recreate bitcoin

宋佳维誓教研究生写论文

宋佳维 [1], **宋佳维** [2], **宋佳维** [3]

(Dated: November 20, 2021)

Abstract: 本人在百忙中抽空写这个小作文，个人认为也是非常的给这门课程面子了。本文只是重新创造比特币的小作文。形式大于内容。我们创造性的提出了重新创造比特币的方法。实现了比特币各个功能模块并将它们有机的结合起来，构成了完整的比特币系统，可以实现交易的产生，交易的传播，交易的生成块，块的传播，以及挖矿等功能。

Keywords: bitcoin; python; 大作业; 密码货币;...

1 Introduction

随着密码技术互联网的发展,比特币横空出世。和与大多数货币不同，比特币不依靠特定货币机构发行,它依据特定算法,通过大量的计算产生,比特币经济使用整个 P2P 网络中众多节点构成的分布式数据库来确认并记录所有的交易行为，并使用密码学的设计来确保货币流通各个环节安全性。P2P 的去中心化特性与算法本身可以确保无法通过大量制造比特币来人为操控币值。基于密码学的设计可以使比特币只能被真实的拥有者转移或支付。这同样确保了货币所有权与流通交易的匿名性。比特币其总数量非常有限，具有稀缺性。该货币系统曾在 4 年内只有不超过 1050 万个，之后的总数量将被永久限制在 2100 万个。虽然这个东西给社会造成了一定的危害，但是不得不承认，区块链技术确实是有一定的经济价值和实际意义的。

1.1 Related Work

本文只是对于基础的比特币的复现，实际上，若是真的想要使用比特币，也有直接的函数库进行调用。比特币实现至今，已有多代的优化，根据发明比特币的人，以太坊的历史与比特币直接相关。比特币推出四年后，来自加拿大的 19 岁的程序员 Vitalik Buterin 创造了一个基于比特币的新平台。Buterin 的想法是创建一个超越比特币允许的金融用途的替代平台。这与比特币有所不同，以太坊没有供应上限。比特币的预设采矿限额为 2100 万比特币。更大的供应量意味着更多的人将有机会以实惠的价格购买以太坊。像比特币一样，以太坊是一个分布式的公共区块链网络，但它们在用途和能力上存在很大差异。虽然比特币是一种货币，但以太坊是一种象征。以太币是为了钱而做的，以太坊是为合同而做的。以太坊允许您编写一个数字协议，其中每个命令都会导致一个精确的操作，而没有其他任何操作。到目前为止，您在页面上签名以声明您同意合同条款。以太坊通过使交易证明数字化并永远不可磨灭，从而显着改善了这一陈旧的系统。然而，在比特币大行其道的同时，也是出现了让人意想不到的炒币狂潮，一时间，为数不少的只是修改几个参数的各种 coin 也是被开发了出来。如下图 Example of coins。这些币多数都是没什么价值的，但是所谓的质数币，可能稍微有些实际的价值，却也是弊大于利。

Figure 1: Example of coins.

1.1.1 contribution

我们只是重新创造了基础的比特币，仅用于学术用途, 个人学习使用。主要贡献在于交易的产生，交易的传播，交易的生成块，块的传播，以及挖矿等功能。已经完美实现其功能，但是代码的效率不能保证。

2 recreate bitcoin

Equations:

$$E = mc^2 \tag{1}$$

$$H\psi = E\psi \tag{2}$$

$\partial\partial = 0$, and

$$\iint_S \vec{F} \cdot \vec{n} d\sigma = \iiint \nabla \times \vec{F} dV$$

瞎写两个公式。区块链或者说是比特币的复现，重点在数据结构的重现；而比特币的有效性，在于其机制的有效性。有效的机制，使得坏人变成好人；而坏的机制，使得好人变成坏人。我们首先放弃使用中心化的电子货币思想，使用去中心化的思想，构建比特币的主要部分，这带来了很大的坏处，比如说，我们无法直接对于每个账户构建一个余额的数值，来判定用户能否实现这个交易。但是，这种去中心化的思想，也给我们带来的巨大的好处：首先，我们无需花费大量的成本，来构建中心的数据库，用以维护用户数据；其次，它也给了我们的系统一个巨大的可扩展性和鲁棒性。体现一种群体智慧，核心为点对点网络架构，Bitcoin 系统将朝着生命体的方向演进。我们一般希望，有一个中心化的服务器维护整个系统，但是这是另一种极端的途径。我们发现，许多系统都是将并行运作的部件拼接在一起，很像大脑的神经元网络或者蚂蚁群落。这类系统的动作是从一大堆乱糟糟且又彼此关联的事件中产生的。它们不再像钟表那样，由离散的方式驱动并以离散的方式显现，更像是有成千上万个发条在一起驱动一个并行的系统。由于不存在指令链，任意一根发条的某个特定动作都会传递到整个系统，而系统的局部表现也更容易被系统的整体表现所掩盖。从群体中涌现出来的不再是一系列起关键作用的个体行为，而是众多的同步动作。这些同步动作所表现出的群体模式要更重要得多。这就是群集模型。具体的技术细节，我们将会在下面的几个小节中详细描述。

群系统的重点在于漩涡模型，这个模型体现了博弈论的内核，我们比特币的安全性，也来自于此:

漩涡的内环：1）个体：个体拥有简单算法 2）组合：个体和个体之间依靠通信，实现组合，并且是信息订阅模式。

漩涡的外环：3）循环导致涌现智能：大量个体产生了概率不均，并且由于自循环，可以将状态保持住。

2.1 交易的产生

我们使用签名技术来实现所有权的验证，使用 sm2 算法的签名算法 (椭圆曲线公钥密码算法)，和 sm3 的 hash 算法。

sm2 基于的椭圆曲线如公式 (3)；sm3 的基本框架如 Figure3。

$$sm2 : y^2 = x^3 - x \tag{3}$$

一个交易的抽象结构如下，使用 JSON 表示。

```
{
  "txid": "135cf9b9cca2640113937dbf5b2fc62222ccf16c63556ccc3f2a5da899020efa",
  "vin": [
    {
      "txid": "db130da8831a37956c0613e8c9f1363f10a1de32faf46f6ef5ad3a3037df5c78",
      "vout": 1,
      "scriptSig": "43daeeba2a1d68339"//解锁脚本
    },
    {
      "txid": "c29ef6fc7af543a9c6af8965af39ab7b20f605a8817549c1fadd3b87621d8f2e",
      "vout": 0,
      "scriptSig": "fsjpesooe98832"//解锁脚本
    }
  ],
  "vout": [
    {
      "value": 3.5,
      "n": 0,
      "scriptPubKey": "3 OP_ADD 5 OP_EQUAL",//锁定脚本
      "addresses": "1HLBhLqDbV5rd7uNJZfFtFTgAuPa8Hwuy"//公钥Bob
    },
    {
      "value": 1.5,
      "n": 1,
      "scriptPubKey": "OP_DUP OP_HASH160 b9b9edb47415c3d6980fec683c60b8b74754df99
            OP_EQUALVERIFY OP_CHECKSIG",//锁定脚本
      "addresses": "1Hw2k2iuhzcrA1Rvm6EuCoiCSp7Sc6mdrv"//公钥Alice
    }
  ]
}
```


Figure 2: transaction section.

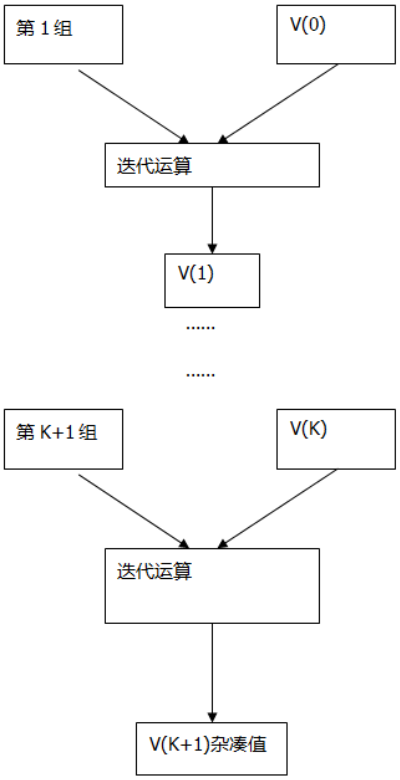

Figure 3: sm3.

具体的交易数据结构改造如下：

1.OUT 部分，给每一笔生成 UTXO 都加上锁定脚本：scriptPubKey

2.IN 部分，给每一笔引用的 UTXO 都加上解锁脚本：scriptSig

3. 移除之前交易中的 ScriptSig 签名脚本。

(见下图 transaction example. 红色为新改造的地方，我们会看到每一笔 UTXO 都有一对解锁和锁定脚本)

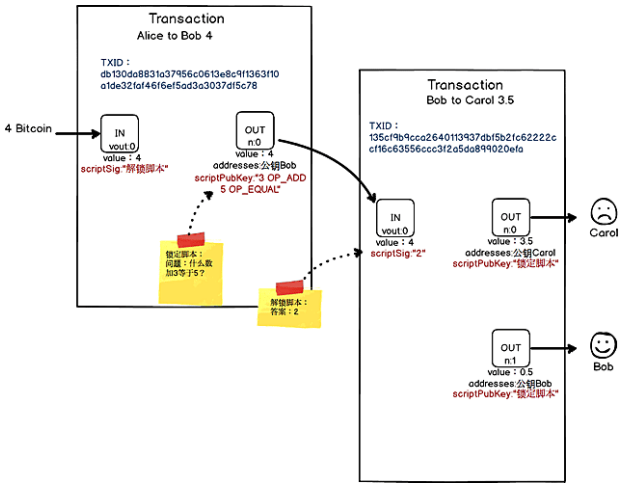

Figure 4: transaction example.

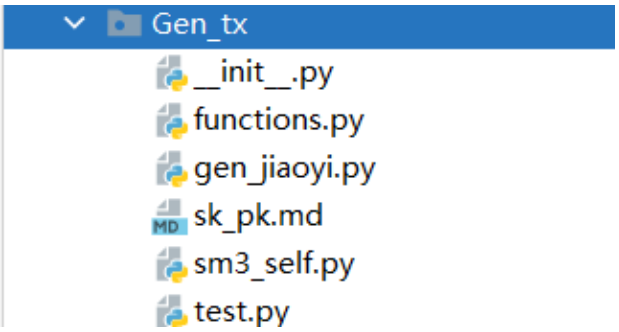

Figure 5: tx codes.

用户输入部分：

Figure 6: tx examples.

函数输出部分：

Figure 7: tx.

2.2 交易的传播

我们构建了在一台机器中构建了两个抽象端口:server 端和 client 端。server 端用以接收交易，client 端用以发送交易。并选择 3-4 个节点为本节点的活跃节点，交易只发送给活跃节点，而不是胡乱的广播；为了简化代码，活跃节点直接写在代码中。

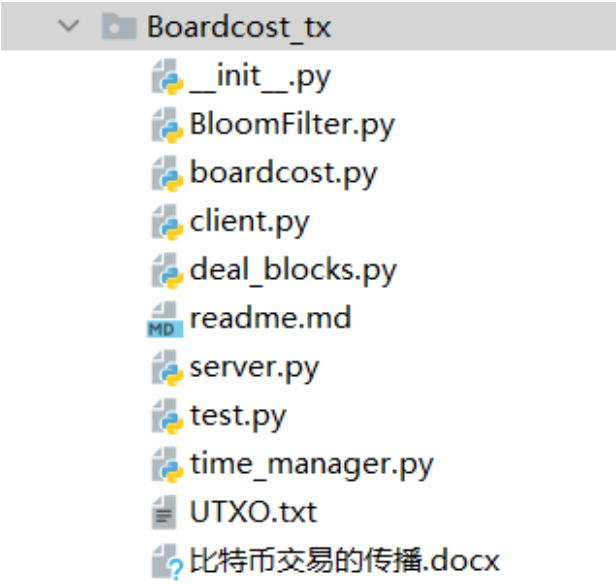

Figure 8: tx.

2.2.1 解决账本不一致的问题

只需要节点互相将自己的账本发送给对方节点，收到对方的账本后，找出自己账本中缺少的交易记录，然后将此交易记录补充到自己的账本，这样俩个节点的账本就又一致了。

每次节点接受到新的交易，都发送全量账本给对方节点，这样做太浪费网络资源了，其实只需要将新增的交易记录发送给对方节点就够了。对方节点只需要将这条交易记录补充到自己的账本即可。

1. 节点 1 接受到用户通过轻节点发来的最新交易：Alice to Bob 50

2. 节点 1 将这条交易记录写入账本，同时将交易记录通过自己的 Socket Server 广播出去。(这里之所以将 SocketServer 发送消息说成是广播，是因为一个 Socket Server 会连接多个 Socket Client，这个传播路径很像广播。而反过来看，Socket Client 主动连接 Socket Server 的行为则是：订阅，就好像用户通过收音机订阅了一个广播频道一样。所以，每个记账节点即能广播又能订阅。)

3. 节点 2 由于订阅了节点 1，所以可以收到这条新增交易消息。

4. 节点 2 不可以直接写入自己的账本，因为自己的账本中没准已经存在相同的交易记录。所以要验证这条交易记录是否已经在账本中存在。节点 1 补充进自己账本的交易记录是：Carol to Alice 30。节点 2 补充进自己账本的交易记录是：Alice to Bob 50。这样在俩个节点中的账本又恢复一致，Alice 的余额都显示为 30。见下图

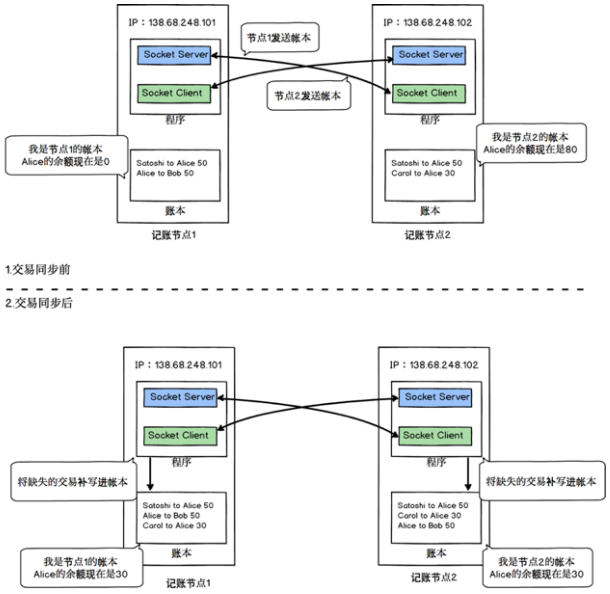

Figure 9: tx.

2.2.2 选择发送的节点

而节点 2 为什么要再次广播呢? 这是因为，如果节点 3 加入到了记账网络中，而节点 3 只订阅了节点 2，没有订阅节点 1，那么就需要节点 2 作为一个传话筒，将自己订阅节点 1 的同步消息传递给节点 3。

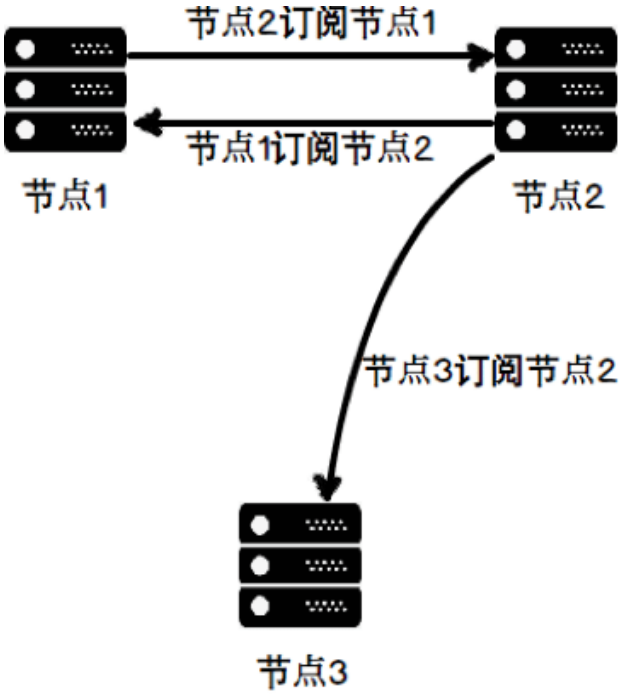

Figure 10: points.

2.2.3 独特的保存机制

既然账本不用频繁同步，我们可以再进行优化，在记账节点中创建一个交易的内存池。将每次收到的新增交易第一时间放入交易内存池中，而不是写入账本。等内存池中的交易积累了一段时间（例如每隔 10 分钟），再一次性的写入到账本中。

所以，实际上是设置了两个保存机制，一个为 menpool, 一个是 UTXO。menpool 接收到交易就更新，而 UTXO 由于是写文件，写硬盘，效率较低，10 分钟更新一次。

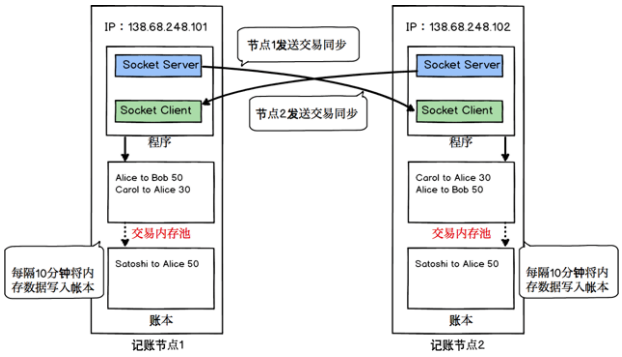

Figure 11: UTXO.

2.3 交易的打包

交易的传播和交易的打包为分离的两个传播机制。Bitcoin 的点对点网络架构中，已经实现了交易的同步和账本的同步。账本的本质是系统中过往交易数据的总和。所以账本代表着系统的过去。而交易内存池中的数据是系统实时进行的行为，所以交易内存池代表着系统的当下。记账这个动作，本质就是在做数据备份。由于全网都要依赖这份数据备份，所以拥有记账权的节点责任重大。在这个 10 分钟内，它承载着系统的正确性。所以，记账这个行为，也要记录在系统中。目的是可以追究记账者责任，当然除了追究责任，也可以给记账者一些利益，例如给记账者送几个 Bitcoin 作为奖励。这些都需要依赖这份行为记录。那么记录在哪呢？传统系统的设计思路，是将记账行为独立写入一个日志文件中，日志文件再与账本关联。但是，我们可以借助账本的拆解，来实现记账行为的记录。这就是所谓区块的概念。

换句话说，账本的拆解可以一石二鸟：既可以，避免全量数据传播的资源浪费。又可以，让记账行为关联的责任和奖励有的放矢。

区块之间唯一 ID 的关联，让后面的 Block 在自己文件的第一行写入前一个 Block 的唯一 ID。这样后面的 Block 就伸手主动的抓住了前面的 Block。一个抓一个就形成一条链了，直到创世区块：MetaBlock。

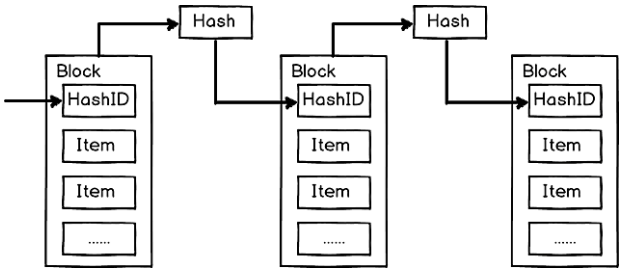

Figure 12: block chain.

我们也可这样来设计 Block，给 Block 文件划分为 2 个部分：第一个部分叫做区块头：Block Header，用来记录非业务数据：系统数据。第二个部分叫做交易集：Transactions，用来记录业务数

据：交易记录。见下图 block。

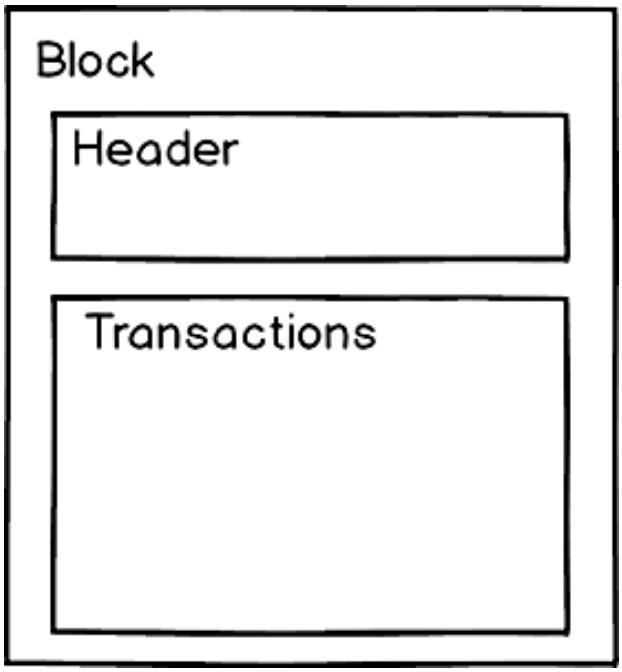

Figure 13: block.

2.3.1 交易打包实现方案

1，选取没有上链的交易，交易已经通过交易的传播获取，更新到最新版本。

2，按照交易中包含的金额排序，选取金额最大的几个交易进行打包，交易的合法性已经在交易的接收时候进行验证了。也可以不选择交易进行挖矿。选择的交易记录越多，收获的手续费就越多，但是处理逻辑和广播速度就会变慢。

3，prove of work 过程: 按照最新的一个区块 hash，和区块头中的其他信息，计算一个随机数 nonce。交易打包完成，发送给其他用户即可完成交易的上链。

区块头的数据结构如公式 (4)，所求的随机数 nonce 如公式 (5)。difficulty 为区块产生难度，由上两个区块的区块头信息决定，若时间间隔小于 10min, 则增加 1 点的 difficulty 数值; 否则减小 1 点 difficulty 数值。若区块的产生规则不满足这个规则，则其他的用户不进行传播；若有部分的用户认同这个区块，并在这个节点后继续增加区块，则区块链发生分叉，这一条链按照新的规则运行，与原本的区块链并行。这也是群策智慧的运用。

而在本算法的代码实现中，我们需要首先将没有打包的交易放入一个文本文件 (unpacklist.txt) 中。作为可能需要打包的交易。

$$blockmessage = \begin{cases} \\ "previoushash" : [], \\ "time" : [], \\ "nonce" : [], \\ "difficulty" : 8, \\ "MerkleRoot" : [] \\ "transaction" : [] \end{cases} \tag{4}$$

$$hash(blockmessage) < string(difficulty) \tag{5}$$

string(difficulty) 表示 hash 由 difficulty 个 0 开头。

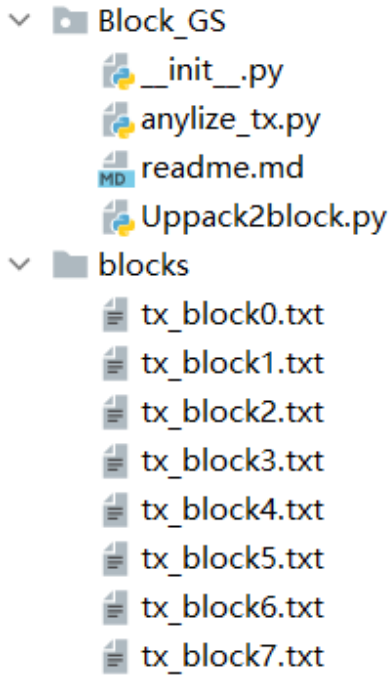

Figure 14: tx pack.

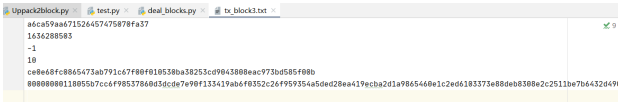

Figure 15: block example.

2.3.2 区块中交易的存储结构

其实也可以不用这一条，只要能用即可。区块中交易的存储结构以 Merkle Tree 表示，从最小粒度交易入手，第一轮分别计算每笔交易的 Hash 值：HA=Hash(TxA)、HB=Hash(TxB)、HC=Hash(TxC)、HD=Hash(TxD) 第二轮继续分治，将上一轮的产出物两两组合后继续计算 Hash 值：HAB=Hash(HA+HB)、HCD=Hash(HC+HD)

如此类推，直到最后剩下一个 Hash 值，我们这个例子只要到第三轮就到头了：HABCD=Hash(HAB+HCD)

这种归纳所有交易的结构就称之为 Merkle Tree，最后一轮得到的树根就是 Merkle Root，如下图。

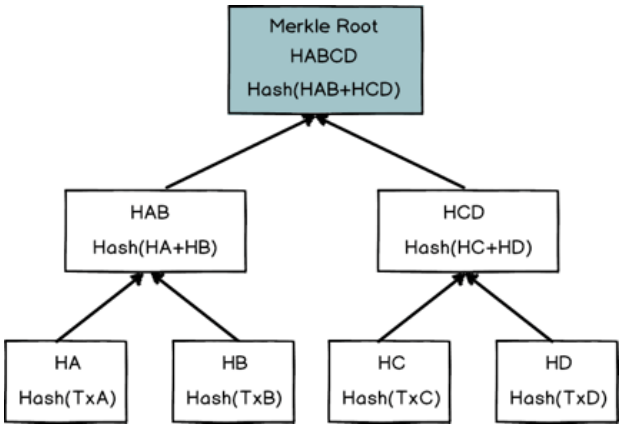

Figure 16: tree.

2.4 交易的上链

把上一节产生的区块发送给其他用户且其他用户也保存在本地链中即可完成。因为 Bitcoin 的点对点网络的特性很像水面，所以大概率，最先发出的水波会最先传遍整个水面。但是也有可能发生分叉。一般这样的分叉会按照最长链原则，较短的链无效；但是也有可能一直保持分叉，此时，就有了一个基于新的规则的链。

这一块部分，同样是和交易的传播相同，也是有 server point 和 client point，区块的传播部分和交易的传播不能使用同一个网络端口，client point 在自己产生新的 block 之后，就传播新的区块。

但是此处 server point 没有加入验证，实际上，有可能被造假。验证部分没有写入函数。client point 代码不需要单独运行，但是 server point 代码需要单独运行。server point 功能仅仅为接收文件，并保存为最新文件。验证部分个人认为不需要通过代码验证，直接通过人力验证即可。比如说，可以验证 hash 和验证 nonce 是否真的能使得最后的 hash 符合现在的 difficulty。

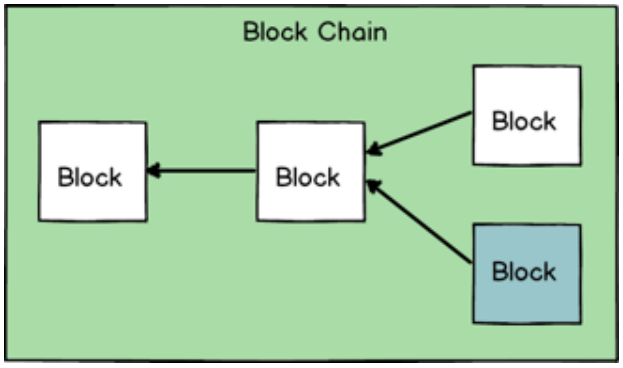

Figure 17: 分叉.

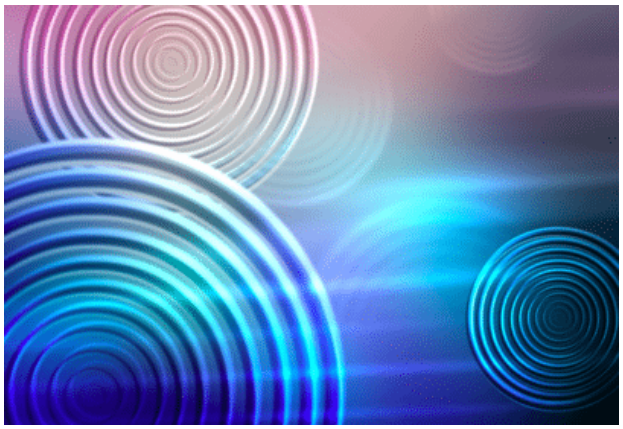

Figure 18: 传播.

Header 可以代表整个区块，Header 的 Hash 值就可以代替之前的区块 Hash 值，这样一来 Block

Chain 本质上就变成了 Block Header Chain。

Block Header Chain 的意义在于，随着系统发展，Transactions 部分越来越大，假设达到了区块平均 2TB，但是 Header 不会增长，永远不会超过 1KB。这时候某些记账节点为了节省存储空间，哪怕裁剪了某些区块的

Transactions 数据，只要还完整保留 Header 部分，就还是可以独立验证任意一个区块的数据是否被篡改。

换句话说，Bitcoin 系统用来保障诚实的存储成本可以很小。见图 header chain。

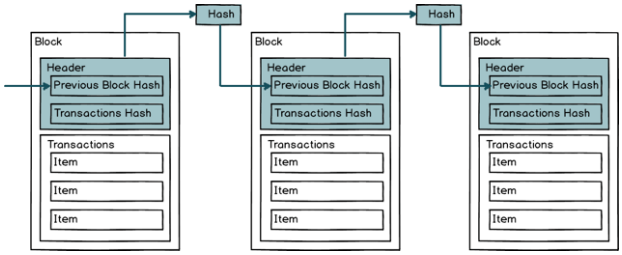

Figure 19: block header

3 Conclusion

由此，我们实现了一个基本的区块链系统，没有多余的机器，可能用处不是很大。

服务端包括这么几个部分：1. 网络连接：负责 Bitcoin 网络的自治。2. 交易处理：负责 Bitcoin 系统的交易处理，属于系统的现在时。3. 区块处理：负责 Bitcoin 系统的交易备份，属于系统的过去时。4. 交易内存池：存储实时的交易数据。5.Block Chain：存储过去的交易数据。区块处理部分：1. 计算随机数：就是上面提到的工作量证明，获得随机数。2. 生成区块：就是在记账，如果计算出了随机数，就可以将自己生成的区块广播出去，获得区块奖励。如果没有顺利计算出随机数，则放弃自己生成的区块，而选择同步网络中传递过来的区块。3. 区块同步：同步其它节点的区块消息到自己的本地账本（Block Chain）。4. 区块验证：通过一系列验证，来确保同步过来的区块是正确的。

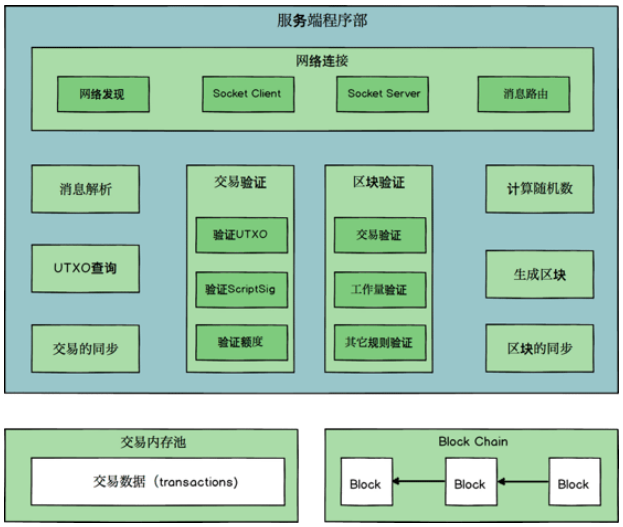

Figure 20: conclusion

Bitcoin 本身是车轮，全球账本（Metanet）才是车，比特币是这个全球账本上的通用货币，其目的是提供这辆车的燃料，维护这个不可篡改账本的安全以及提供经济激励。在有了使用价值之后，才延伸出“通用货币”，或者说“钱”的属性。只是在造车的时候，需要先造轮子，再有了轮子之后，才上车架。

Acknowledgments

These are acknowledgments. 我心中有大不平，忙着开新课题呢，引用文献就不加了。按道理说，至少应当给我 3 倍的分数。

References

[1] This is reference.

[2] This is reference.