

**课程报告**

**课程名称：区块链技术与应用**

**专业班级： 2020级计算机硕04班**

**学 号： M202073361**

**姓 名： 万 静 意**

**指导教师： 肖 江**

**报告日期： 2020/12/26**

**计算机科学与技术学院**

**目 录**

[1 报告题目：三种POX共识算法 3](#_Toc55309446)

[1.1 研究背景及目的意义 3](#_Toc55309447)

[1.2 研究内容及挑战 3](#_Toc55309448)

[1.3 研究成果及创新点 3](#_Toc55309449)

[1.4 测试环境/应用说明 3](#_Toc55309450)

[1.5 测试结果 3](#_Toc55309451)

[1.6 总结 3](#_Toc55309452)

**1 报告题目：三种POX共识算法**

**1.1研究背景及目的意义**

区块链是一个不可变的、透明的公共分类账，分布在网络的各个节点上。它是一个分散的系统，事务在不可信的设备上运行。为了确保平等和公平，交易必须根据一些协议进行处理，它们被称为共识算法[1]。共识算法是区块链的基础和核心，决定着区块链的工作方式，在维护区块链的安全和效率方面起着至关重要的作用。使用正确的算法可以显著提高区块链应用程序的性能，使得未经授权的用户几乎不可能破解区块中的机密信息。随着区块链技术的不断发展，运用于区块链的共识算法也在不断发展，研究其原理、优势、适用场景以及存在的问题，对于其能更好地应用于区块链中有着重要的意义。

**1.2研究内容及挑战**

1.2.1研究内容

（1）介绍了共识算法的基本原理和特点；

（2）重点研究了三种共识算法的机制和性能，并对其进行实现和测试；

（3）分析对比了PoW、PoS以及DPoS 的特点及其适合的应用场景，并对如何选择合适的共识算法提出自己的看法；

（4）简要总结了区块链技术中共识算法存在的局限性以及对未来发展展望。

1.2.2问题与挑战

由于本文只是对目前应用较为广泛的基本共识算法及其原理进行实现，并未对其最前沿研究以及实际应用进行深入探讨，因此深度和广度都不够。通过阅读相关的文献了解到了区块链中共识算法主要面临以下问题与挑战：

（1）成本过高;

（2）51%攻击问题依然存在;

（3）共识时间消耗过长；

（4）时间不统一存在的分叉。

**1.3研究成果及创新点**

1.3.1数据结构

如图一为基本的区块结构，本文实现的四种算法基本结构类似，部分有特殊的新增字段，区块头中包含以下元素：Index：区块高度；HashCode：包含该块的所有已验证事务的hash值；

（1）Prev\_hash:：对父项的引用，前一个块的hash值；

（2）TimeStamp：区块建立的时间戳；

（3）Diff：指示证明的难度级别；

（4）Nonce：随机数，即为添加区块到链上所付出的努力值；

（5）Transactions list：交易信息（即区块数据Data）用String类型来表示。

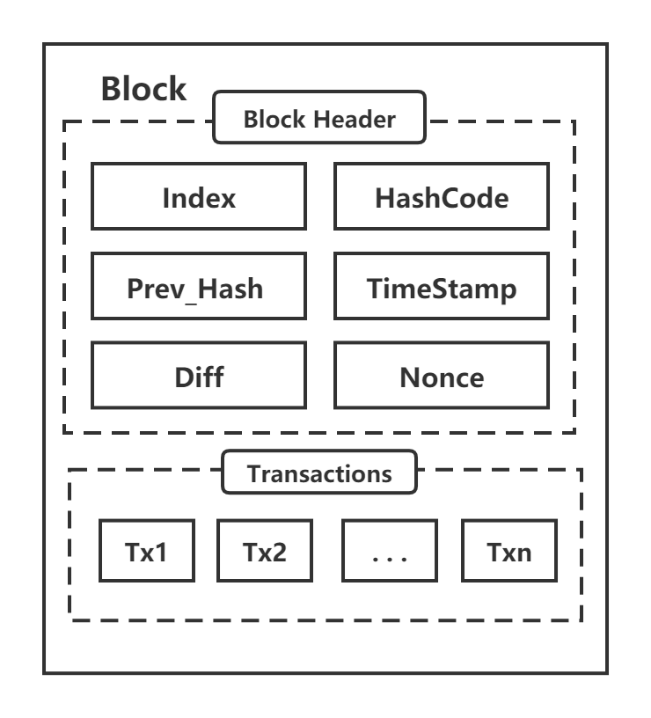


图1 区块数据结构示意图

Figure 1 The structure and fields of a block

1.3.2 PoW（Proof of Work）

1.3.2.1 基本原理

PoW，即工作量证明，又称为挖矿，基于前一个块的交易数据信息，不同节点计算一个数学问题的具体解。当某一个节点成功得到满足要求的解之后，会马上对全网进行广播，网络中的其余节点收到打包区块后对其进行验证。如果验证通过，则表明已有节点成功解谜，自己则不需要再竞争当前区块，而是接受这个打包块并记录到自己的账本中。只有最快求出解的节点才能获得奖励。

1.3.2.2 算法设计

中本聪用HashCash在比特币中设计了这个数学问题[2]，分为(i)获取难度; (ii)收集事务; (iii)计算和(iv)重启四个步骤。本文基于这个流程，为了方便测试，简化了其中的一些设定，如图2所示为算法的流程示意图。

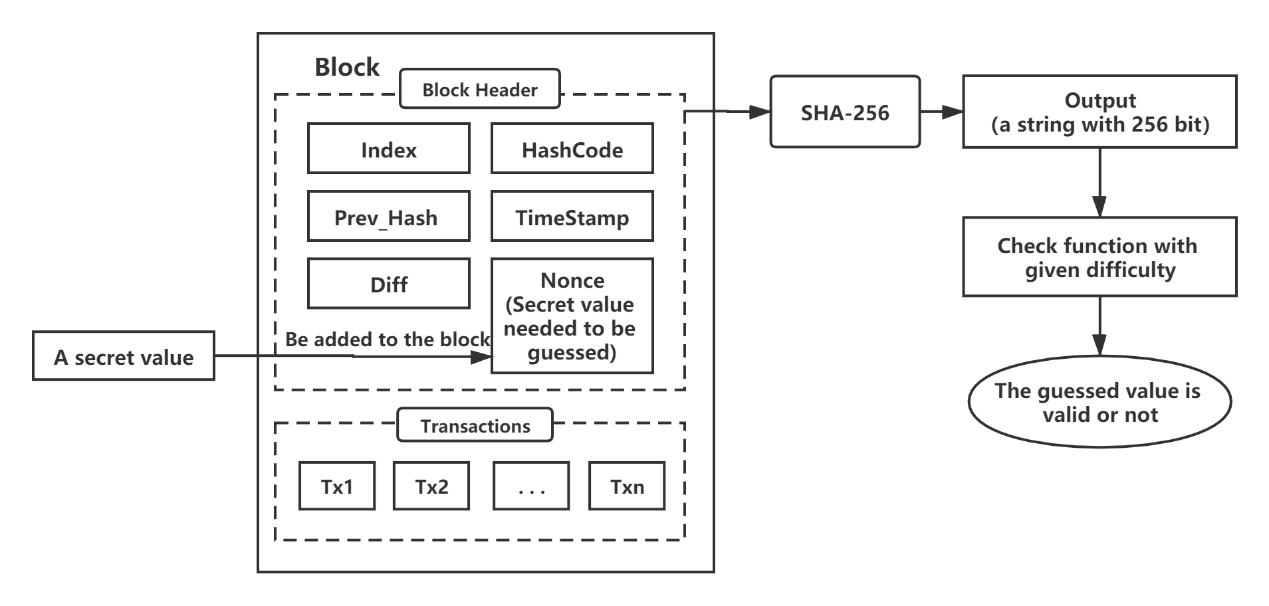


图2 处理随机数获取正确值的过程示意图

Figure 2 The process for handling with the nonce: guessed secret value

1.3.2.3算法实现

基于PoW的设计方案，对算法进行了简单实现（Algorithm 1），其中的关键工作量证明是通过对于一个给定初值的随机数，依次递增并计算hash值，直到其满足条件hash值的前个数为0，其中是指难度，这是一个动态调整的值。

|  |
| --- |
| **Algorithm 1** Proof of work |
| 1: **while** next block hasn’t been selected **do**  2: **for** **each** Node **do**  3: hash: = GenerateBlockHashValue(block)  4: **if** hash.hasPrefix(Repeat(“0”,diff)) **then**  5: **return** block  6: **else**  7: block.nonce++  8: **end if**  9: **end for**  10: blockchain.addNextBlock(block)  11: **end while** |

1.3.2 PoS

1.3.2.1基本原理

PoS（Proof of Stake），权益证明，是基于币龄（coin age）的概念，币龄=持有的货币数\*持有时间。币龄越大的节点获取记账权，也就是生成区块的概率越大。每当新的区块生成完毕，该节点的币龄就清空。目前应用最为广泛的两个基于PoS的共识协议是Oroboros和Casper[3]。如图3所示为PoS机制的基本流程。



图 3 PoS机制流程示意图

Figure 3 The process of how PoS algorithm to validate

1.3.2.2算法设计

最早应用POS的是PPCoin[4]，基于其对算法的描述进行了PoS的代码实现（Algorithm 2）。由于刚获取的币不能直接参与币龄计算，一般是30天之后开始计算币龄，设开始计算币龄的时间为，由于30天时间太长，因此缩短为。为防止数据过大，设最大累加时长为，同时了防止囤币来获取绝对记账权的情况发生，会设置一个最大概率，将其时长设为1分钟，即。当节点区块满足 时，则将其添加到链上，并在币池中累加其Coin。验证的的计算公式如下所示：

其中是指左位移，是指当前计算难度，，是当前区块的币龄，，即根据当前币龄对难度进行动态调整，用来证明的hash值的计算如下所示：

1.3.2.3算法实现

PoS算法（Algorithm 2）的实现难点在于计算和更新节点区块的币龄，并根据币龄动态调整难度目标。与PoW的验证算法不同，在PoS的实现中采用的是比较大小来验证hash值。

|  |  |
| --- | --- |
| **Algorithm 2** Proof of Stake | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17: | **while** next Block has not been selected **do**  **for** each Block b in CoinPool **do**  curTime=getCurrrentTime()  b.curCoinTime = ComputeCoinHasTimeUplimit(b.Time,curTime)  b.coinAge = b.curCoinNum\* b.curCoinTime  b.time=curTime  curDiff=initialDiff+increaseDiffUplimit(b.coinAge)  target=getTarget(currDiff)  proofHash= sha256(b.PrevHash,b.Data,b.Height,b.TimeStamp,nonce)  **if** proofHash<coinAge\*target **then**  return b  **else** **do**  nonce++  **end for**  blockchain.addBlock(b)  coinPool.addCoin(b.newCoin)  **end while** |

1.3.3 DPoS

1.3.3.1基本原理

DPoS（Delegated Proof-of-Stake），即委托权益人证明，是PoS算法的改进。是由被社区选举的可信账户（受托人），即得票数排名前的，按照指定的方式一个接一个地创建新地区块，并获得一些奖励[5]。用户根据自己持有的加密货币数量占总量的百分比来投票。DPoS机制类似于股份制公司[6]，普通股民进不了董事会，要投票选举代表（受托人）代他们做决策。

1.3.3.2算法设计

基于Bitshare[7]白皮书中对于DPoS的设计原则，算法的流程如图4所示，为了成为正式受托人，用户要去社区拉票，获得足够多（超过%50）用户的信任。且加入了对于恶意节点的惩罚[8]，受托人必须保持足够长的在线时长，如果无法创建指定区块，利益相关者（委托人）就会在下一轮投票中让它出局选举新的受托人。

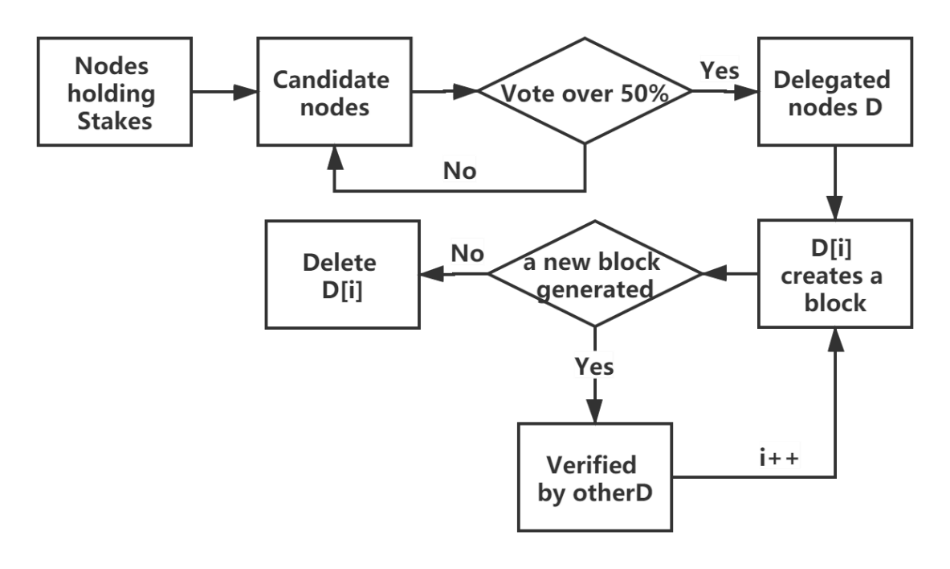


图 4 DPoS机制的流程示意图

Figure 4 The process of selecting delegates to create blocks

1.3.3.3算法实现

在具体的算法实现中（Algorithm 3），分成了很多轮，无限循环，并依次对选出的个受托人（打乱顺序）安排创建区块任务。

|  |  |
| --- | --- |
| **Algorithm 3** Delegated Proof of Stake | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17:  18: | **while(true)**  **for** round i **do**  dList\_i = get top N delegates sorted by votes  dList\_i = shuffle(dList\_i)  **loop**  slot = global\_time\_offset / block\_interval  pos = slot%N  **if** dList\_i[pos] exists in this node **then**  result b=GenerateNextBlock(KeyPair of dList\_i[pos])  **if** b==Null **then**  punish and delete dList\_i[pos]  **else**  blockchain.addBlock(b)  **else do**  skip  **else** **loop**  **end for**  **end while** |

**1.4 测试环境/应用说明**

1.4.1 测试环境

* 系统：Ubuntu 18.04.3 LTS
* 内存：８Ｇ
* 处理器：Intel® Core™ i7-7700HQ CPU @ 2.80GHz × 8
* 图形：llvmpipe (LLVM 8.0, 256 bits)
* 磁盘：SSD 256 GB

1.4.2 应用说明

* 实现语言：golang
* 编译环境：go version1.12
* 开发环境：vim go
* 运行环境：Linux终端

**1.5 测试结果**

1.5.1 PoW

在不同的难度（Diff）下进行了六组实验，统计了PoW需要产生的随机数及其挖矿时间，如表1所示

表 1 PoW实验结果统计表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Diff=1** | | | | | | |
| **Nonce** | 15 | 0 | 29 | 1 | 2 | 9.4 |
| **time(ms)** | 0.018846 | 0.004155 | 0.097186 | 0.004681 | 0.007252 | 0.026424 |
| **Diff=2** | | | | | | |
| **Nonce** | 87 | 54 | 85 | 269 | 122 | 123.4 |
| **time(ms)** | 0.118972 | 0.124993 | 0.11165 | 0.346396 | 0.139957 | 0.168394 |
| **Diff=3** | | | | | | |
| **Nonce** | 1001 | 9318 | 8714 | 1602 | 2662 | 4659.4 |
| **time(ms)** | 1.07645 | 16.57436 | 9.748008 | 1.94594 | 3.144281 | 6.497809 |
| **Diff=4** | | | | | | |
| **Nonce** | 5258 | 27081 | 62372 | 2670 | 30767 | 25629.6 |
| **time(ms)** | 6.075431 | 32.35095 | 97.99706 | 3.101373 | 36.38205 | 35.18137 |
| **Diff=5** | | | | | | |
| **Nonce** | 21314 | 696049 | 1701734 | 1843505 | 847756 | 1022072 |
| **time(ms)** | 25.84754 | 767.6554 | 1870.528 | 2009.52 | 911.1075 | 1116.932 |
| **Diff=6** | | | | | | |
| **Nonce** | 1438353 | 9303623 | 1730020 | 3267361 | 14258303 | 5999532 |
| **time(ms)** | 1652.828 | 10347.14 | 1881.936 | 3538.42 | 16714.46 | 6826.957 |

根据实验结果绘制了计算解谜时间，即需要产生的随机数大小和证明难度的关系，从图5可以看出是成指数相关。

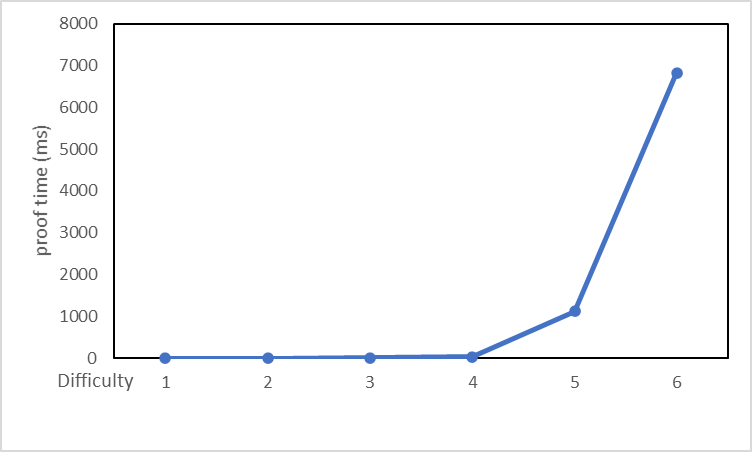


图 5 PoW时间-难度关系图

Figure 5 The relation between PoW difficulty and proof time

1.5.2 PoS

PoS算法进行多组测试后实验结果如表2所示:

表 2 PoS实验结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **blockIndex** | **realDif** | **lshBit** | **nonce** | **time** | **coin.num** | **coin.owner** |
| 0 | 2 | 253 | 32 | 0.107185 | 2 | 2 |
| 1 | 2 | 253 | 11 | 0.273047 | 2 | 2 |
| 2 | 5 | 250 | 9 | 0.136298 | 1 | 2 |
| 3 | 7 | 248 | 107 | 0.82266 | 5 | 2 |
| 4 | 2 | 253 | 5 | 0.133998 | 4 | 1 |
| 5 | 6 | 249 | 166 | 1.425672 | 4 | 1 |
| 6 | 4 | 251 | 16 | 0.040308 | 5 | 1 |
| 7 | 4 | 251 | 81 | 0.61426 | 3 | 1 |
| 8 | 11 | 244 | 2025 | 10.30681 | 3 | 1 |
| 9 | 15 | 240 | 40764 | 117.267 | 1 | 1 |
| 10 | 4 | 251 | 180 | 1.292489 | 2 | 2 |
| 11 | 6 | 216 | 249 | 1.442438 | 4 | 2 |

1.5.3 DPoS

分别在不同的参数设定下（表3），代理人节点数（delegateNode），参与竞选节点数（campaignNode），参与投票节点数（voteNode）交叉进行试验。

表 3 DPoS测试参数表

|  |  |  |
| --- | --- | --- |
| **delegateNode数** | **campaignNode数** | **voteNode数** |
| 3 | 10 | 100 |
| 6 | 20 | 200 |
| 9 | 30 | 300 |
| 12 | 40 | 400 |
| 15 | 50 | 500 |

**1.6 总结**

通过对三种共识算法：PoW、PoS和DPoS的实现，对三种算法的原理与流程有了更深的认识，为了突出POX算法的特性，三种POX共识算法与其它几种主流的共识协议的通用属性对比如下所示：

Table 4 Main consensus protocols comparison[8]

表4 POX共识算法和其它共识协议的对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 性质 | PoW | PoS | DPoS | PBFT[9] | RAFT[10, 11] |
| 类型 | Probabilistic-finality | Probabilistic-finality | Probabilistic-finality | Absolute-finality | Absolute-finality |
| 容错 | 50% | 50% | 50% | 33% | 20% |
| 功耗 | 大 | 较小 | 较小 | 非常小 | 非常小 |
| 可扩展性 | 较好 | 较好 | 较好 | 较弱 | 较弱 |
| 应用场景 | 公开 | 公开 | 公开 | 经许可的 | 经许可的 |
| 实例 | Bitcoin | PIVX, NavCoin,  Startis | Ethereum | Hyperledger Fabric | Etcd,  Tidb/TiKV |

通过自己实验总结验证的一些具体属性对比如表2所示：

Table 5 Comparisons between PoW, PoS, and their Hybrid form

表 5 三种POX共识算法对比

|  |  |  |  |
| --- | --- | --- | --- |
| **对比标准** | **PoW** | **PoS** | **DPoS** |
| Energy efficiency | No | Yes | Yes |
| Forking | 当两个节点同时找到合适的随机数时 | 很困难 | 基本不会 |
| Block creating speed | 低，取决于变量 | 快 | 快 |
| Double spending  attack | 无法避免 | 很难发生 | 很难发生 |
| Pool mining | 可以避免 | 很难避免 | 很难避免 |

**参考文献**

[1] SHARMA K, JAIN D. Consensus Algorithms in Blockchain Technology: A Survey[C]//2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2019:1-7.

[2] BACK A. Hashcash-a denial of service counter-measure[J]. 2002.

[3] GARCIA RIBERA E. Design and Implementation of a Proof-of-Stake Consensus Algorithm for Blockchain[D]. Universitat Politècnica de Catalunya, 2018.

[4] KING S, NADAL S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[J]. self-published paper, August, 2012,19:1.

[5] MINGXIAO D, XIAOFENG M, ZHE Z, et al. A review on consensus algorithm of blockchain[C]//2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2017:2567-2572.

[6] ANDREY A, PETR C. Review of Existing Consensus Algorithms Blockchain[C]//2019 International Conference" Quality Management, Transport and Information Security, Information Technologies"(IT&QM&IS). IEEE, 2019:124-127.

[7] LARIMER D. Delegated proof-of-stake (dpos)[J]. Bitshare whitepaper, 2014.

[8] ZHANG S, LEE J. Analysis of the main consensus protocols of blockchain[J]. ICT Express, 2020,6(2):93-97.

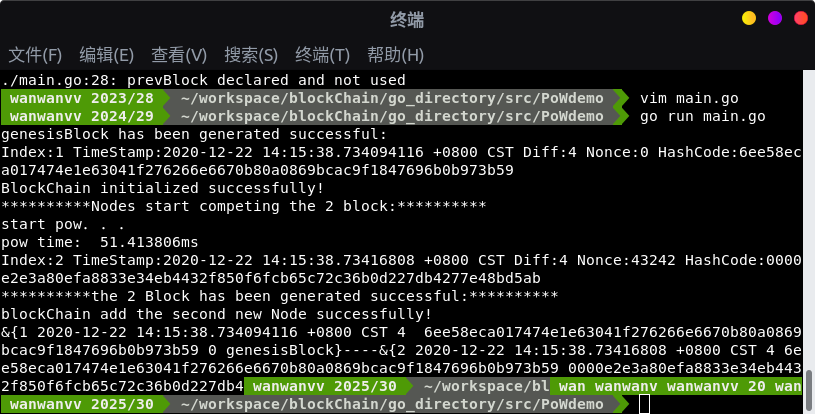
[9] CASTRO M, LISKOV B. Practical Byzantine fault tolerance[C]//OSDI. 1999:173-186.

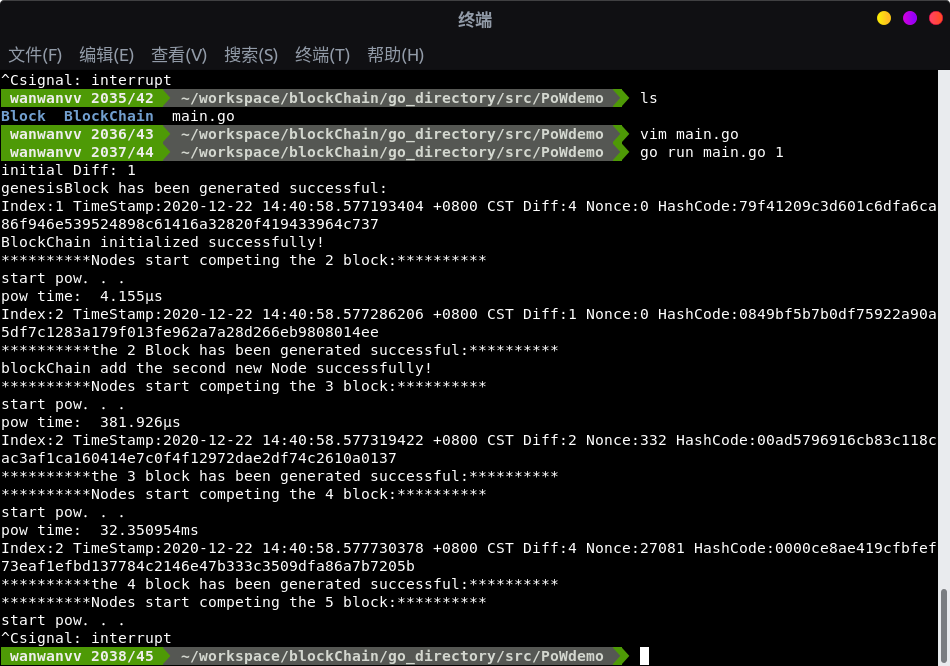
[10] LAMPORT L. Paxos made simple[J]. ACM Sigact News, 2001,32(4):18-25.

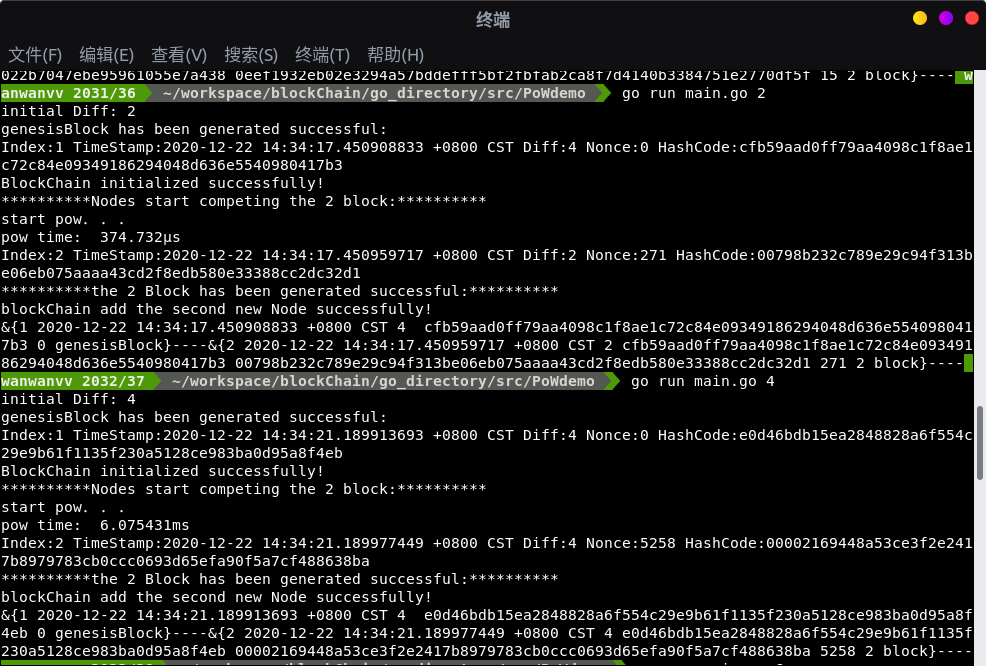
[11] LAMPORT L. The part-time parliament[M]//Concurrency: the Works of Leslie Lamport. 2019:277-317.

**Appendix**

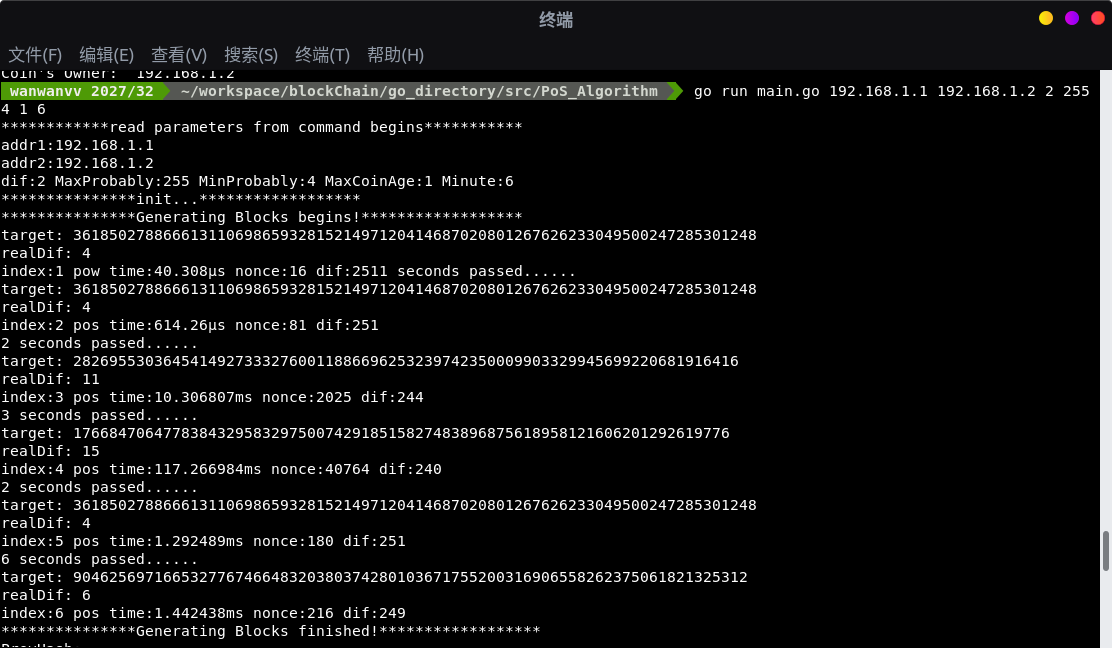
A. PoW算法运行截图

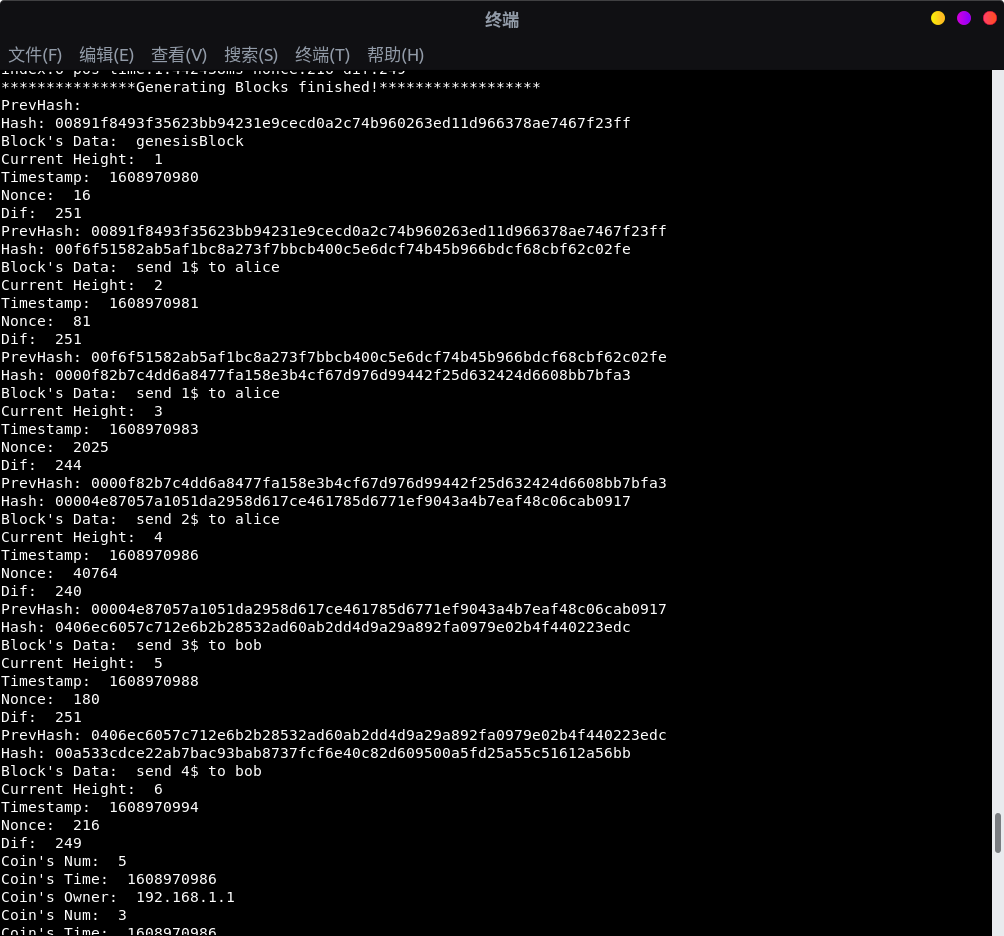






B. PoS算法运行截图







C. DPoS算法运行截图



