情况是，比特币的协议发生了改变，要修改比特币系统需要软件升级。在一个去中心化的系统里，升级软件的时候没有办法保证所有的节点同时都升级软件。

假设大部分节点升级了软件，少数节点因为种种原因可能没有升级，有可能是还没来得及升级，也可能是不同意对这个协议的修改。即假如你想把协议改成某个样子社区中可能是有人不支持的，这个时候也会出现分叉，这种分叉叫protocol fork(协议分叉)。因为对比特币协议产生了分歧，用不同版本的协议造成的分叉，我们称作protocol fork。

根据对协议修改的内容的不同，我们又可以进一步分成硬分叉和软分叉。出现硬分叉的情况:如果对比特币协议增加一些新的特性，扩展一些新的功能，这些时候那些没有升级软件的这些旧的节点，它是不认可这些新特性的，认为这些特性是非法的，这就属于对比特币协议内容产生了意见分歧，所以会导致分叉。

hard fork

硬分叉的一个例子就是比特币中的区块大小限制(block size limit)。比特币系统规定每个区块最多是1M字节，有些人认为1M的限制太小了，也增加了交易的延迟。可以计算一下:1M=1百万 一个交易大概认为是250个字节 1百万/250=4000 一个区块大概是4000个交易 平均10分钟出现一个区块 4000/(60×10)=7 大概每秒钟产生7笔交易即7tx/sec 这个传输速度是非常低的。

有人发布一个软件更新，把block size limit从1M增加到4M。假设大多数节点更新这个软件，把block size limit更新到4M，少数节点没有更新。这里的大多数节点和少数节点不是按照账户数目来算的，而是按照算力，即系统中拥有大多数哈希算力的节点都更新了软件。新节点认为区块大小限制是4M，旧节点认为是1M。

如图(第11分第40秒)这时运行系统，会有什么结果?假如一个新节点挖出一个区块，这个区块比较大，但旧节点不认可，它忽略大区块的存在会继续沿着它的前一个小区块接着挖。而旧节点如果挖出了区块新节点是认可的，因为4M的限制指不能超过4M，比4M小是可以的。

那为什么会产生分岔呢?大区块挖出之后，因为大多数区块是更新了的，是认可新的大区块的，所以会沿着它继续挖。只有少数旧节点会接着下面链往下挖，这时新节点认为上下两条链都是合法的，但上面那条是最长合法链，所以会沿着上面一条挖。而且算力足够大会使上面那条链越来越长。而旧节点认为上面的链无论多长都是非法的，它们只会沿着下面的链挖。当然上面的链也可能出现小区块，因为新节点也可能挖出大小不到1M的区块，虽然这种是新旧节点都认可的，但这是没有用的，因为这条链上它们认为有非法的区块。所以这种分叉是永久性的，只要旧节点不更新软件，分叉就不会消失，所以才叫它硬分叉。

比特币社区当中有些人是比较保守的，提高block size limit有些人就是不同意。而且区块的大小也不是越大越好，比特币底层系统是个P2P overlay network，它的传播主要采用flooding的方式，所以对带宽的消耗是很大的，带宽是瓶颈。

那么旧节点挖出的小的区块还有没有出块奖励呢？出现hard fork后出现了两条平行运行的链，平行运行链彼此之间有各自的加密货币。下面链的出块奖励在下面链里是认的。而分叉之前的币按道理应该是上下两条链都认可，所以会拆成两部分。

曾经出现过这样的问题:分叉前有A→B的交易，分叉后在上面链出现了B→C，下面链也出现了B→C，因为账户，私钥都是一样的。既然如此，就会有人利用这个特性，想收到上下两条链的转账。但如果没有人转账给他怎么办？

可以这样做:比如说B去购物，花一笔钱，给了C。后来B要退货，要取消这笔交易，C又把钱交给B。然后B又在下面一条链进行回放，就赚了一笔钱。那么在开始B转给C的交易在下面链会不会回放呢？所以这样做也是有风险的。为了解决这个问题，就让这两条链各带一个chain ID，所以现在以太坊的分叉已经没有问题了，就是两条独立运行的链了。

soft fork:

软分叉出现的情况是什么?如果对比特币协议加一些限制，加入限制之后原来合法的交易或区块在新的协议当中有可能变的不是合法了，这就引起软分叉。

假设有人发布一个软件更新，把这个区块大小变小了。调整区块大小不止是改变一个参数那么简单。一个去中心化的系统，改变一个参数，就可能导致分叉，而且取决于这个参数是怎么改的。有可能是硬分叉，有可能是软分叉。这里把区块大小变小只是为了解释软分叉这个概念，实际中是不会这么做的。

假设新节点把区块大小改为0.5M，旧节点依然以1M为准，这时候会出现什么情况？假如一个区块链开始分叉，新节点挖出小区块，这种区块旧节点也是认的。而旧节点挖出的大区块新节点是不认的。这样下去，旧节点看到上面链更长，而且是合法的之后，就会转去挖上面链。

所以为什么称这种分叉是软分叉?因为这种分叉是临时性的。所以旧节点如果不更新软件，它们挖的区块可能就白挖了。旧节点转向上面链挖的话，问题可能又会出现:它们可能又挖出了大区块。而新节点不认这个，新节点会继续沿着大区块前面一个小区块挖，如图(第29分第25秒)所示。

软分叉：旧节点仍承认新节点
硬分叉：旧节点不承认新节点，彻底分成两派

实际中可能出现**软分叉**的情况:**给某些目前协议中没有规定的域增加一些新的含义，赋予它们一些新的规则**，典型的例子就是coinbase域。前面讲过每一个发布的区块里可以有一个铸币交易(coinbase transaction)，coinbase transaction里有一个域叫coinbase域，这个域用来干什么是没人规定也没人检查的。

前面讲过coinbase域的一个用途:可以把它作为extra nonce。挖矿的时候要不断调整block header里的nonce，但block header里的nonce只有四个字节，最多只有2的32次方个可能性，所以实际中可以把coinbase前八个字节用来做extra nonce。两个合在一起就成了2的96次方，对于目前的挖矿难度，这个域已经是足够了。但coinbase域不止是八个字节，后面还有很多，剩下的字节有人就提议做UTXO集合的根哈希值。

目前这个集合只是每个全节点自己在内存中维护的，主要是为了快速查找、判断该交易是不是属于double spending，但这个集合的内容并没有写到区块链里，这跟前面讲到的merkle proof是不太一样的。

merkle proof能证明什么？证明某个交易是不是在给定的区块里。比如一个轻节点，没有维护整个区块的内容，只知道block header。轻节点问一个全节点:该交易是不是在这个区块里?全节点返回一个merkle proof作为证明，轻节点就可以验证是否属实。但如果是另外一种情况，想要证明某个账户上有多少钱，这个目前在比特币系统中是证不出来的。如果是全节点还可以算一下，方法如下:想要知道A账户有多少钱，就看一下A在UTXO里对应的输出总共收到多少个币，就是该账户上有多少钱。

对于全节点是可以算出来的，但如果是区块链钱包、有的手机上的APP，它不可能在手机上维护一个完整的区块链，它实际上是个轻节点，它想要知道账户的余额需要询问全节点。全节点返回一个结果，怎么知道这个结果是否属实呢？现在是证不出来的。如果你自己不维护一个UTXO集合，就没法用merkle proof 证出来。

有人提议把UTXO集合当中的内容也组织成一颗merkle tree，这个merkle tree有一个根哈希值，根哈希值写在coinbase域里面。因为block header没法再改了，改block header动静就太大了，coinbase域正好是没人用的，所以就写入UTXO的根哈希值。coinbase域当中的内容最终往上传递的时候会传递到block header里的根哈希值里。所以改coinbase域的内容，根哈希值会跟着改。

因此这个提案就是说把UTXO集合的内容组织成merkle tree，算出一个根哈希值来，写入coinbase域里某个位置。coinbase域的内容本身也会算哈希，算到block header里的根哈希值，这样就可以用merkle proof证出来了。

假设有人发布一个软件更新，规定coinbase域要按照这个要求来填写，大多数节点都升级了软件，少数节点没有更新，这属于软分叉，因为新节点发布的区块旧节点认为是合法的，因为旧节点不管新节点写什么内容。但旧节点发布的区块新节点可能是不认的，因为如果coinbase域不按要求写它是不认的，所以属于软分叉。

比特币历史上比较著名的软分叉的例子是pay to script hash。P2SH这个功能在最初的比特币版本里是没有的，它是后来通过软分叉的功能给加进去的。这是什么意思呢?你支付的时候不是付给一个public key的哈希，而是付给一个赎回脚本的哈希。花钱的时候要把这个交易的输入脚本跟前面币的来源的交易的输出脚本拼接在一起执行。执行的时候验证分为两步，第一步是要验证输入脚本中给出的redeem script跟前面那个输出脚本给出的script的哈希值是对的上的，证明输入脚本里提供的script是正确的。第二步再执行redeem script，来验证输入脚本里给出的签名是合法的。

对于旧节点来说，它不知道P2SH的特性，只会做第一阶段的验证，即验证redeem script是否正确。新节点才会做第二阶段的验证，所以旧节点认为合法的交易新节点可能认为是非法的(如果第二阶段的验证通不过的话)。而新节点认为合法的交易旧节点肯定认为是合法的，因为旧节点只验证第一阶段。

总结:soft fork是什么?只要系统中拥有半数以上算力的节点更新了软件，那么系统就不会出现永久性的分叉，只可能有一些临时性的分叉。hard fork特点是什么？必须是所有的节点都要更新软件，系统才不会出现永久性的分叉，如果有小部分节点不愿意更新，那么系统就会分成两条链。