Blockchain Technology

2023.03.07

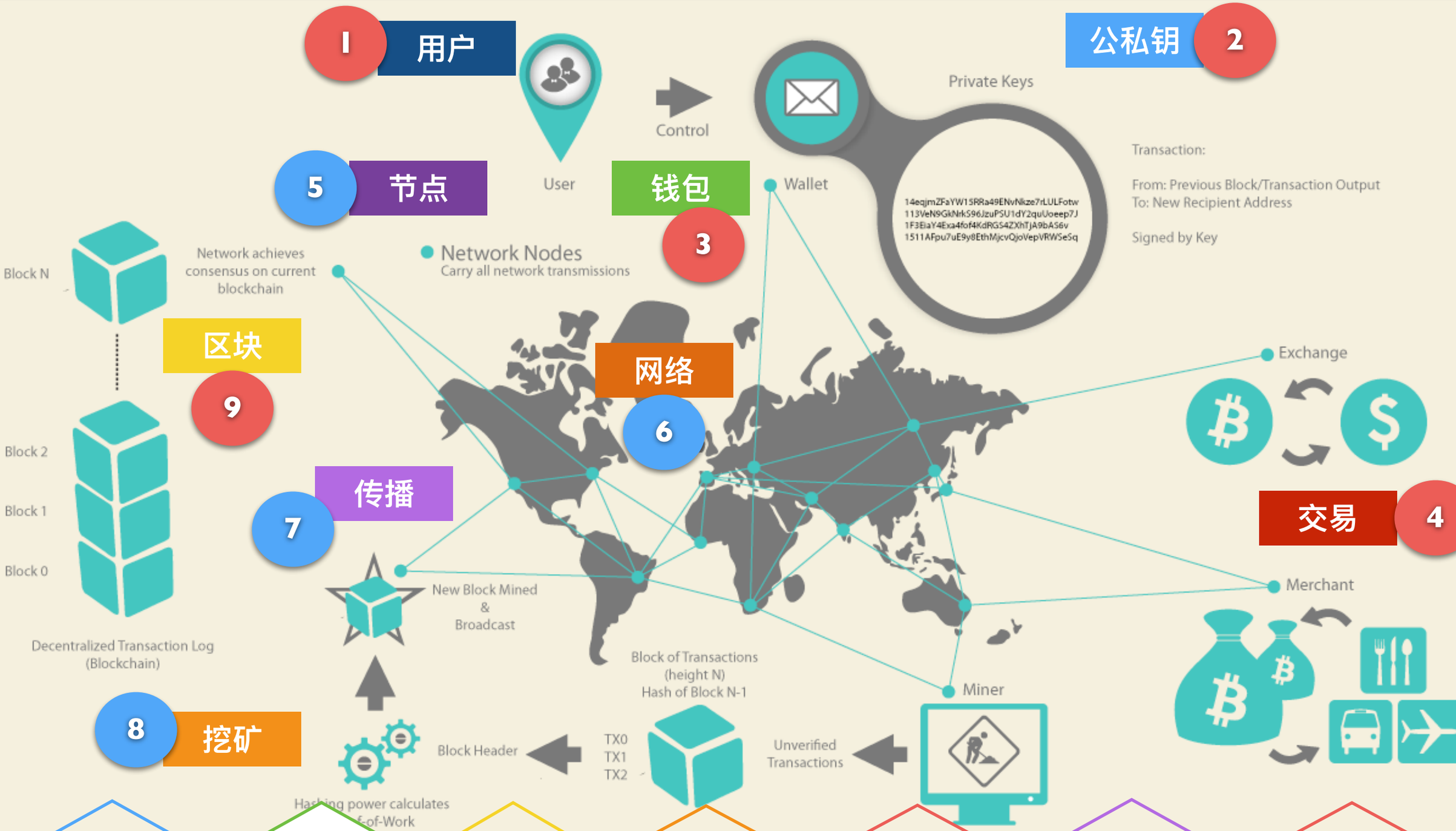
# 比特币-02


北京大学 软件与微电子学院
School of Software and Microelectronics, Peking University

Huiping Sun(孙惠平)
sunhp@ss.pku.edu.cn

Bitcoin

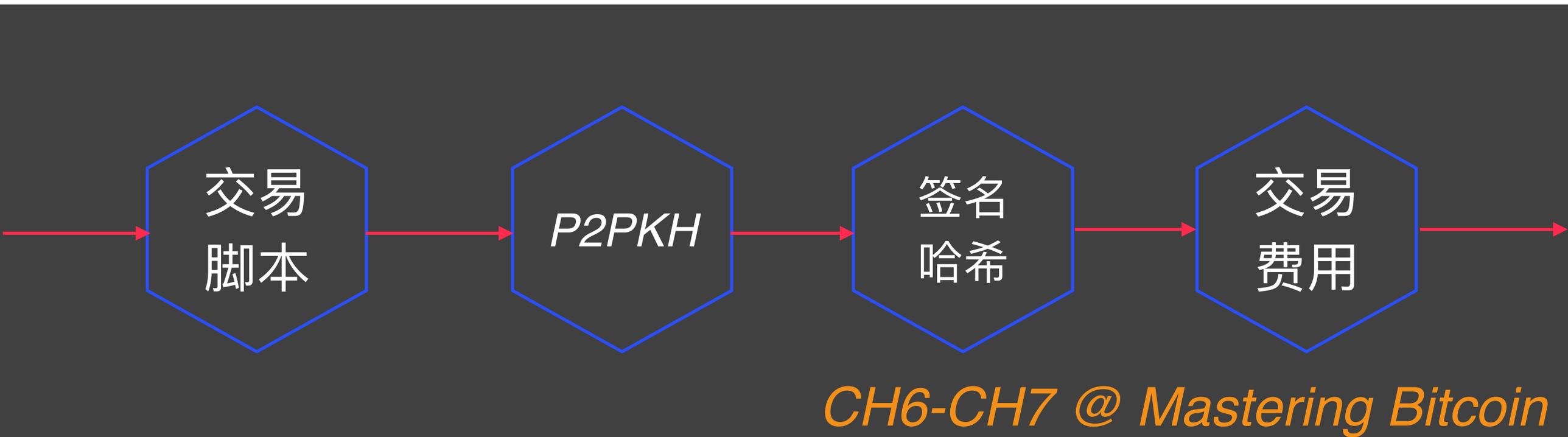
# 上次课程内容回顾




简介 地址 密码 钱包 交易 UTXO 区块

# 本次课程内容

# 脚本




*CH6-CH7 @ Mastering Bitcoin*

# 比特币交易脚本

```
{
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid":"7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
      "vout": 0,
      "scriptSig":
      "3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb
      02204b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
      0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376
      789d172787ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 0.01500000,
      "scriptPubKey": "OP_DUP OP_HASH160
      ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": 0.08450000,
      "scriptPubKey": "OP_DUP OP_HASH160
      7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
    }
  ]
}
```

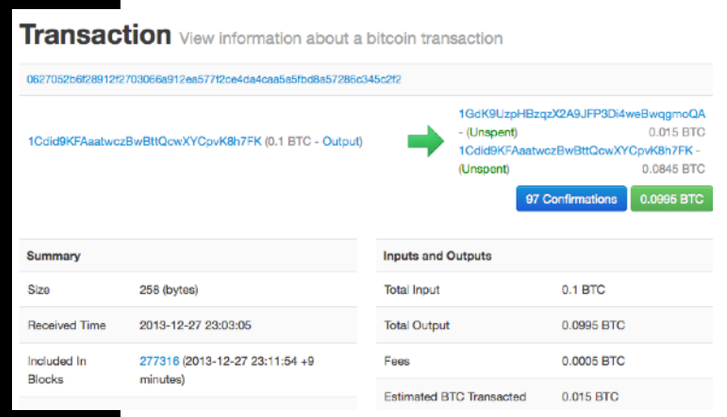
解锁脚本

锁定脚本

Scripts

图灵不完备

无状态验证




**OP_DUP OP_HASH160 <PubKey> OP_EQUALVERIFY OP_CHECKSIG**

# 比特币脚本

Unlocking Script (scriptSig) + Locking Script (scriptPubKey)

`<sig> <PubK>` | `DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG`

Unlock Script (scriptSig) is provided by the user to resolve the encumbrance

Lock Script (scriptPubKey) is found in a transaction output and is the encumbrance that must be fulfilled to spend the output

**基于堆栈**

**FIFO** **Push** **Pop**

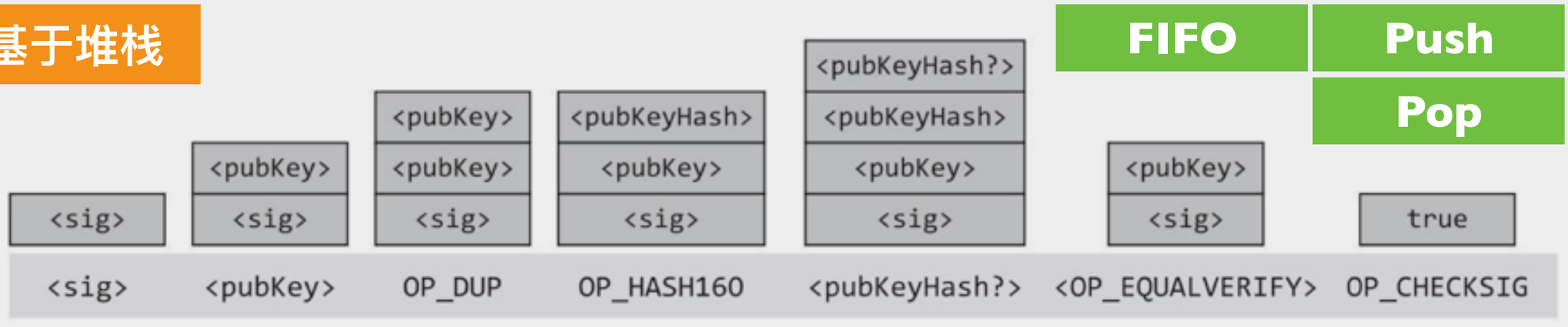
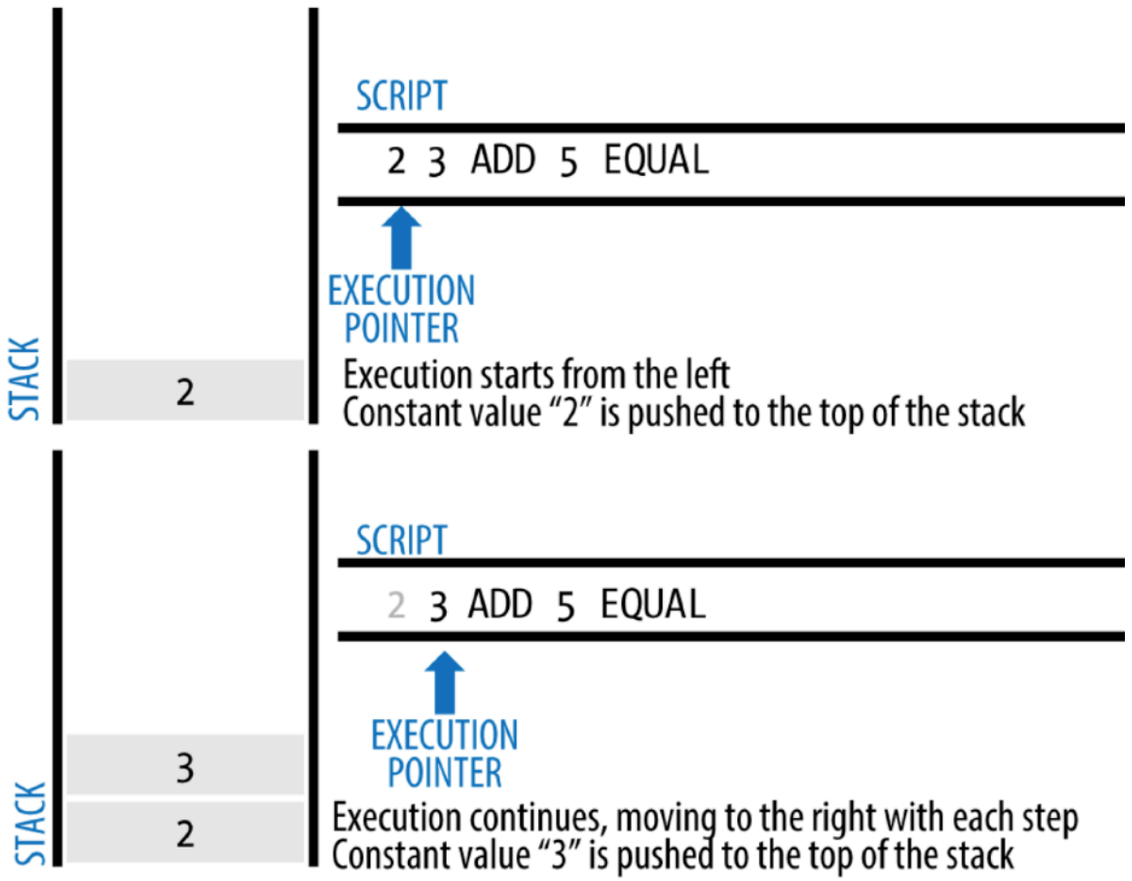


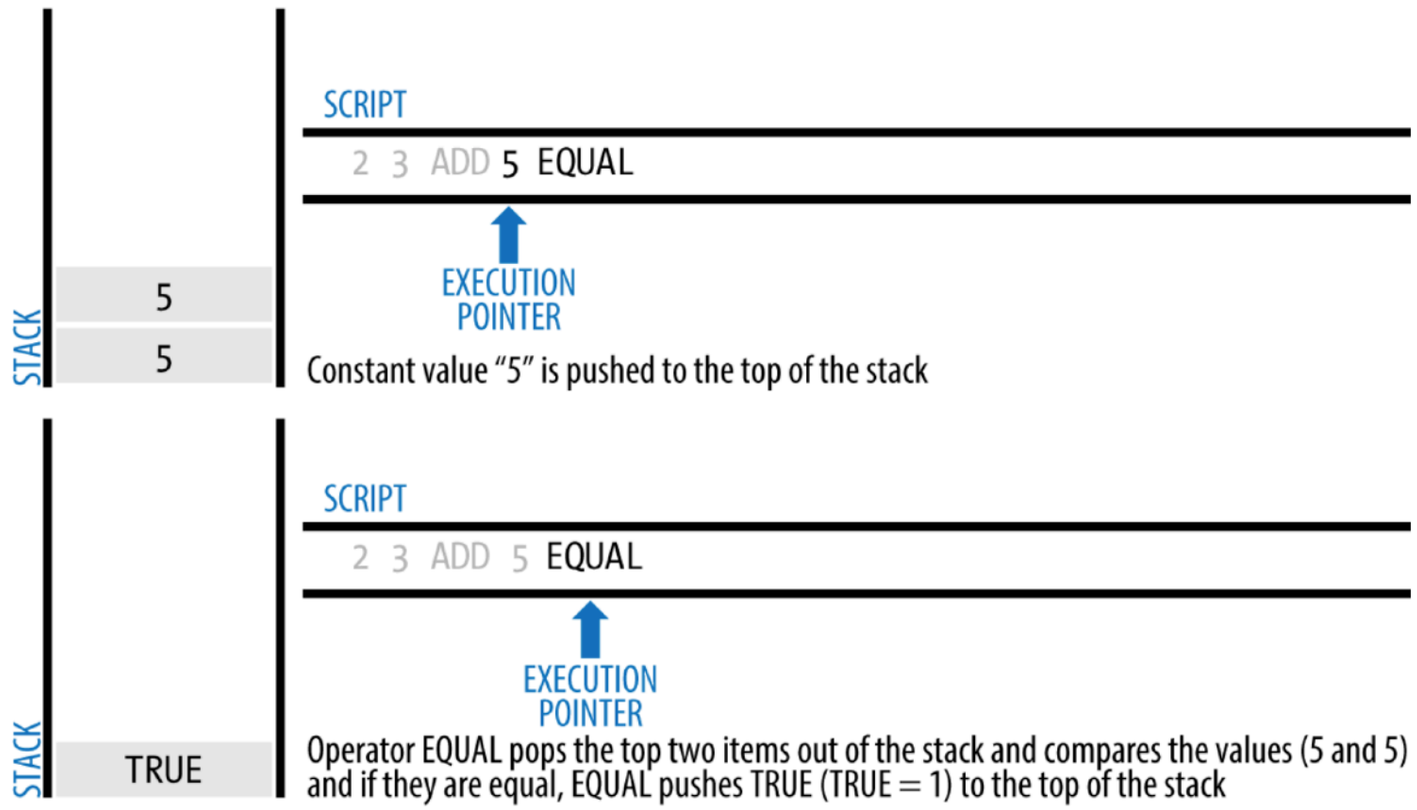

图3.6 比特币脚本的执行堆栈状态图

注：图中底部列出了相对应的指令：尖括号里的是数据指令，以OP开头的是工作码指令，指令上方对应的是指令执行之后的堆栈状态。

# 脚本例子

## 2 3 OPADD 5 OP_EQUAL

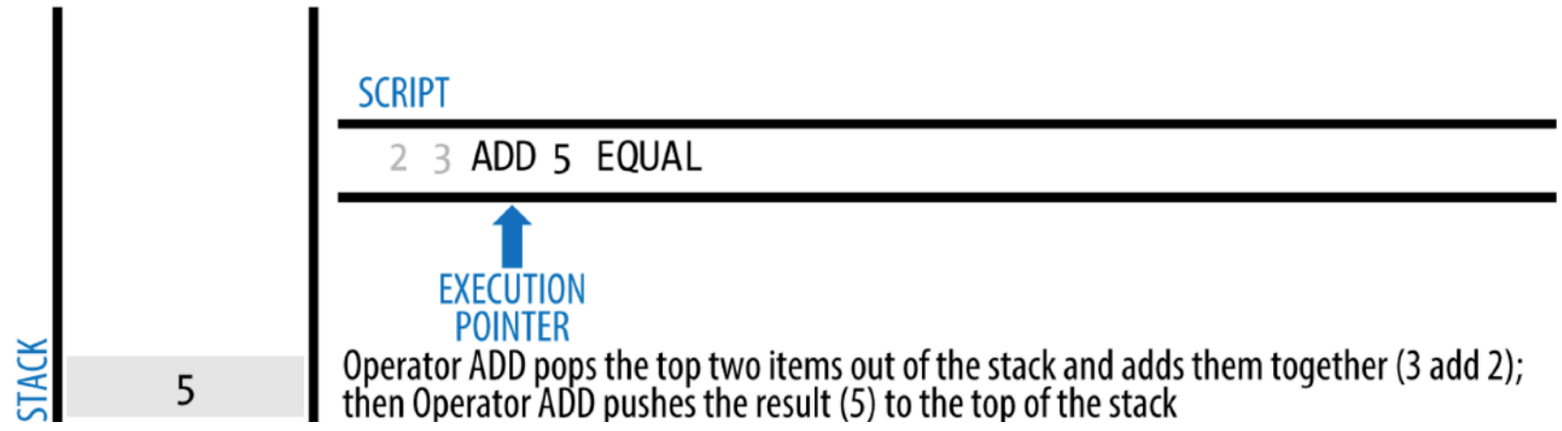




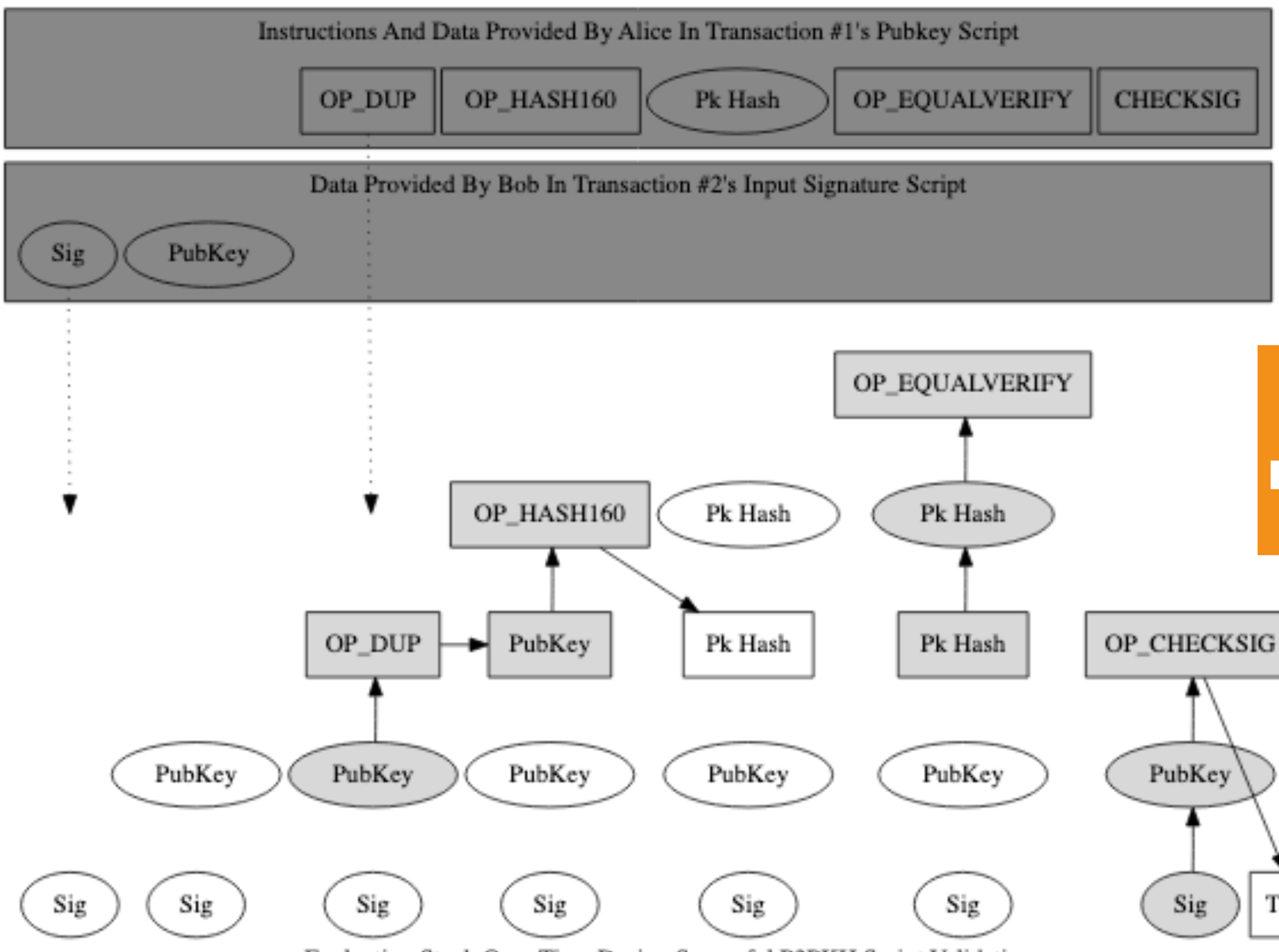




# P2PKH

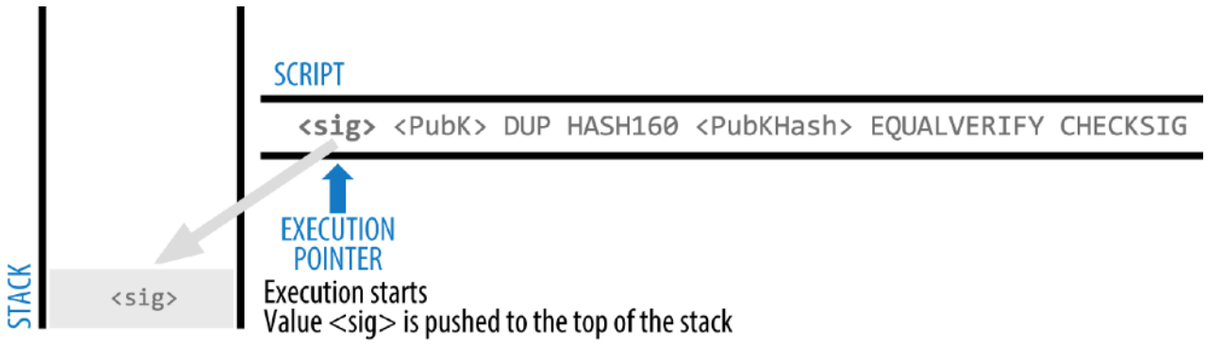
**OP_DUP OP_HASH160 <PubKey> OP_EQUALVERIFY OP_CHECKSIG**

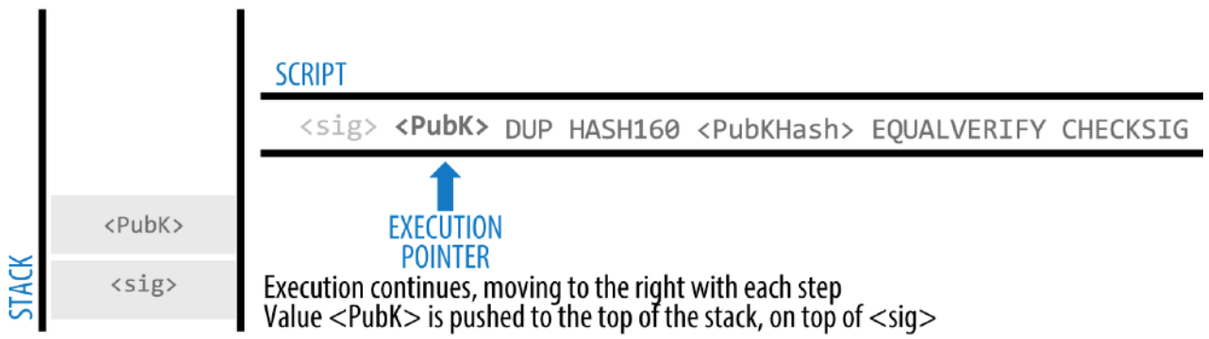


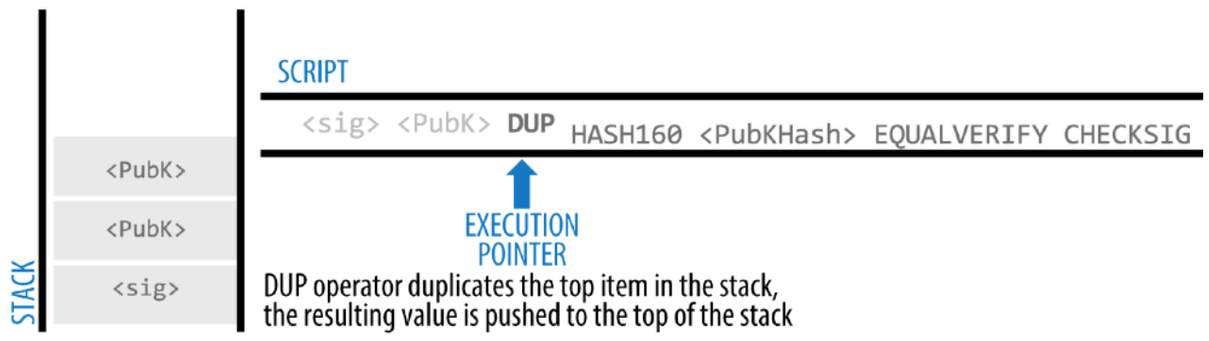

Evaluation Stack Over Time During Succesful P2PKH Script Validation

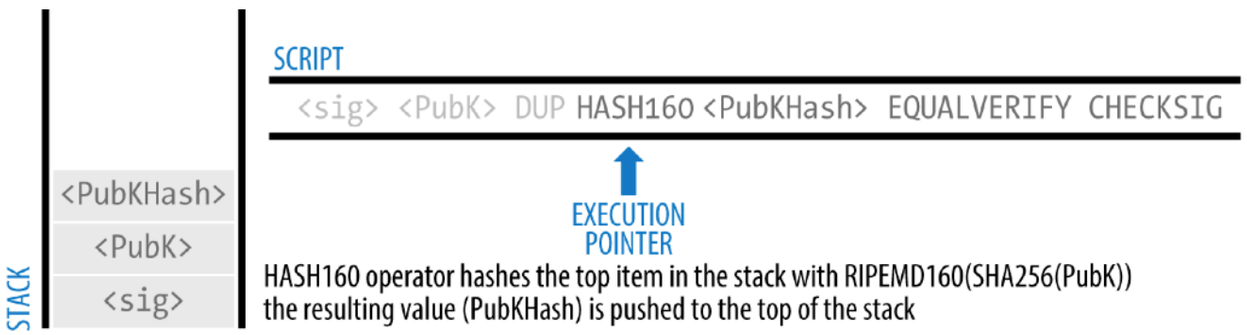
**P2PKH: Pay-to-Public-Key-Hash**

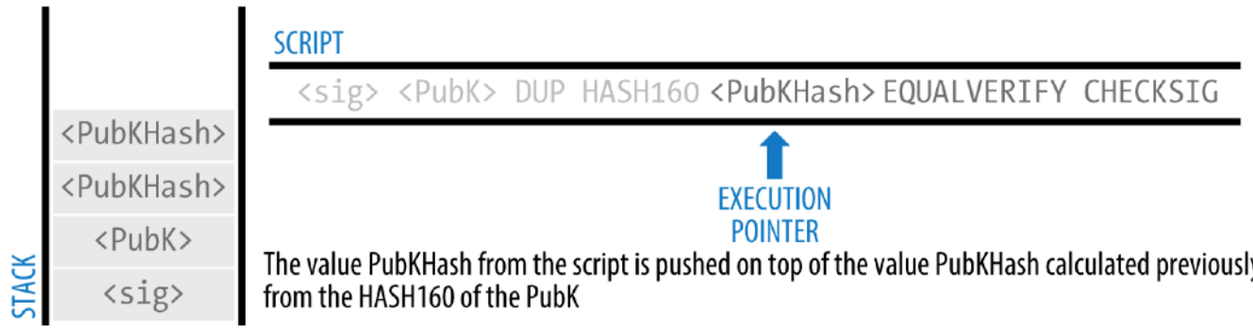
# P2PKH

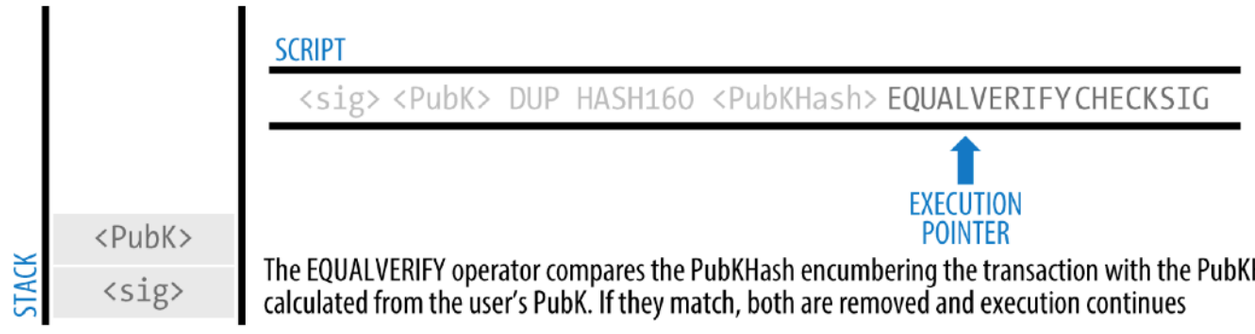

SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<sig>
Execution starts
Value <sig> is pushed to the top of the stack

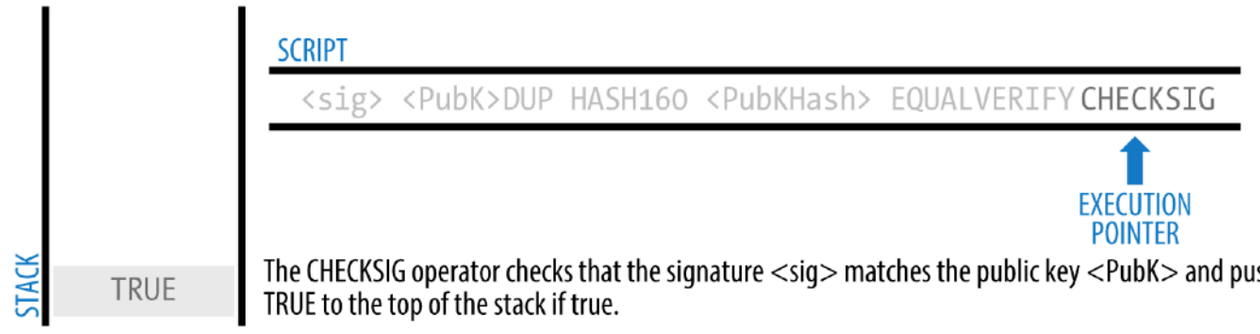

SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<PubK>
<sig>
Execution continues, moving to the right with each step
Value <PubK> is pushed to the top of the stack, on top of <sig>


SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<PubK>
<PubK>
<sig>
DUP operator duplicates the top item in the stack,
the resulting value is pushed to the top of the stack

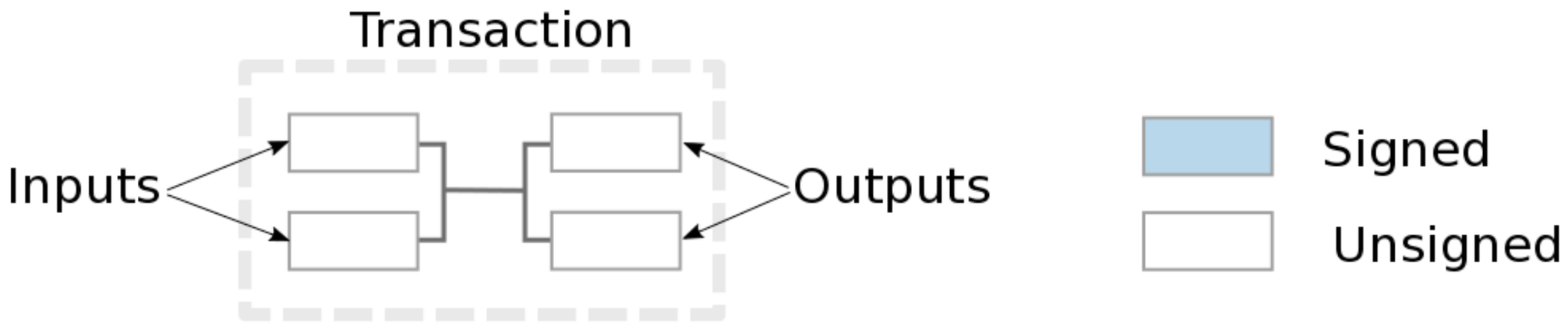

SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<PubKHash>
<PubK>
<sig>
HASH160 operator hashes the top item in the stack with RIPEMD160(SHA256(PubK))
the resulting value (PubKHash) is pushed to the top of the stack

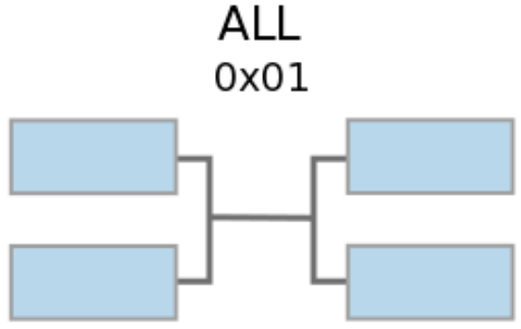

SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<PubKHash>
<PubKHash>
<PubK>
<sig>
The value PubKHash from the script is pushed on top of the value PubKHash calculated previously
from the HASH160 of the PubK


SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<PubK>
<sig>
The EQUALVERIFY operator compares the PubKHash encumbering the transaction with the PubKHash
calculated from the user's PubK. If they match, both are removed and execution continues

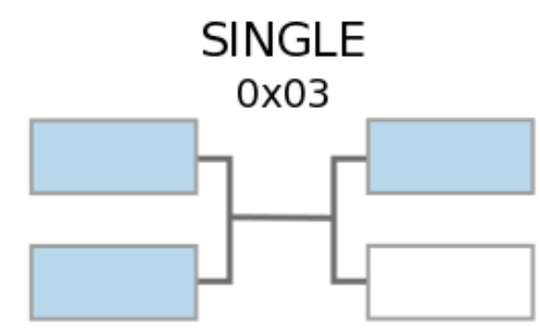

SCRIPT
<sig> <PubK>DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
TRUE
The CHECKSIG operator checks that the signature <sig> matches the public key <PubK> and pushes
TRUE to the top of the stack if true.

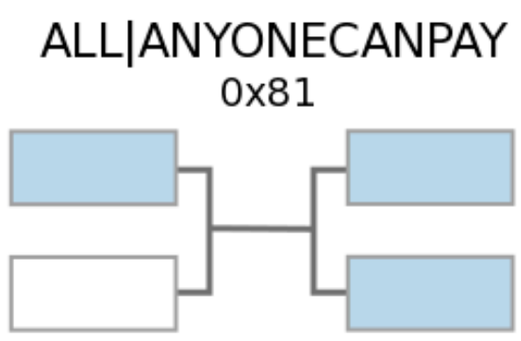
# 签名哈希类型(SIGHASH)



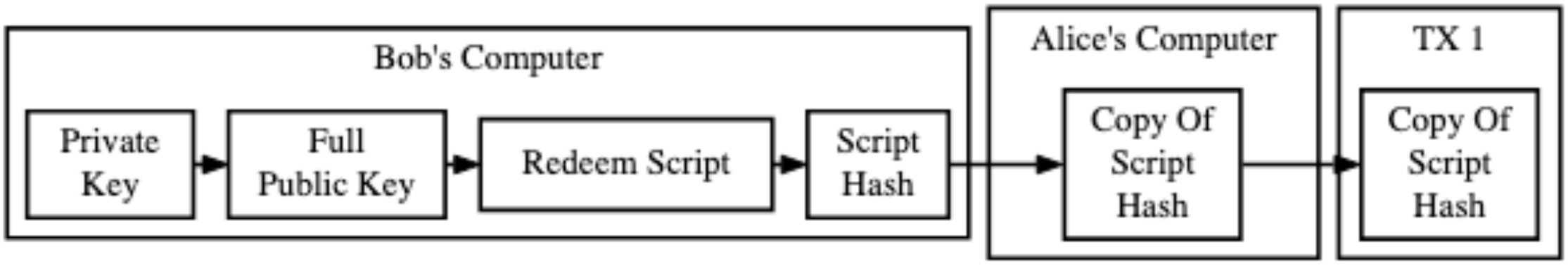




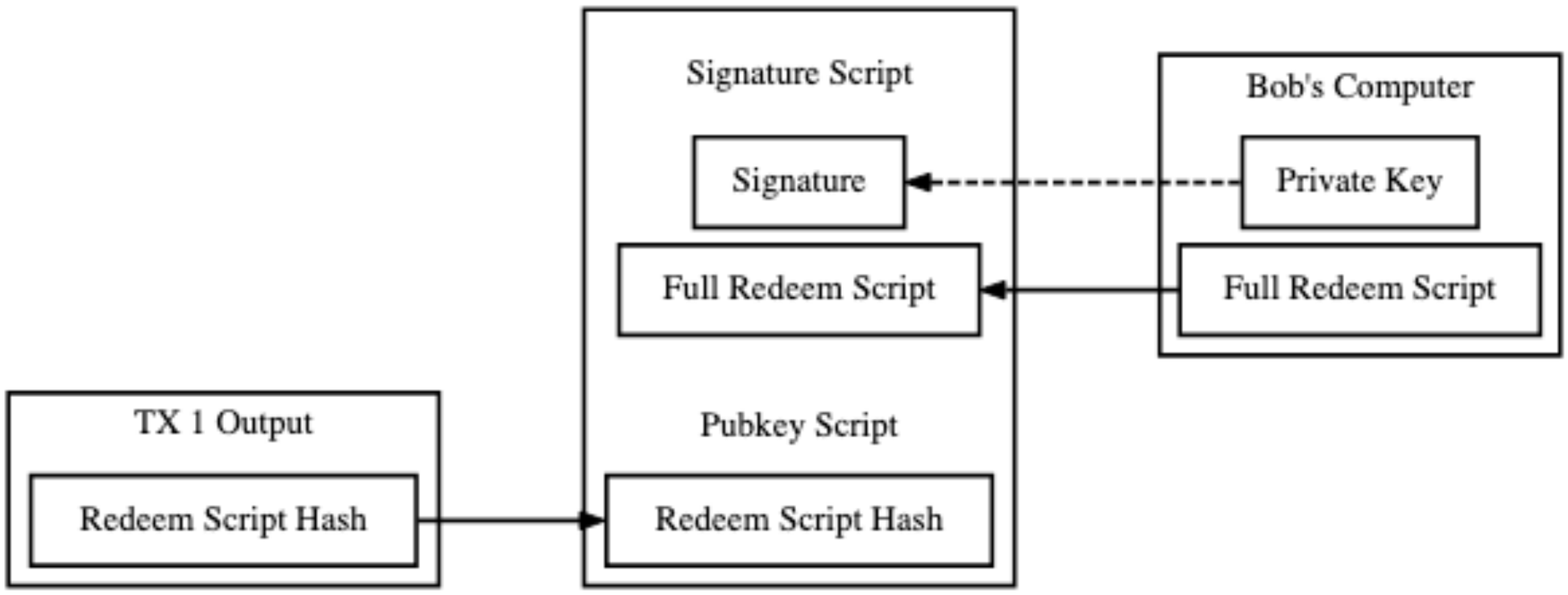





# P2SH




Creating A P2SH Redeem Script Hash To Receive Payment

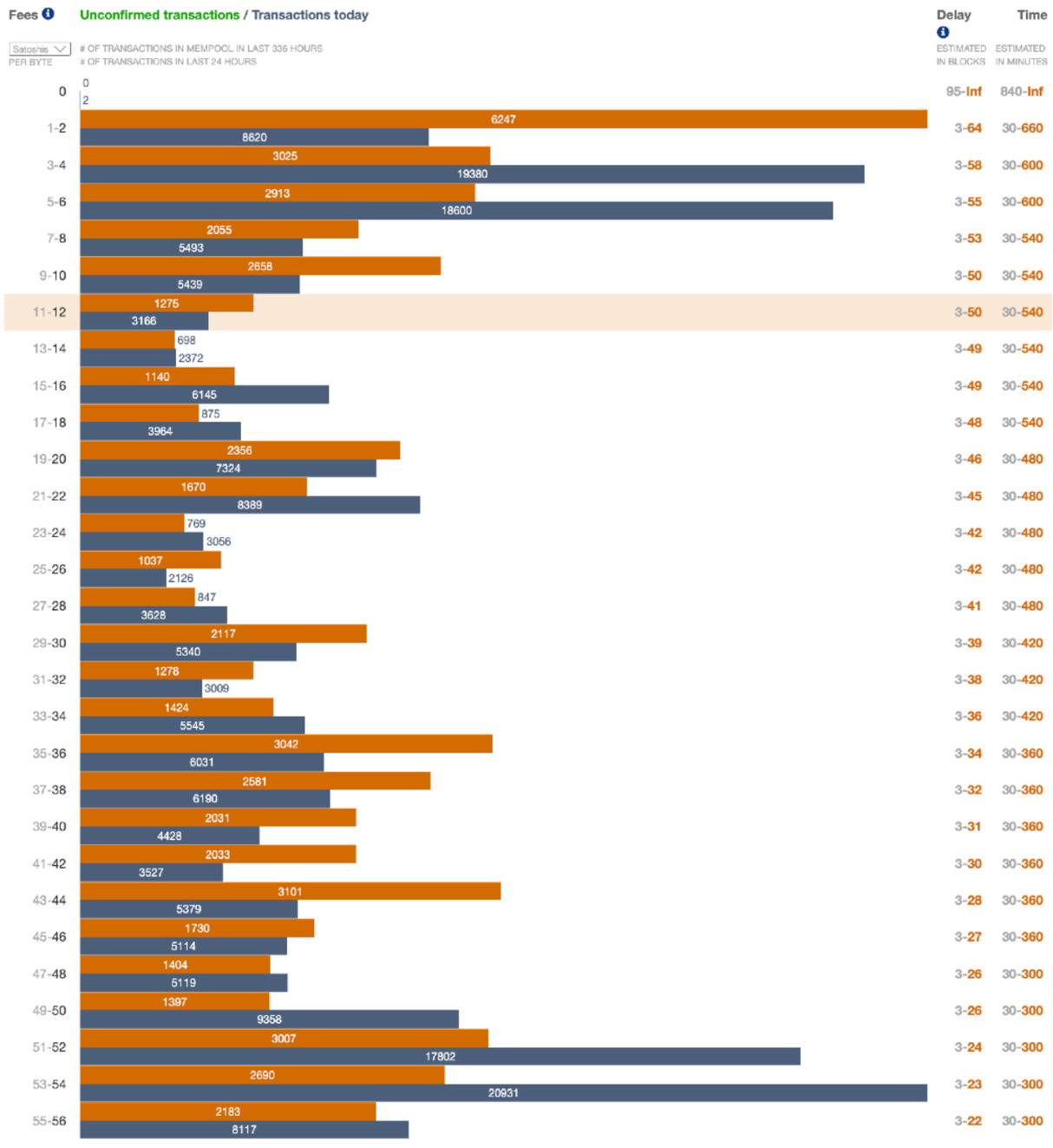


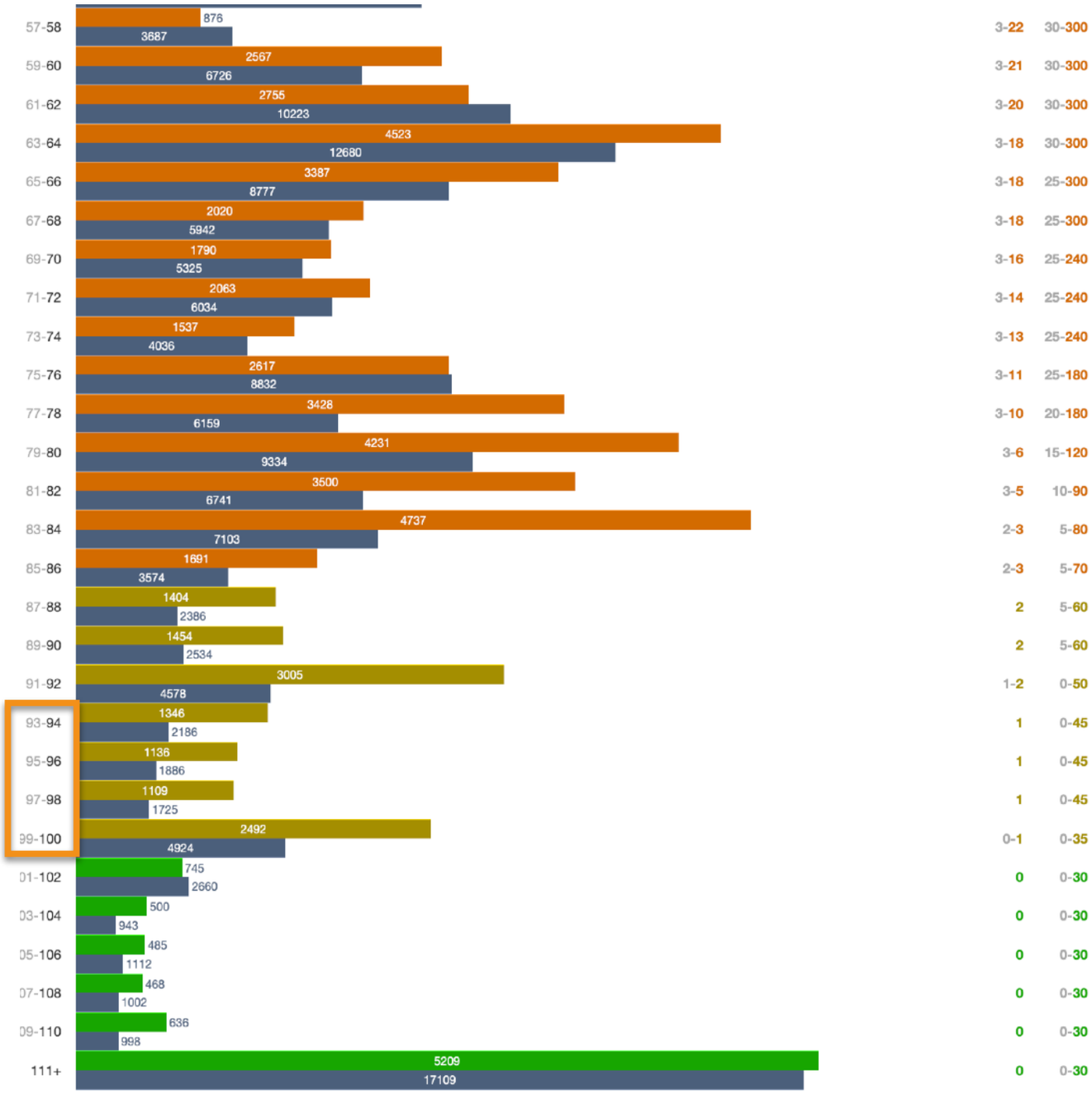

Spending A P2SH Output

# Bitcoin-02

# 交易费用

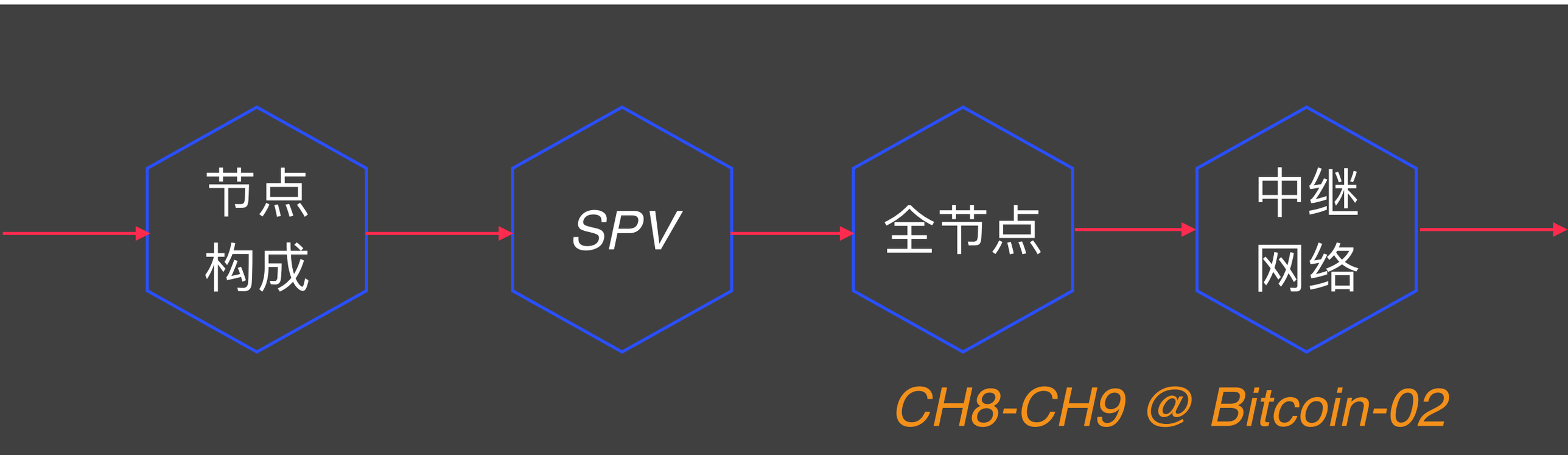
https://bitcoinfees.earn.com/



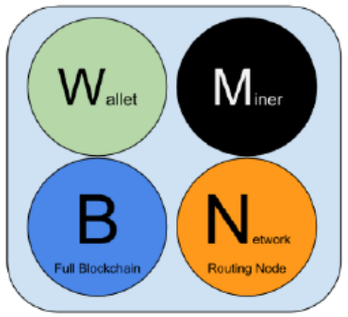




90-100

# 节点

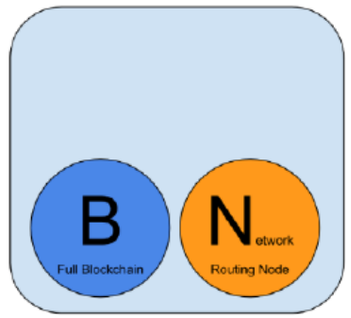


# 节点类型



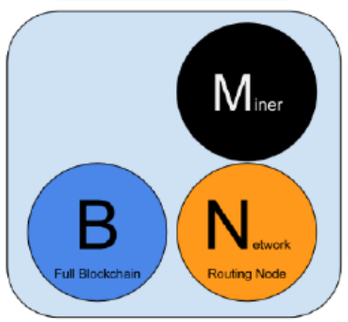

## Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



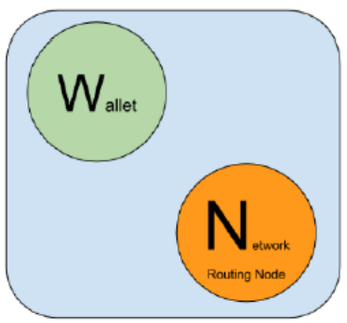

## Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



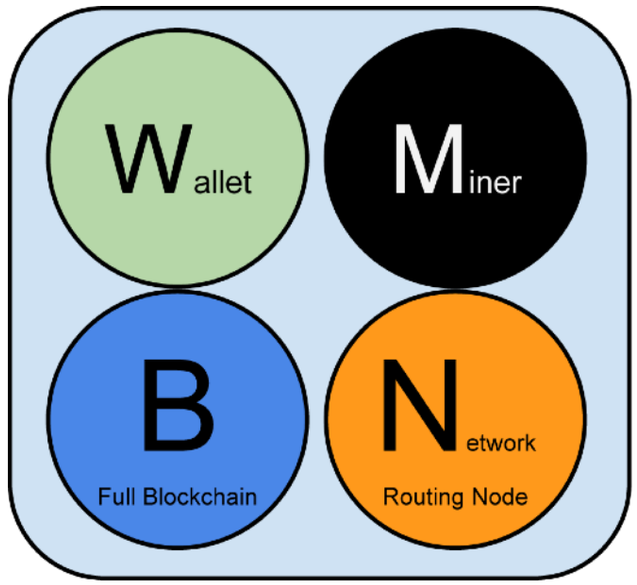

## Solo Miner

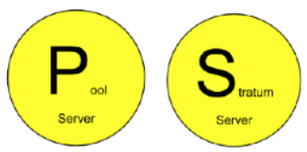
Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.




## Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.



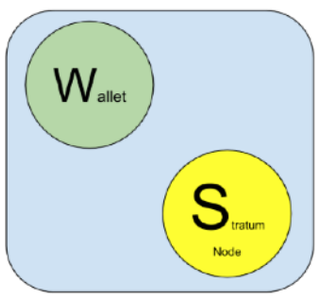




## Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.


## Mining Nodes

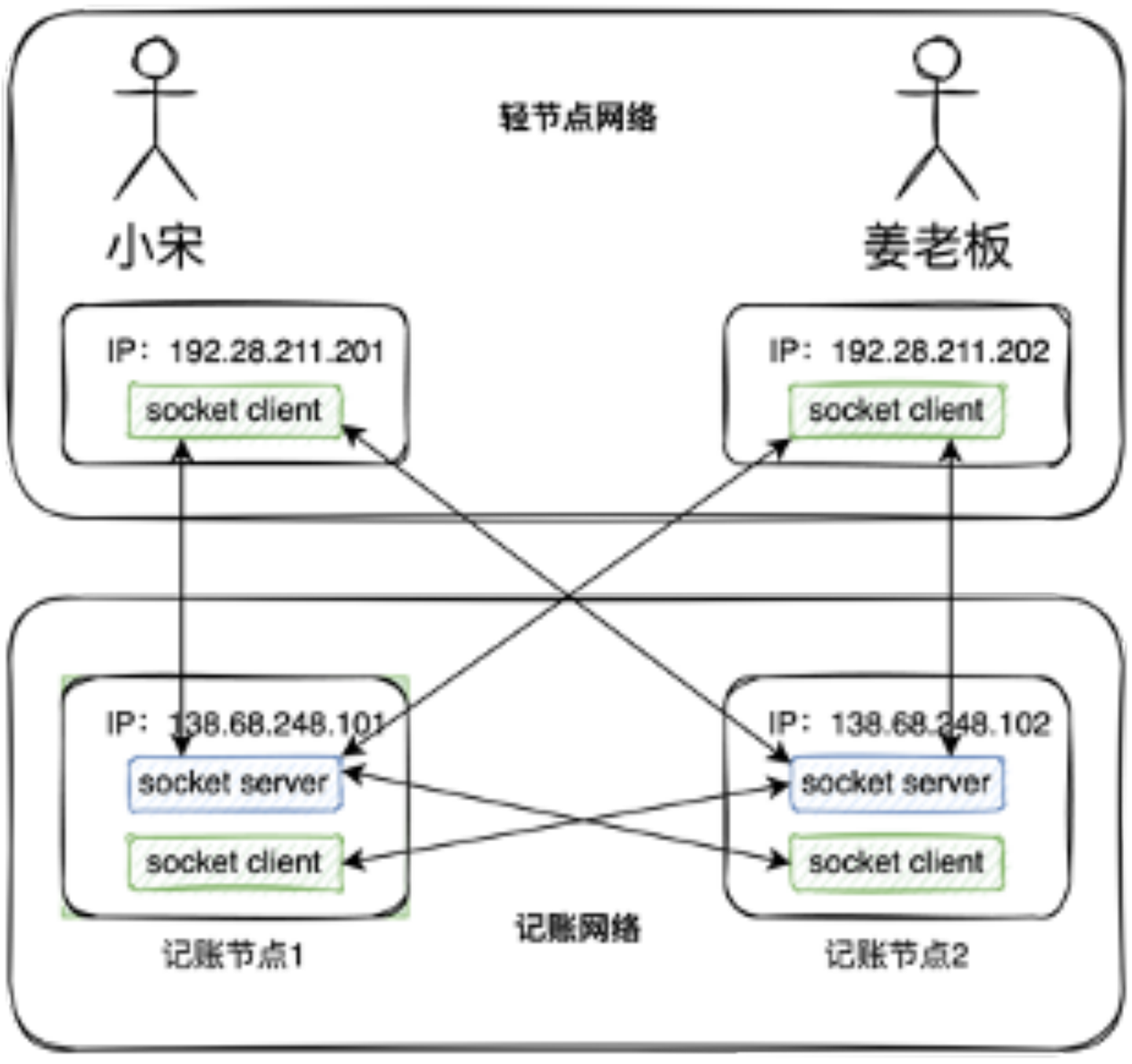
Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.




## Lightweight (SPV) Stratum wallet

Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.

# SPV



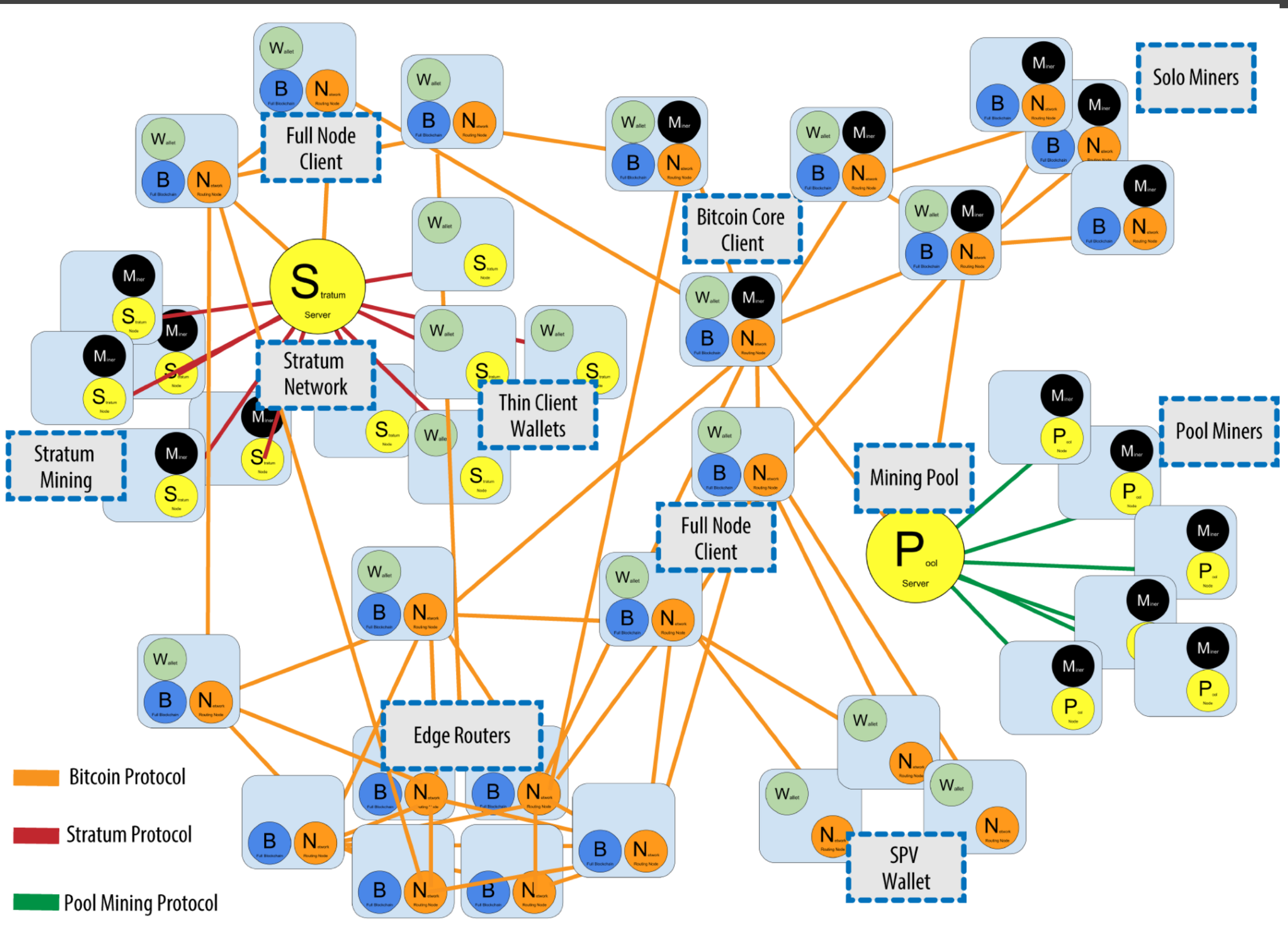

每个节点都参与验证并传播交易及区块信息，发现并维持与对等节点的连接。另外还有一些节点只保留了区块链的一部分，它们通过一种名为“简易支付验证（SPV）”的方式来完成交易验证。这样的节点被称为“SPV节点”，又叫“轻量级节点”。

因此，节点又可以分为“全节点”和“轻节点”。全节点就是拥有全网所有交易数据的节点，轻节点就是只拥有和自己相关的交易数据节点。

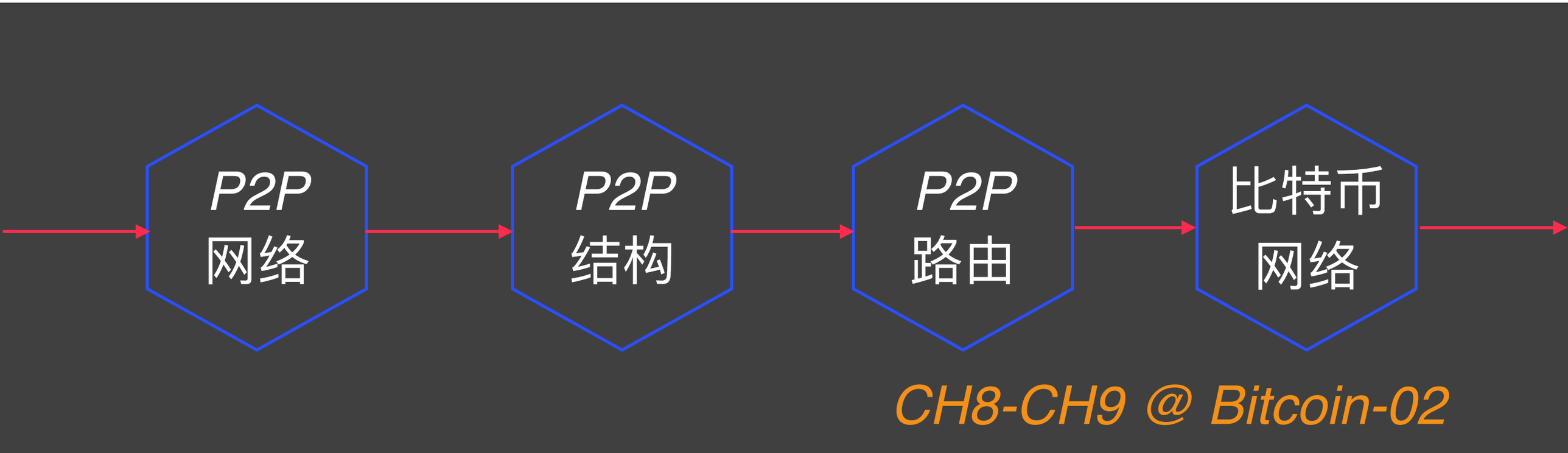
# 全节点 vs. SPV

| 全节点 | 轻节点 |
| --- | --- |
| 一直在线 | 不是一直在线 |
| 在本地硬盘上维护完整区块链信息 | 不保存整个区块链，只需要保存每隔区块块头 |
| 在内存中维护UTXO集合，以便于快速检验交易合法性 | 不保存全部交易，只保存和自己有关的交易 |
| 监听比特币网络中交易内容，验证每个交易合法性 | 无法验证大多数交易合法性，只能检验和自己相关的交易合法性 |
| 决定哪些交易会打包到区块中 | 无法检测网上发布的区块正确性 |
| 监听其他矿工挖出的区块，验证其合法性 | 可以验证挖矿难度 |
| 挖矿：<br>1. 决定沿着哪条链挖下去。<br>2. 当出现等长分叉，选择哪一个分叉 | 只能检测哪个是最长链，不知道哪个是最长合法链 |

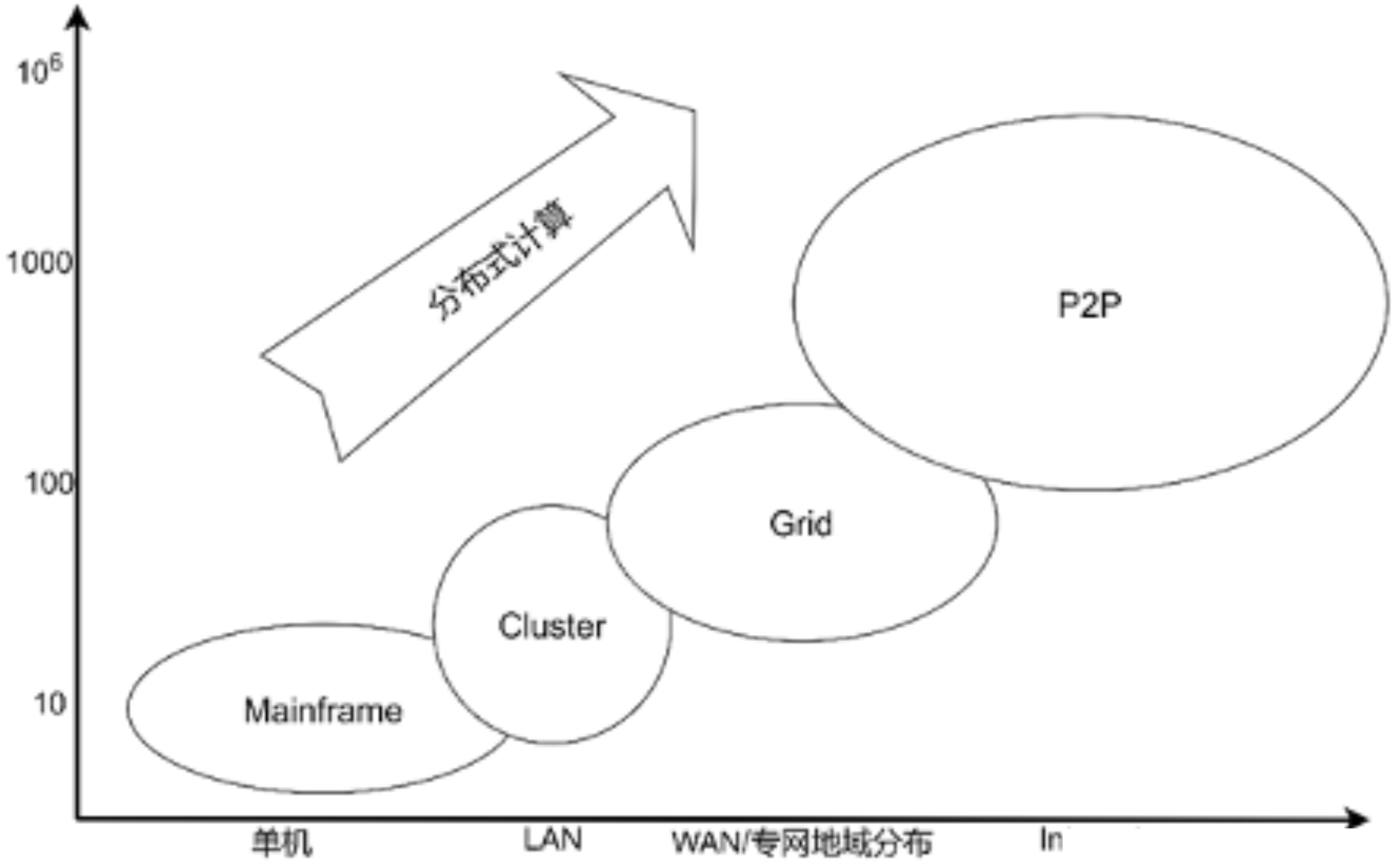
# 中继网络


Solo Miners
Full Node Client
Bitcoin Core Client
Stratum Server
Stratum Network
Thin Client Wallets
Stratum Mining
Pool Miners
Mining Pool
Pool Server
Full Node Client
Edge Routers
SPV Wallet
Bitcoin Protocol
Stratum Protocol
Pool Mining Protocol

# 网络



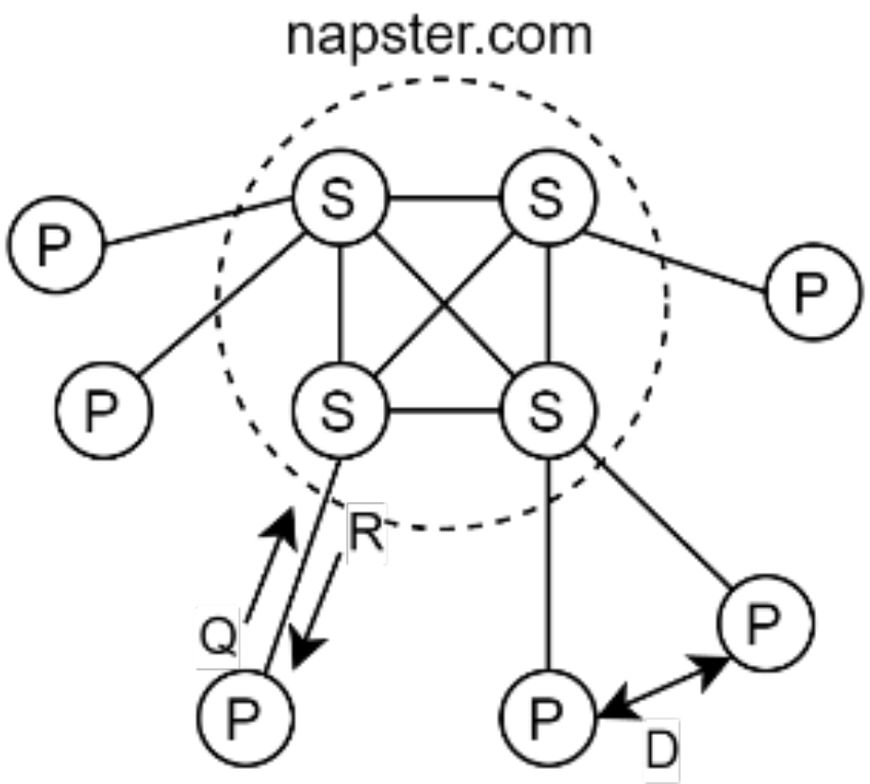
# 对等网

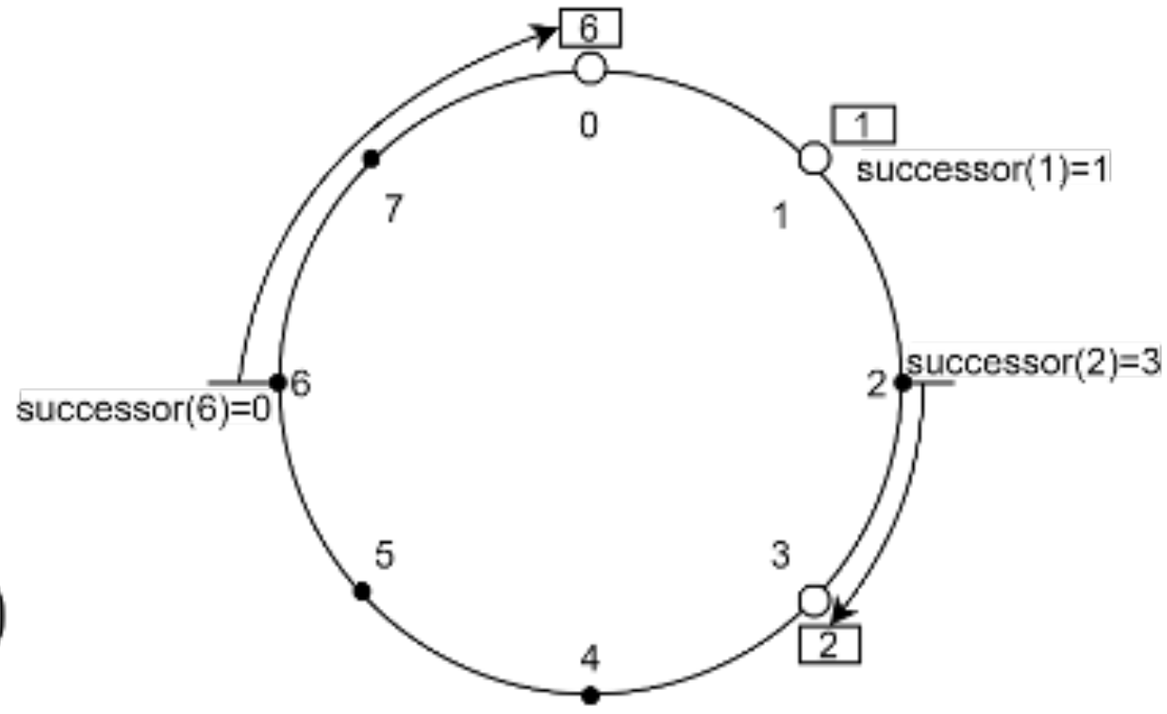

$10^6$
1000
100
10
分布式计算
P2P
Grid
Cluster
Mainframe
单机
LAN
WAN/专网地域分布
In

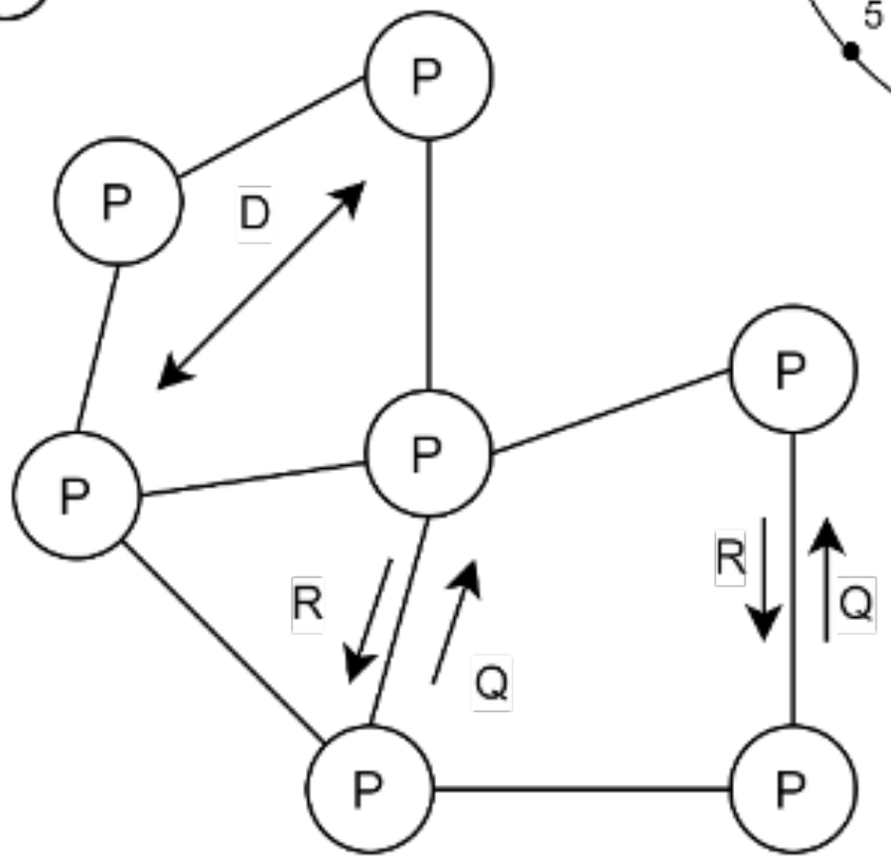

eDonkey


napster
TM

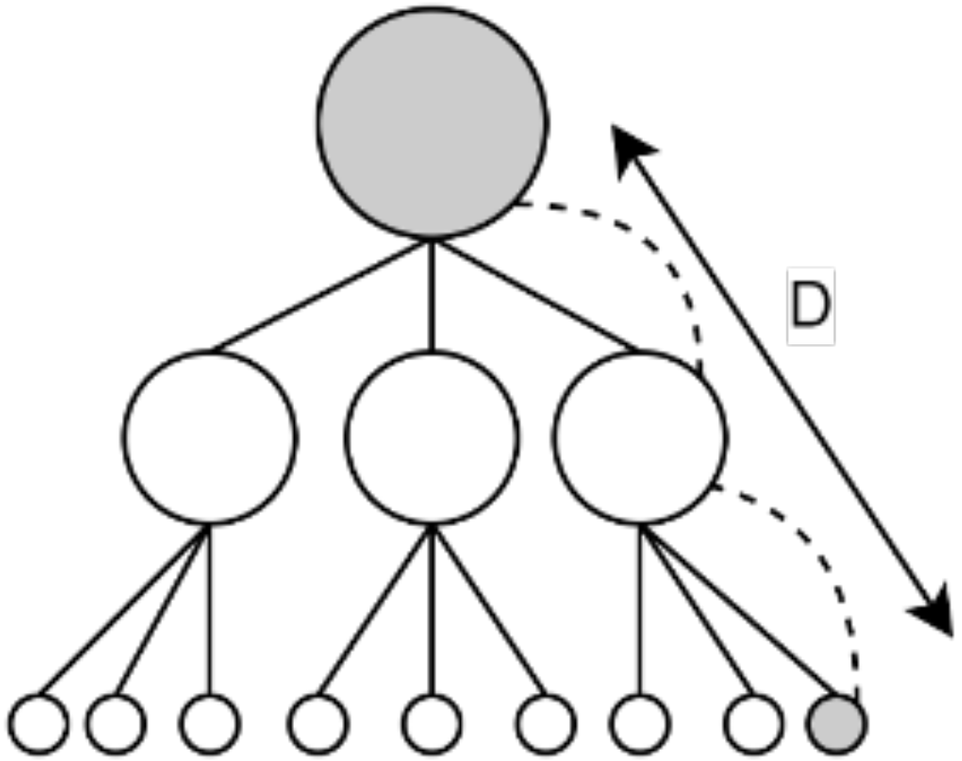
# P2P网络结构

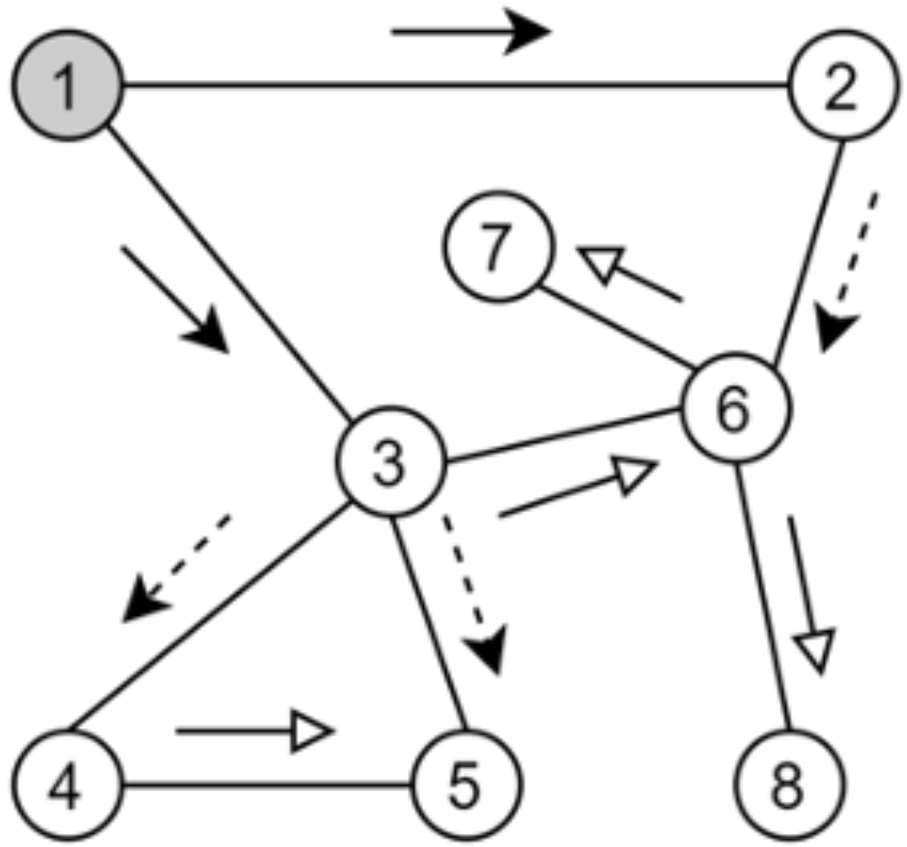

napster.com
S
S
S
S
P
P
P
P
P
P
P
R
Q
D

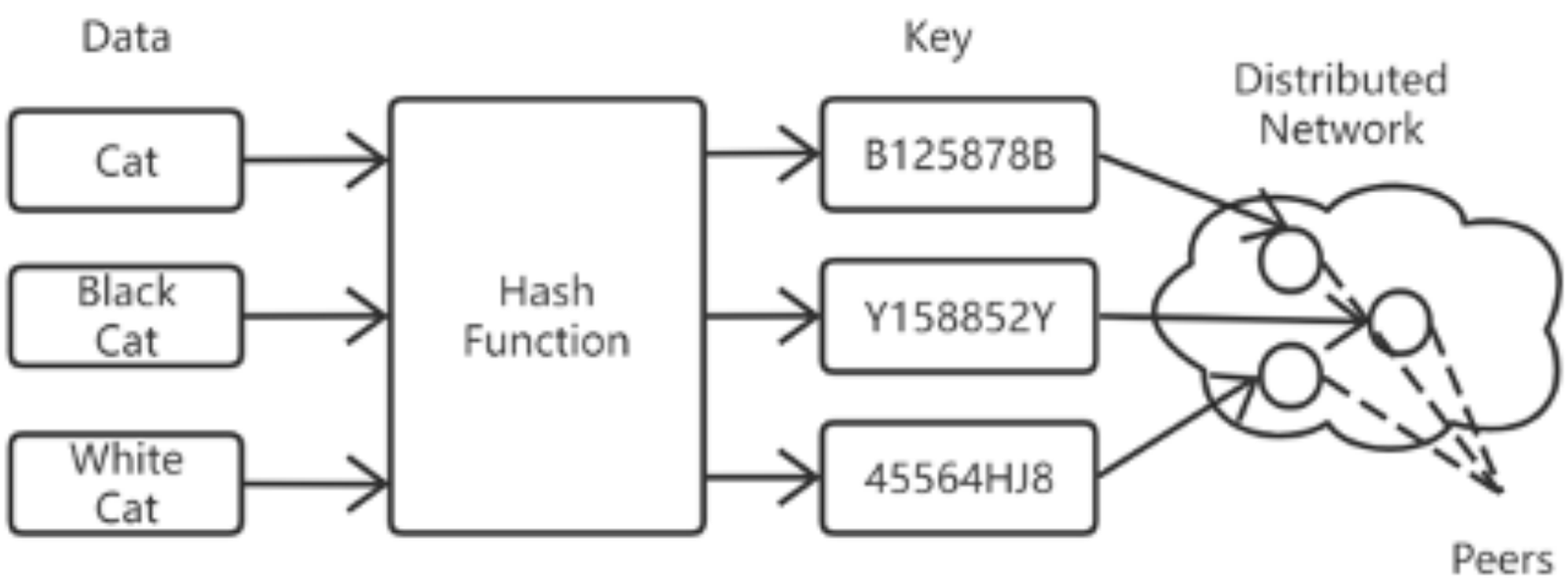

6
0
1
successor(1)=1
7
1
successor(6)=0
6
2
successor(2)=3
5
3
4
2


P
P
P
P
P
P
P
P
D
R
Q
R
Q

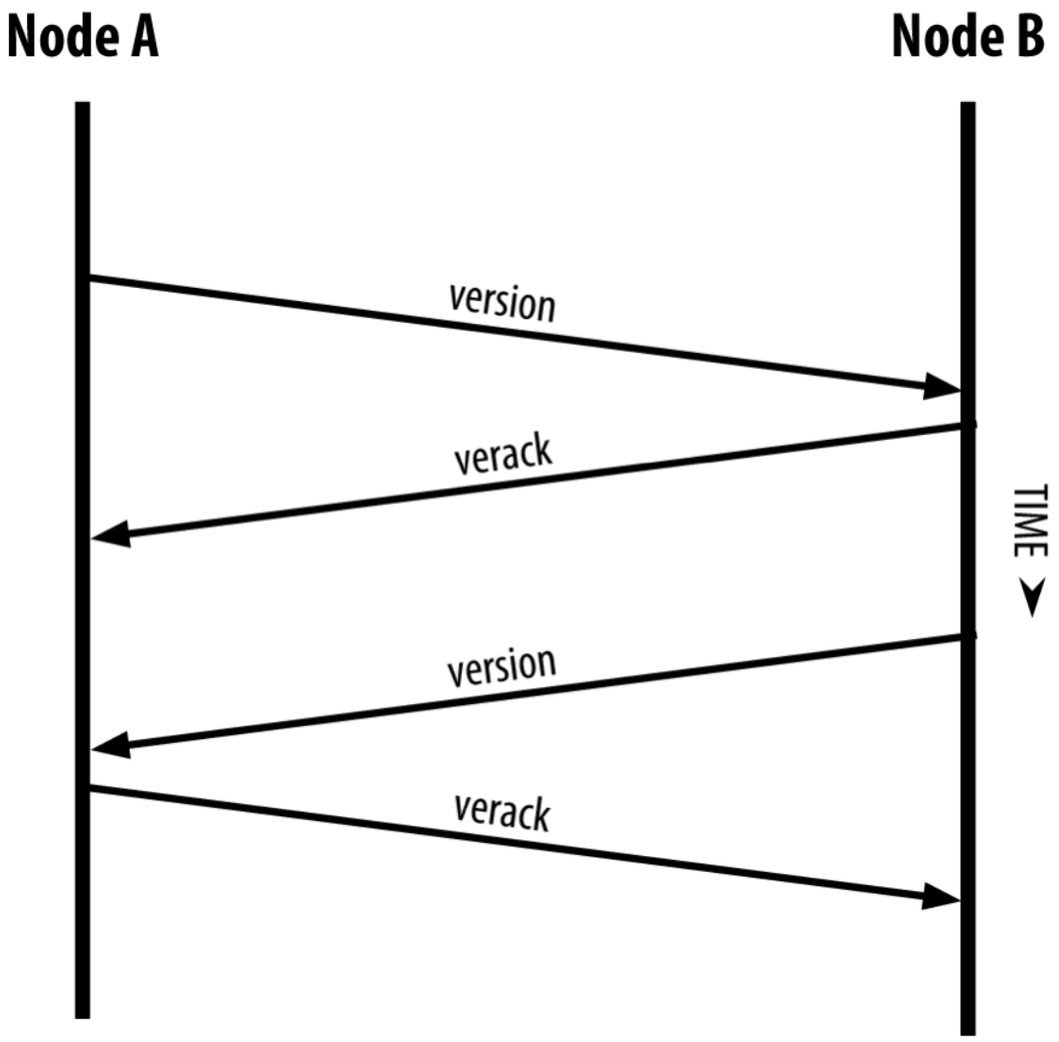
# P2P网络路由

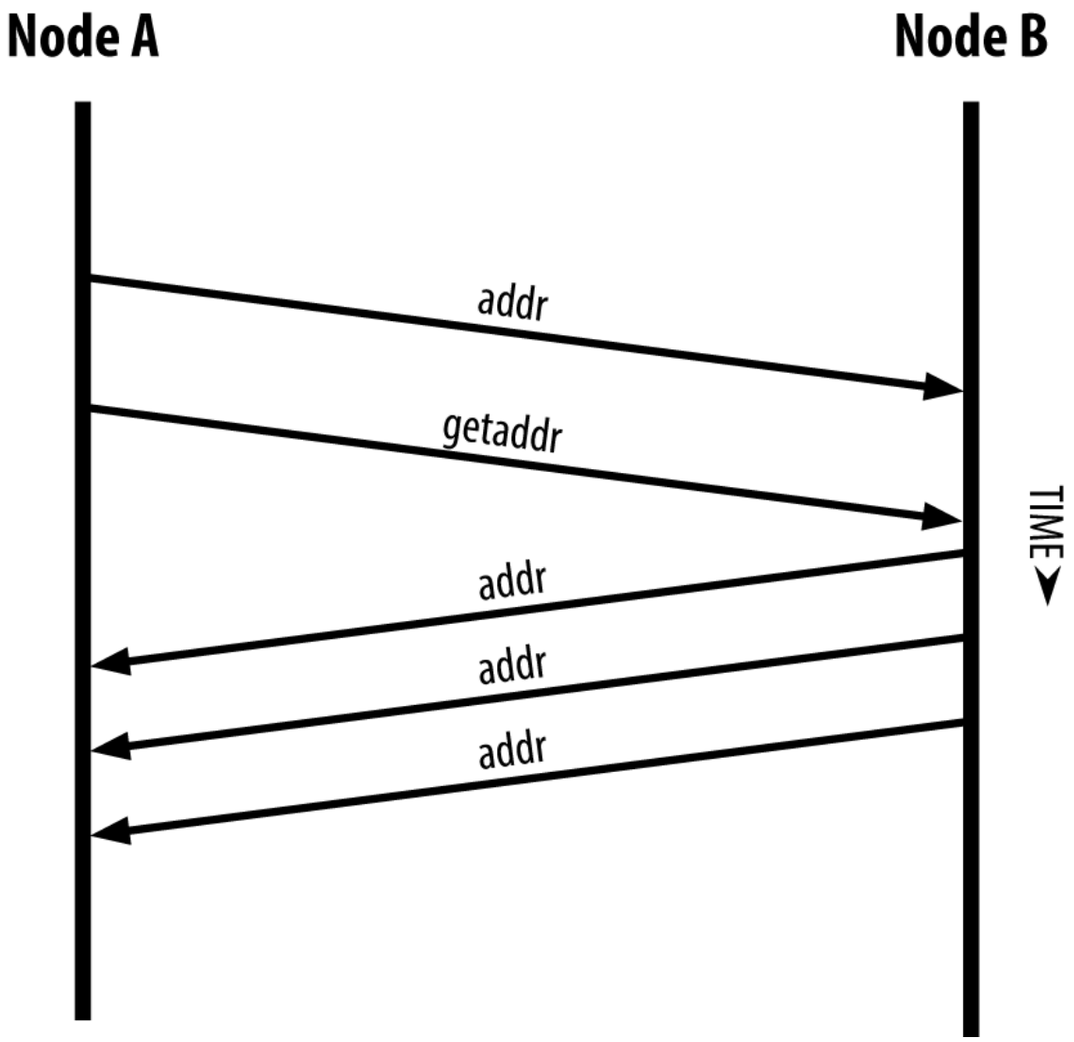

D


1
2
7
6
3
4
5
8

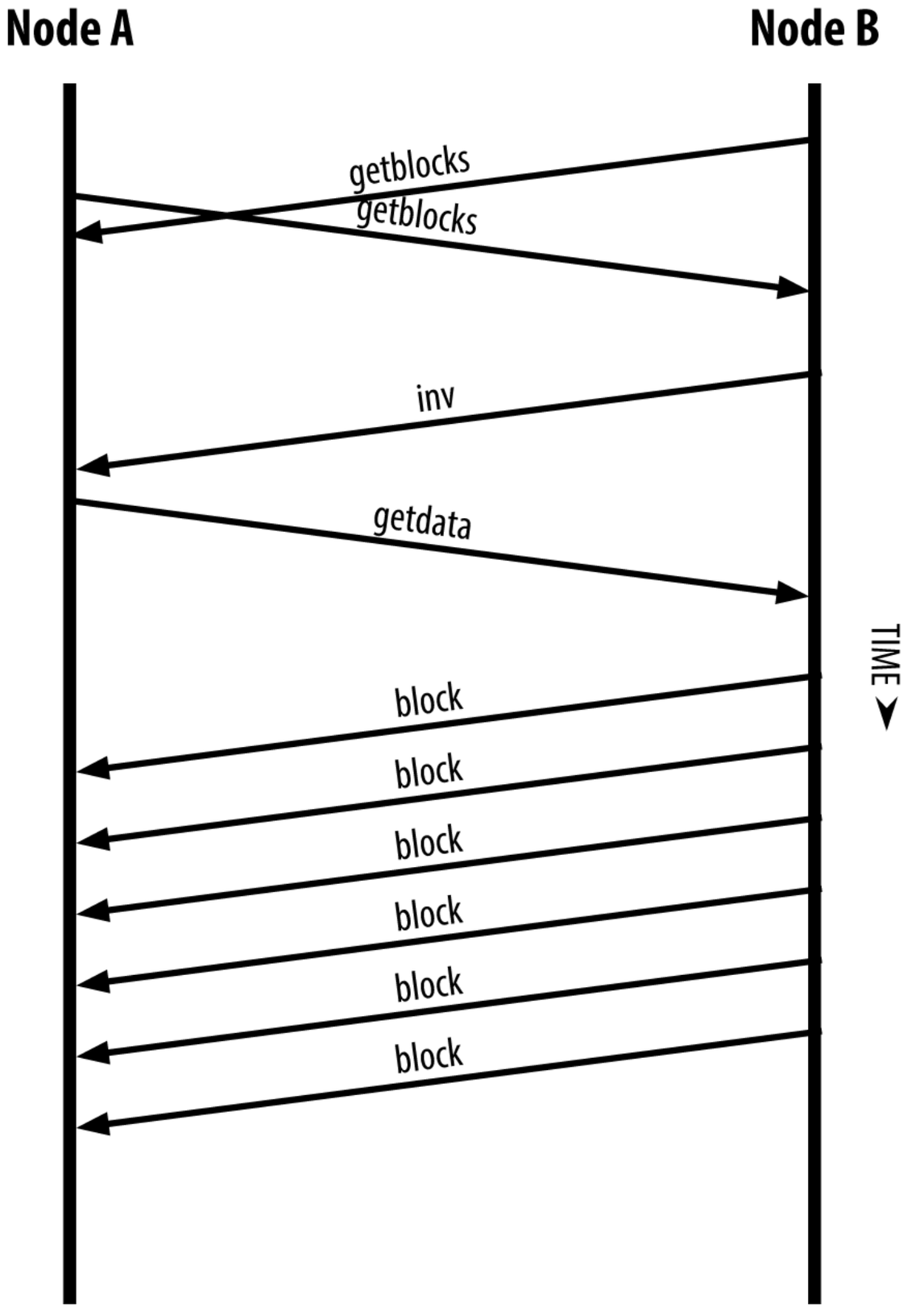

Data
Key
Distributed Network
Cat
Black Cat
White Cat
Hash Function
B125878B
Y158852Y
45564HJ8
Peers

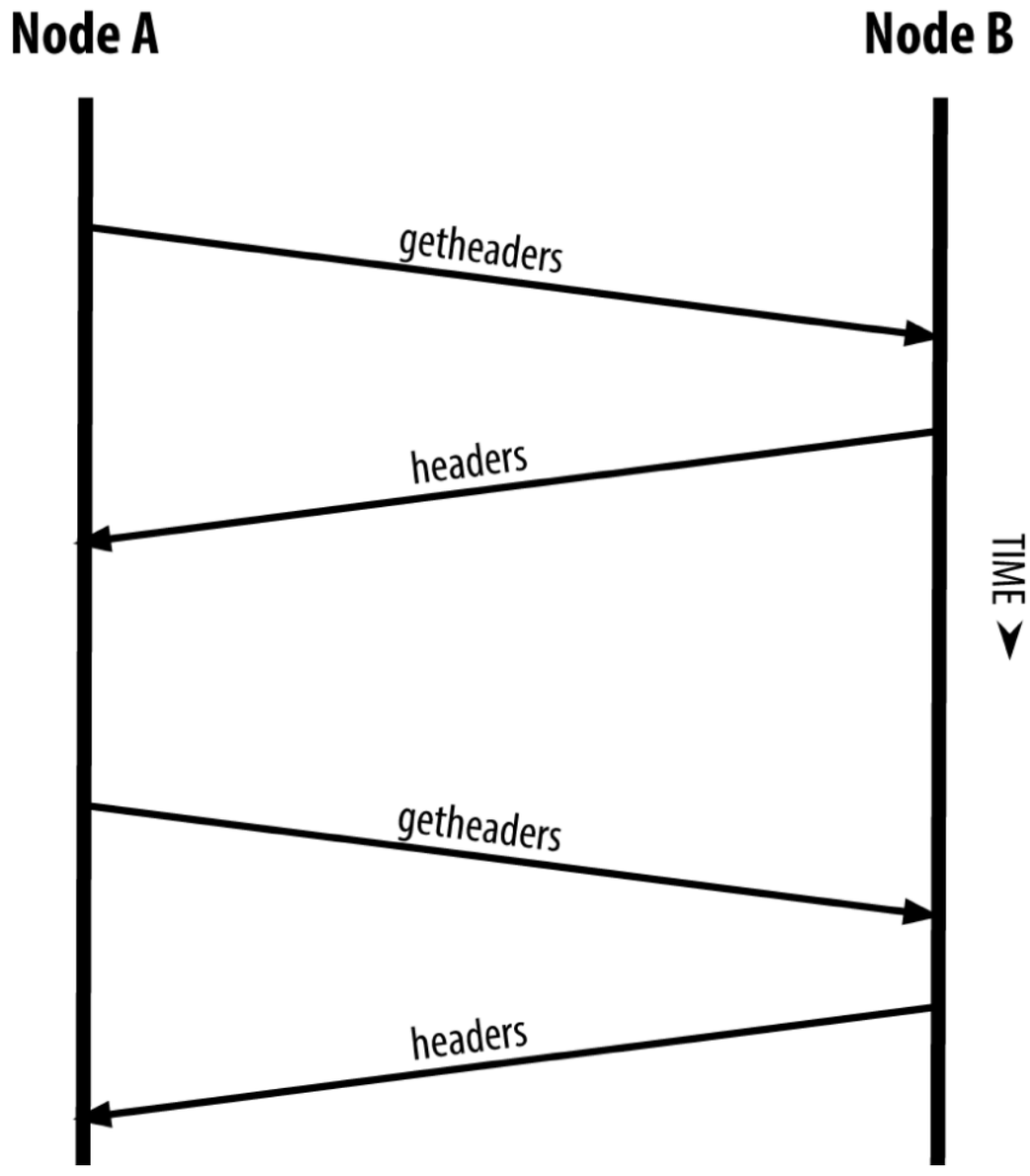
# 网络协议


Node A
Node B
version
verack
version
verack
TIME


Node A
Node B
addr
getaddr
addr
addr
addr
TIME

# 网络协议

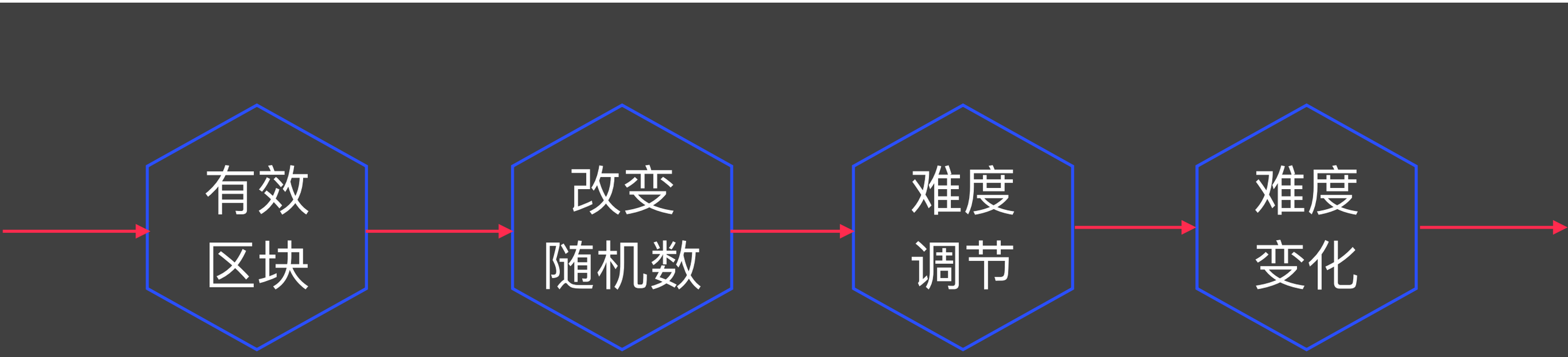

Node A
Node B
getblocks
getblocks
inv
getdata
block
block
block
block
block
block
TIME


Node A
Node B
getheaders
headers
getheaders
headers
TIME

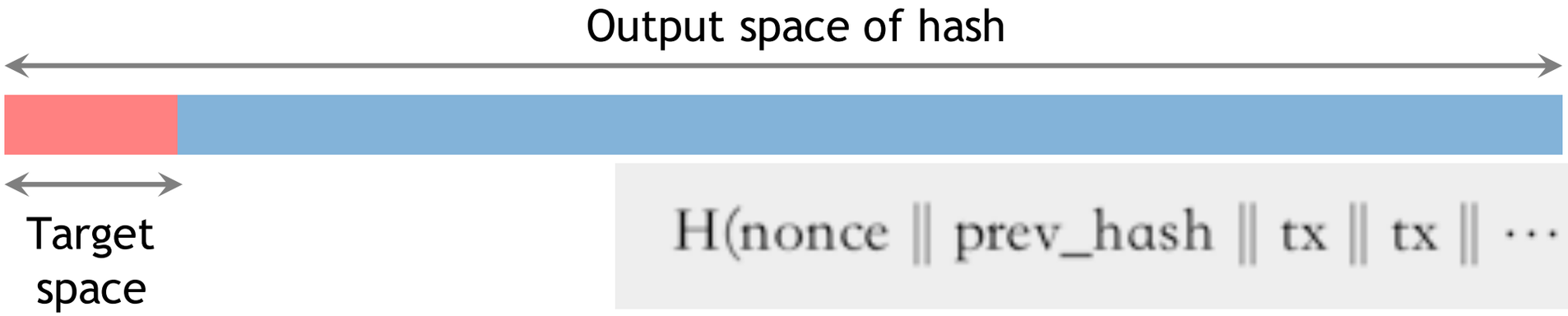
Bitcoin-02

# 比特币网络

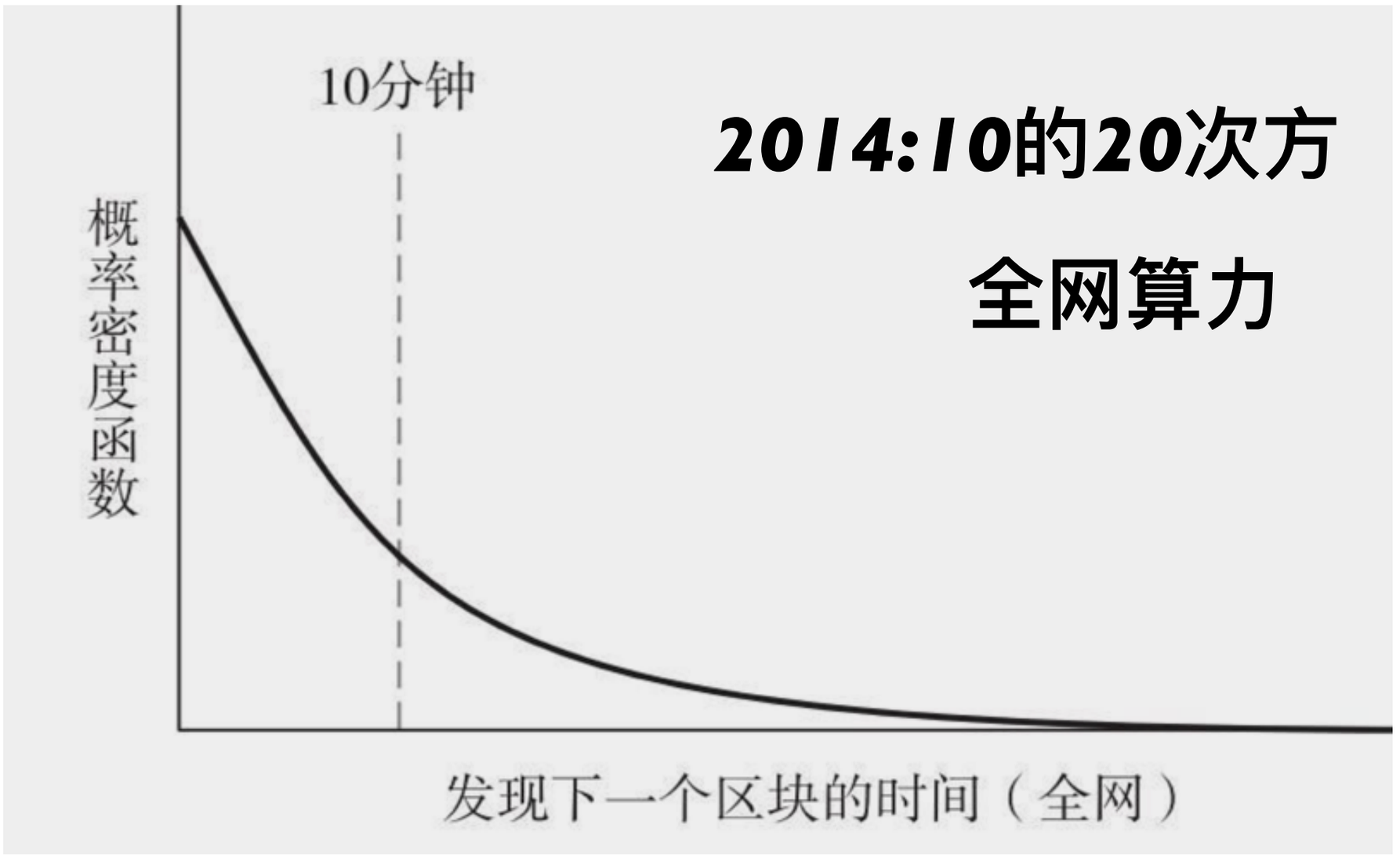
# PoW共识



# 工作量证明




$$H(nonce \parallel prev\_hash \parallel tx \parallel tx \parallel \cdots \parallel tx) < target$$




**2014:10的20次方**

**全网算力**

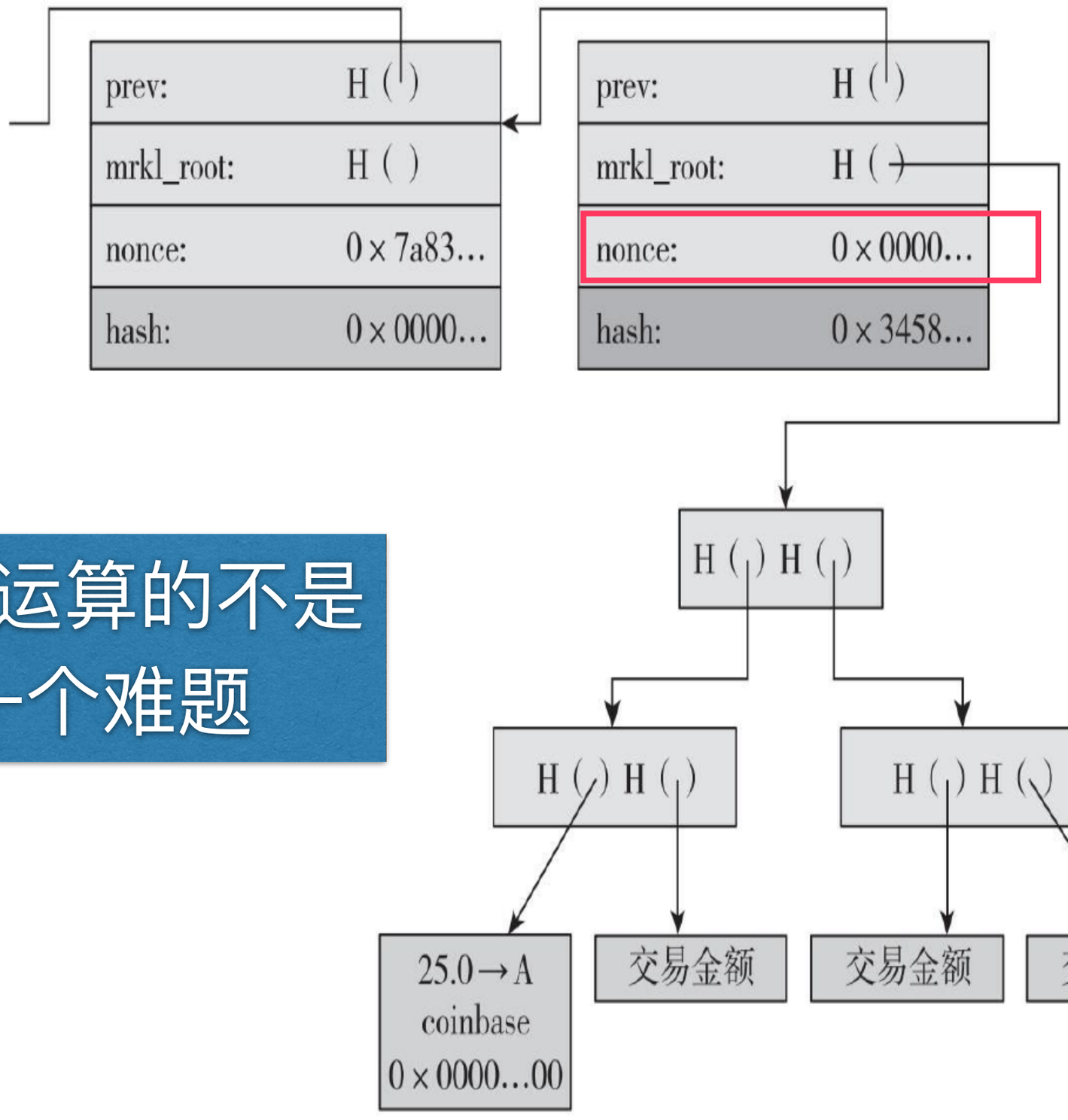
**限定*Hash*的输出范围**

**临时随机数**

***PoW*：工作量证明**

***PoS*：权益证明**

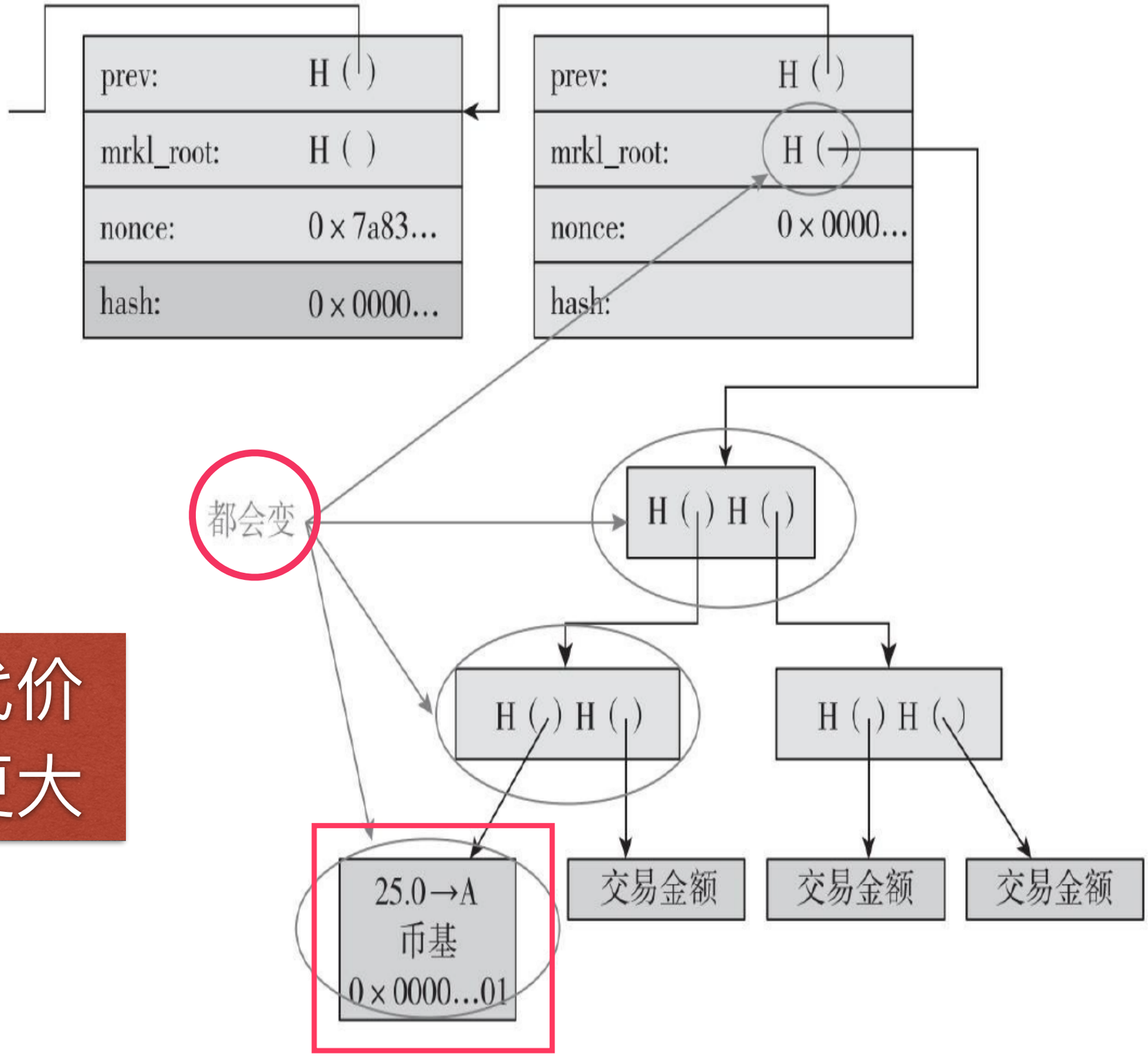
# 寻找有效区块




32位随机数

每个人运算的不是同一个难题

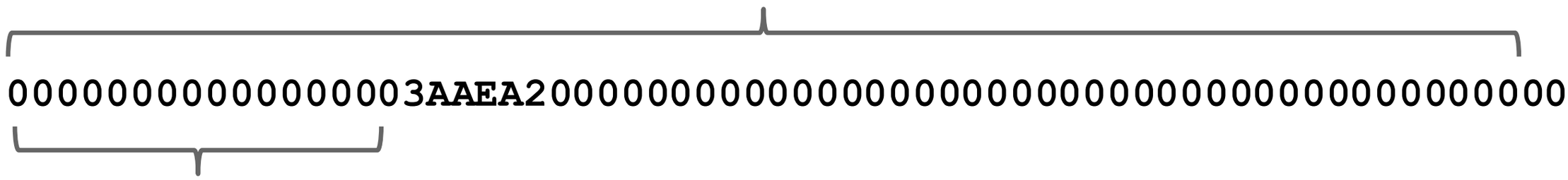
# 改变临时随机数




代价更大

# 挖矿难度

256 bit hash output




当前难度 = $2^{66.2}$

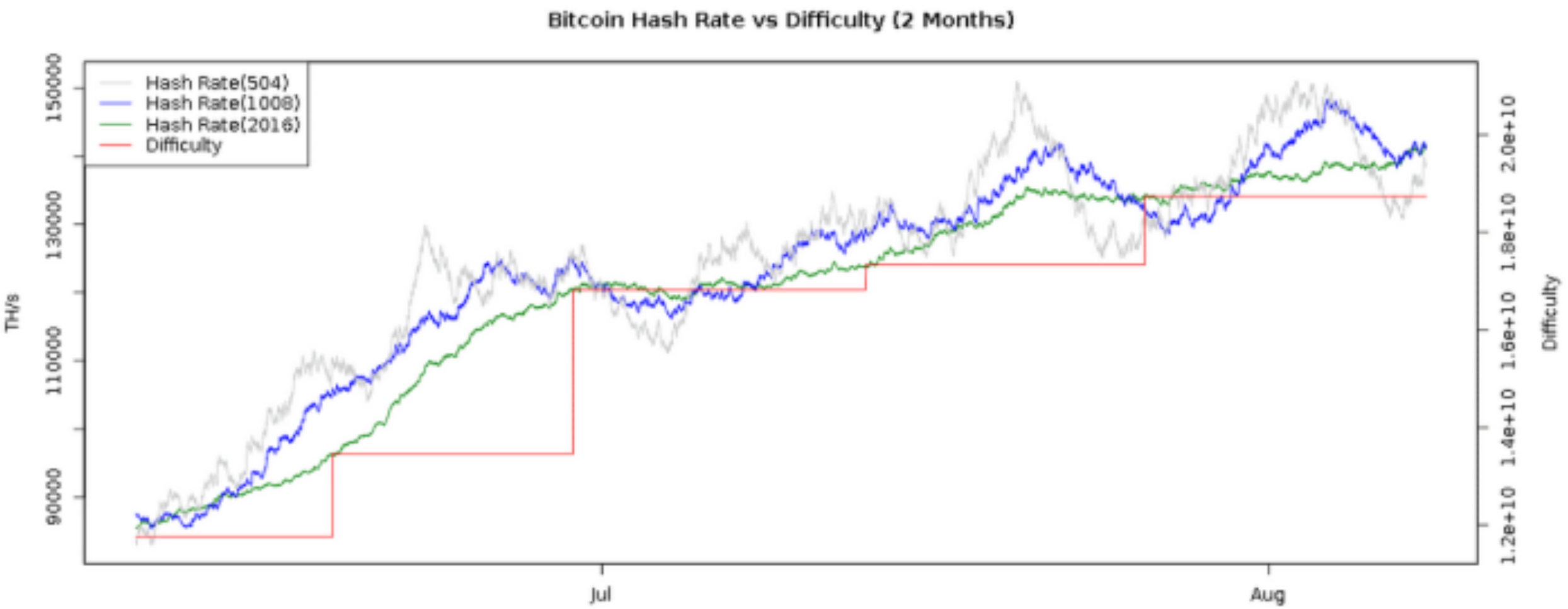
$$\text{下一个难度} = \frac{\text{上一个难度} * 2016 * 10\text{分钟}}{\text{产生上2016个区块所花费时间}}$$

# 币基交易

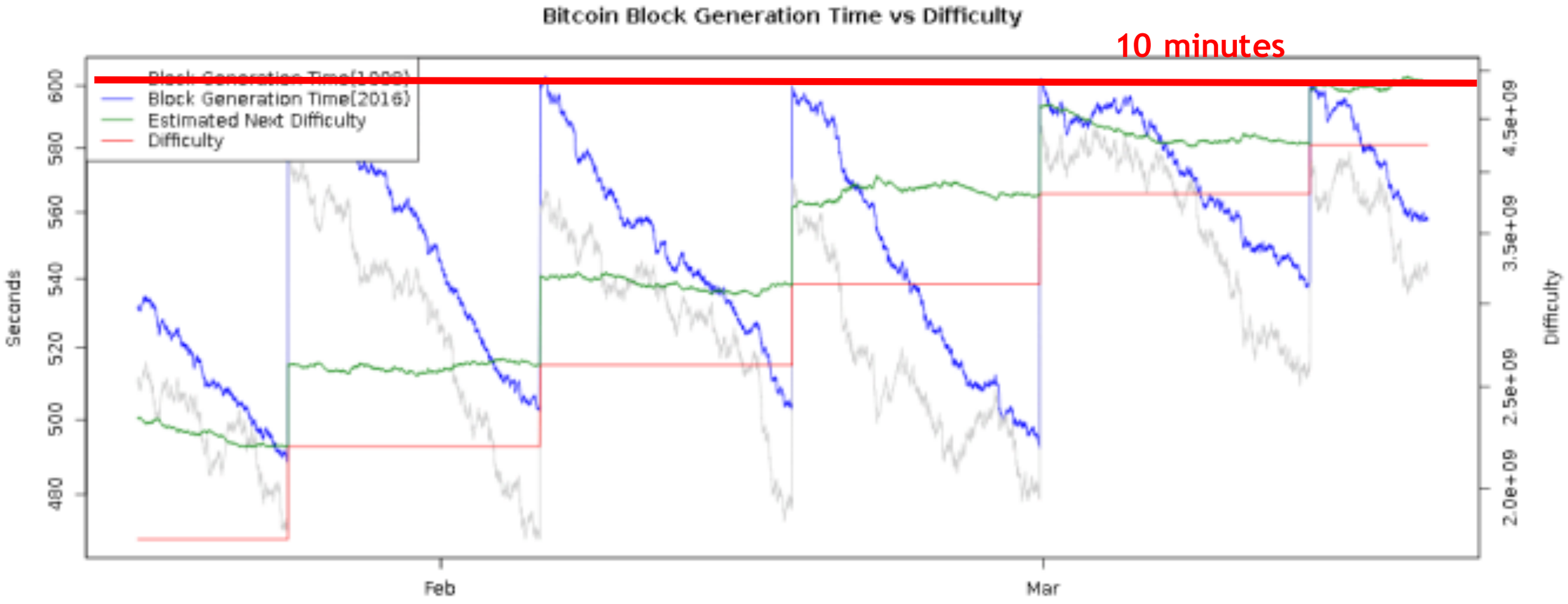
```
"in":[
        {
          "prev_out":{
            "hash":“000000.....0000000",
            "n":4294967295
          },
           "coinbase":"..."
      },
      [
  "out":[
        {
          "value":"25.03371419",
          "scriptPubKey":"OPDUP OPHASH160 ... ”
        }
       ]
```

图3.8 币基交易

# 难度随时间变化


Bitcoin Hash Rate vs Difficulty (2 Months)
Hash Rate(504)
Hash Rate(1008)
Hash Rate(2016)
Difficulty
TH/s
150000
130000
110000
90000
Difficulty
2.0e+10
1.8e+10
1.6e+10
1.4e+10
1.2e+10
Jul
Aug

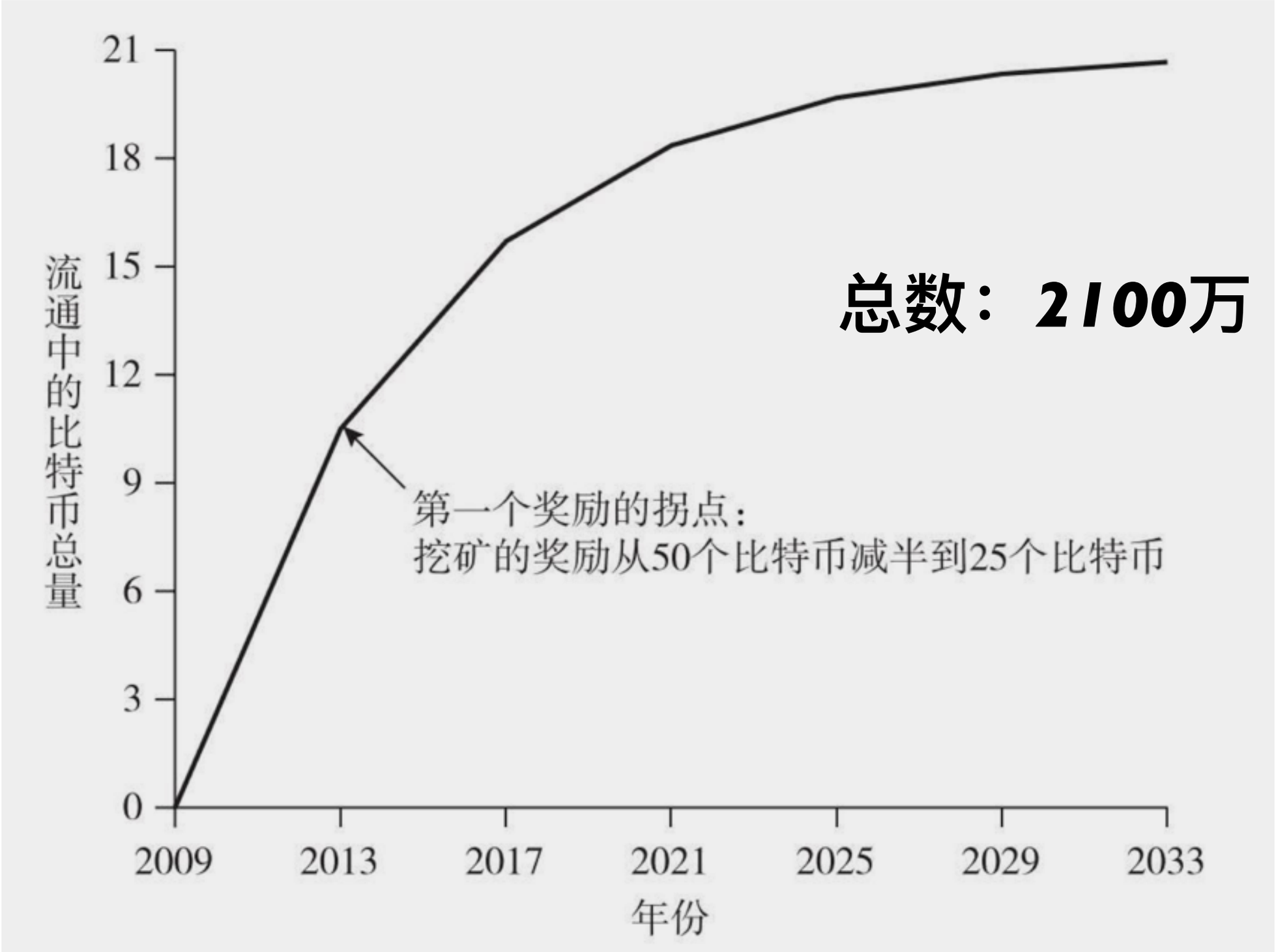
# 发现一个有效区块的时间



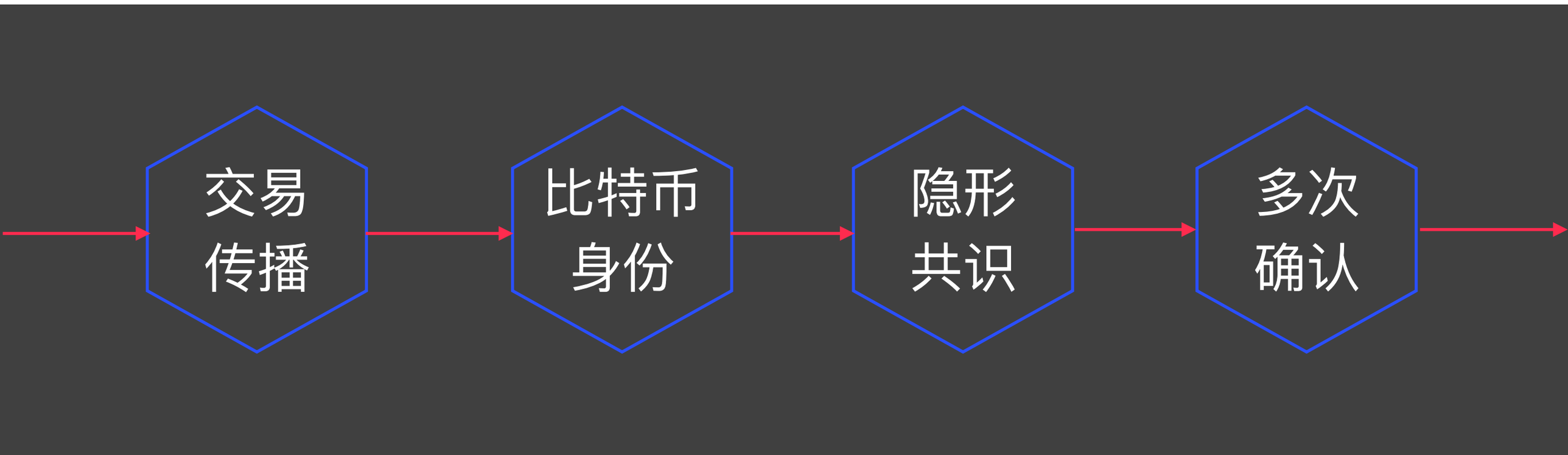
# 挖矿难度


如上图所示，可以看出在2010年至2016年，难度值位于非常低的范围。但是从2016年开始至今，难度值开始大幅度提高，这与挖矿设备算力有着密切关系。

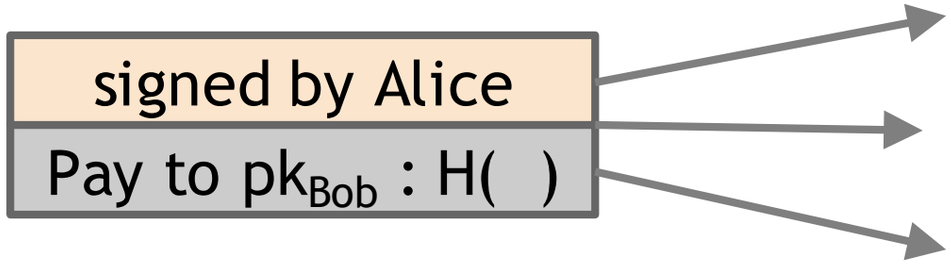
# 比特币奖励

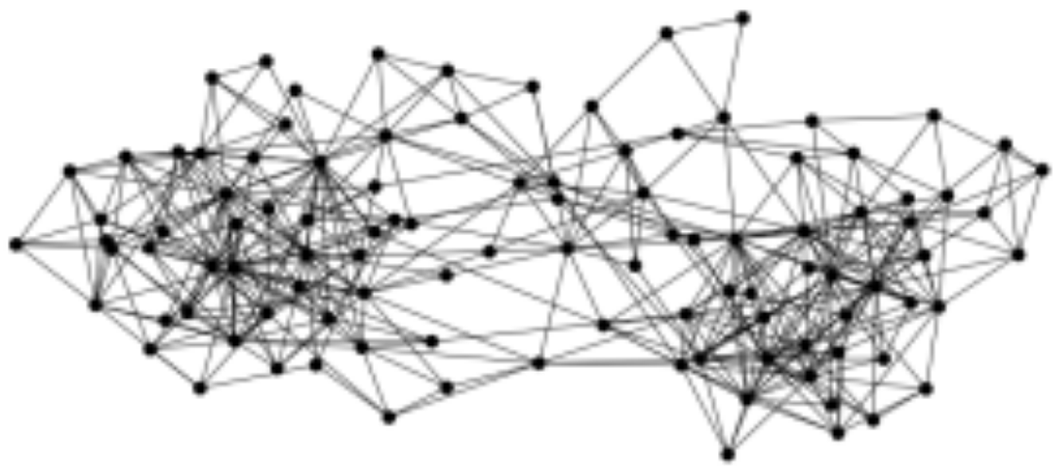

21
18
15
12
9
6
3
0
流通中的比特币总量
2009
2013
2017
2021
2025
2029
2033
年份
总数：2100万
第一个奖励的拐点：
挖矿的奖励从50个比特币减半到25个比特币

# 比特币共识



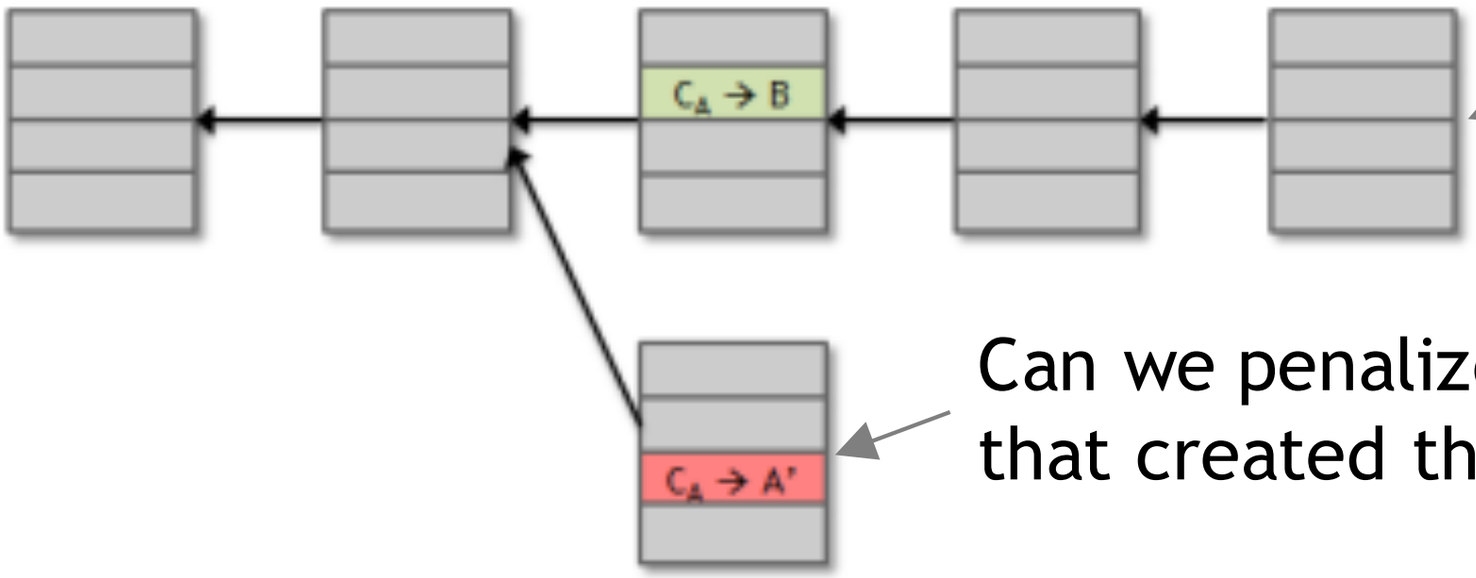
# 比特币的交易传播



- 比特币是一个*P2P*网络
- *Alice* 需要广播她完成的交易給所有的节点
- *Bob*计算机当时可以不在*P2P*网络中
- *A single, global ledger for the system*
- 等待共识的业务、已共识的业务

# 激励节点诚实

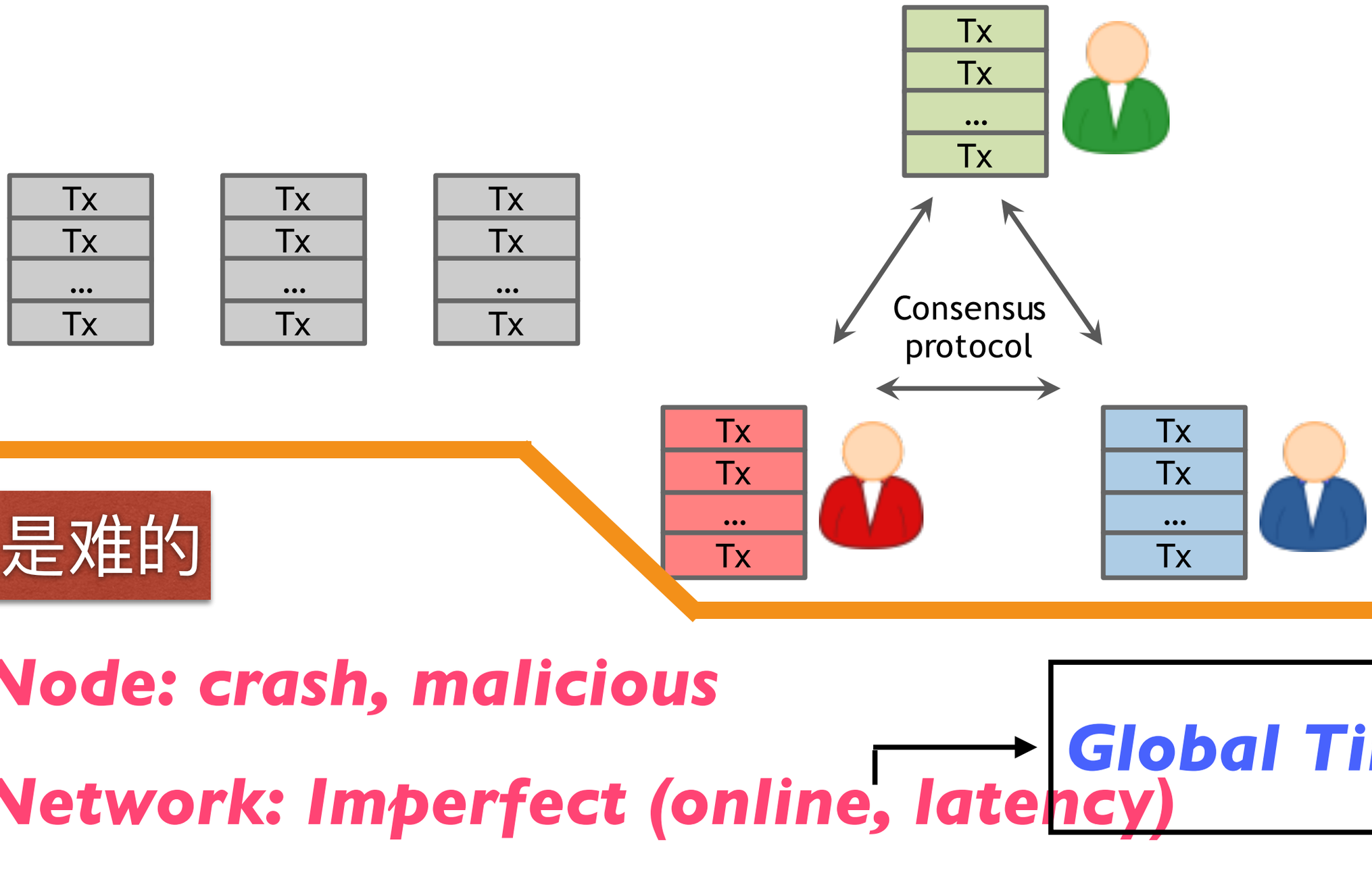



## 区块奖励 *vs.* 交易费奖励

## 交易费：输入和输出不等

# 比特币的分布共识

## 每一个节点输出它的未共识的业务竞争下一个Block




## 共识是难的

- ***Node: crash, malicious***
- ***Network: Imperfect (online, latency)***

***Global Time***

# 比特币和身份

- 比特币节点需要身份 *(ID)*
- 比特币假设恶意节点小于*50%*
- 但是*P2P*系统中，*ID*面临很大问题
  - ✴ *Sybil Attack*
- *Pseudonymity*是比特币的目的

---

- 比特币跟踪和验证*ID*是困难的
- 比特币采用的应对方法：随机的选择节点

# 比特币的隐形共识机制

- 新的交易被广播到所有节点
- 每个节点将新的交易放进一个区块
- 在每一轮中，一个随机的节点被选择可以广播它的区块
- 其余节点可以选择接受这个区块，前提是区块的交易是可验证的
- 节点将以上区块的*Hash*放进自己的区块，表示它认可这个新区块

隐形共识：接受该块并扩展 vs. 拒绝该块，扩展前面的块

# 恶意节点

窃取比特币

拒绝服务攻击

双重支付攻击

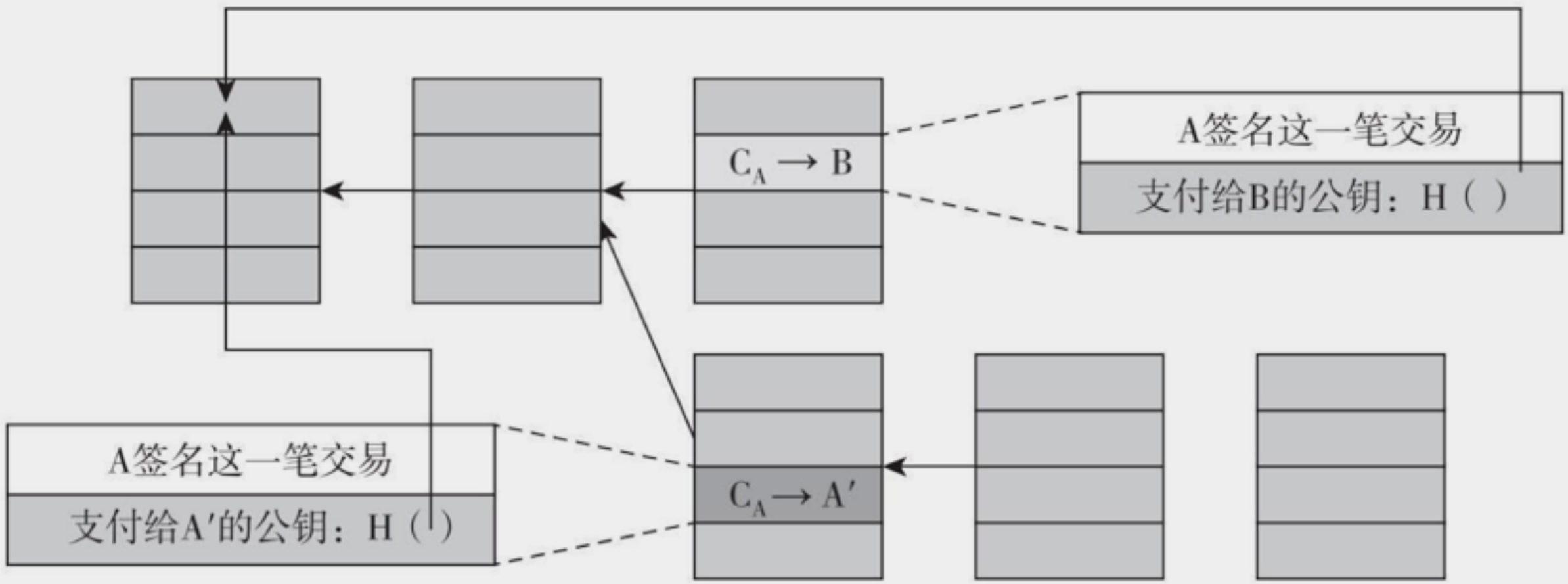



图2.2 双重支付攻击

注：爱丽丝创建了两笔交易：一笔是她付给鲍勃比特币的交易，另一笔是她将这笔比特币重复支付到她控制的另一个地址。因为这两笔交易用相同的比特币支付，所以只有一笔会被放进区块链。图中的箭头表示一个区块链接到前一个区块的指针，通过在前一个区块自己的内容中包含了一个哈希值进行了扩展。$C_A$代表爱丽丝拥有的币。

# 双重攻击防止：等待多次确认

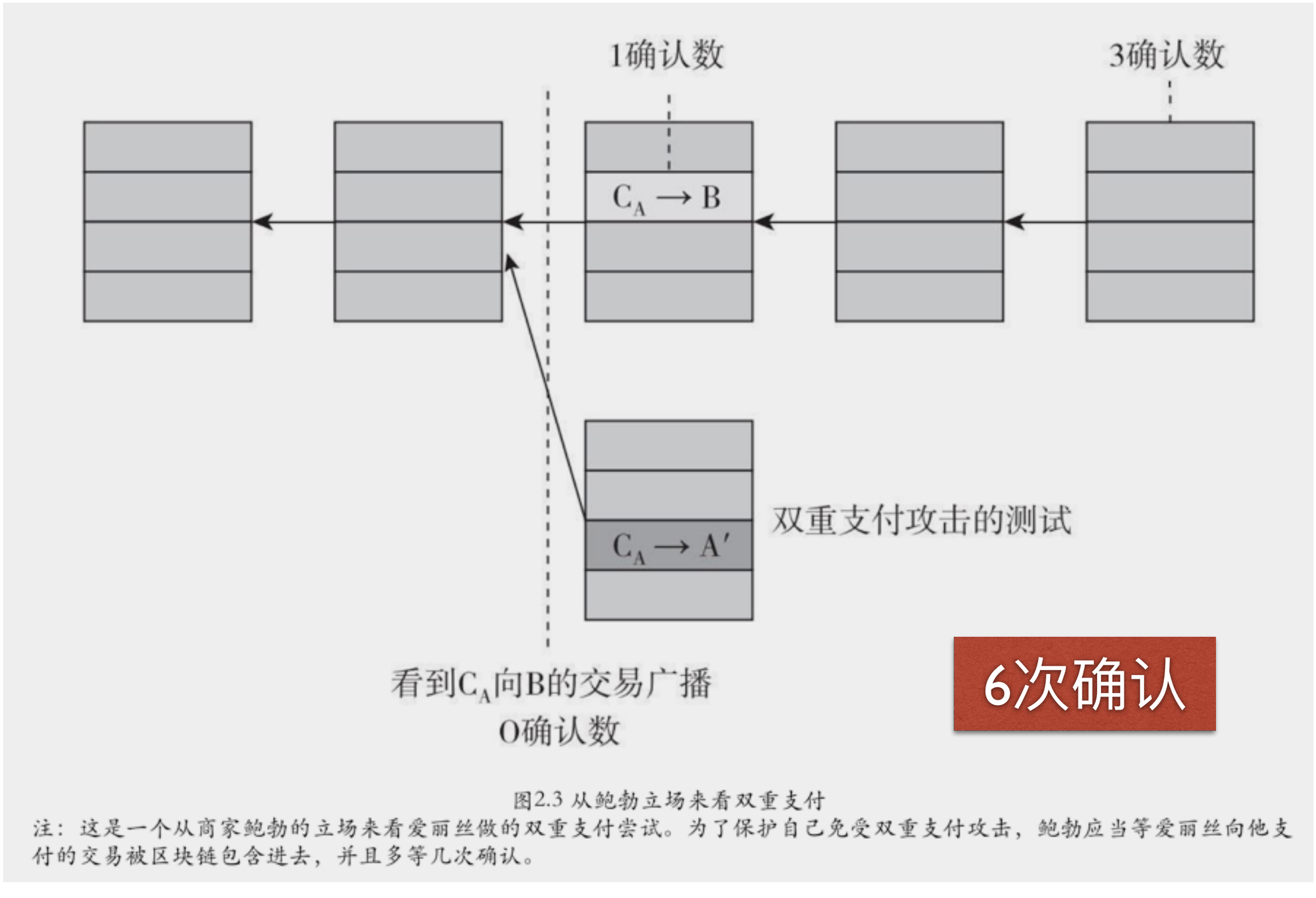
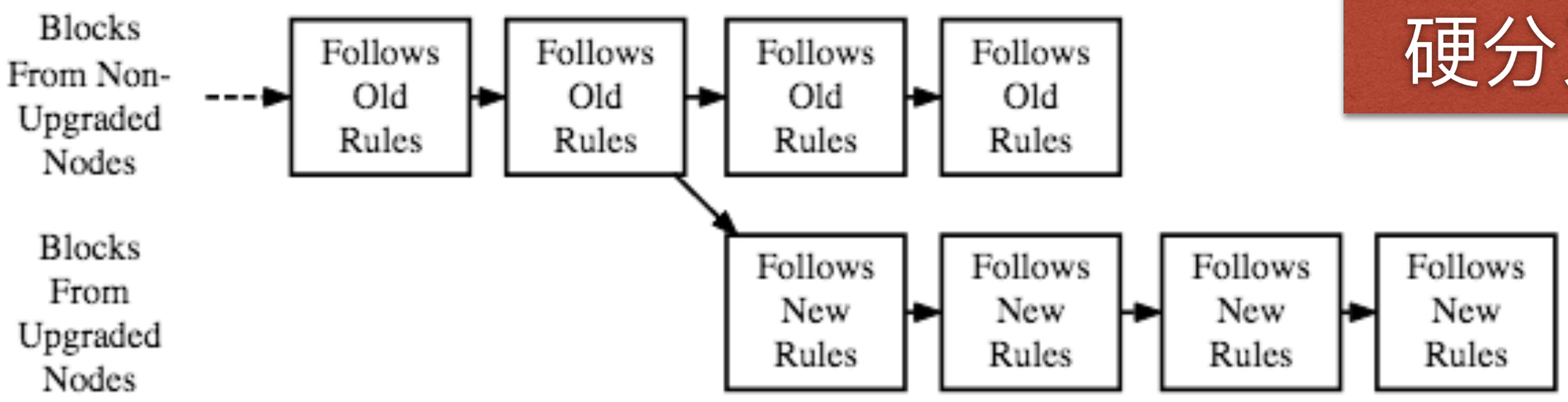



图2.3 从鲍勃立场来看双重支付

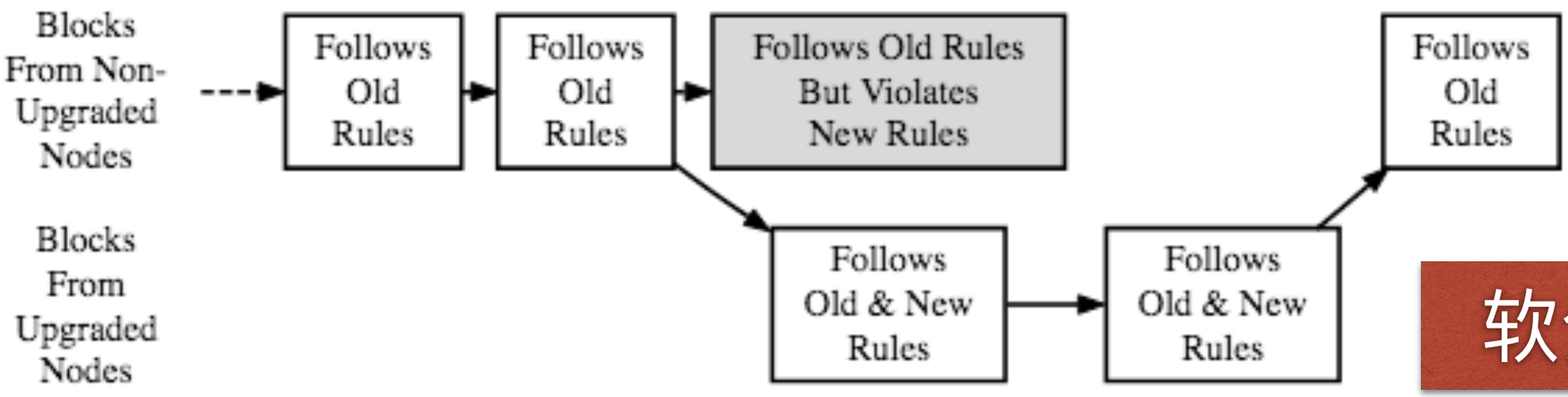
注：这是一个从商家鲍勃的立场来看爱丽丝做的双重支付尝试。为了保护自己免受双重支付攻击，鲍勃应当等爱丽丝向他支付的交易被区块链包含进去，并且多等几次确认。

# 共识机制改变




硬分叉

A Hard Fork: Non-Upgraded Nodes Reject The New Rules, Diverging The Chain




BIP16

BIP30

BIP34

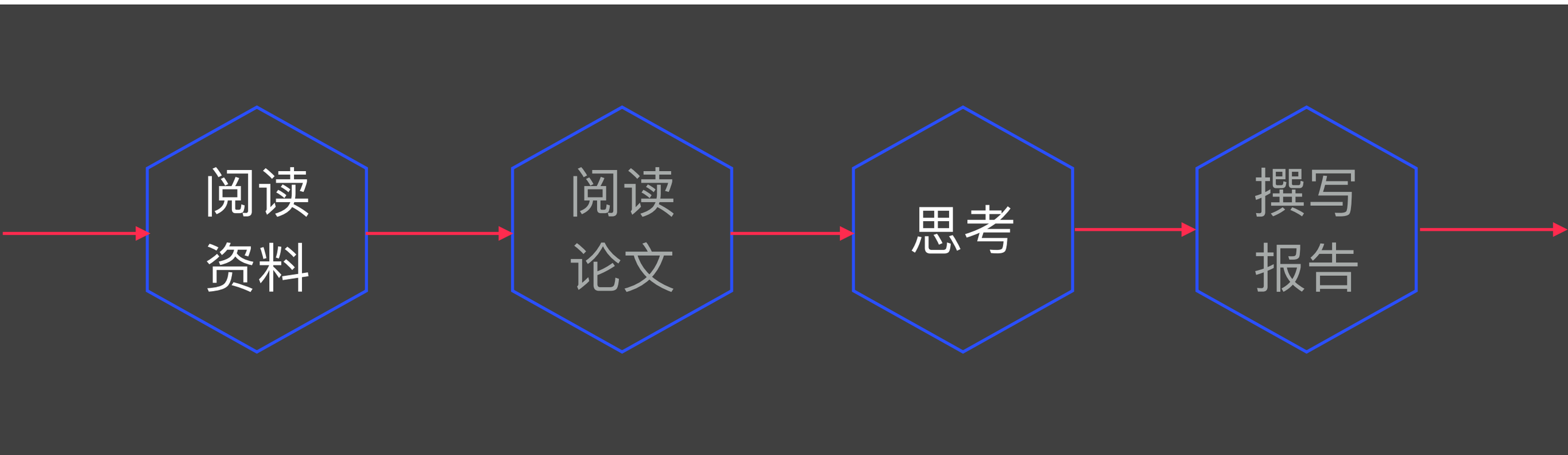
软分叉

A Soft Fork: Blocks Violating New Rules Are Made Stale By The Upgraded Mining Majority

# 比特币的共识总结

- 理论落后于实践
- 引入了*Incentive*
  - ✳ 是电子货币
- 利用了随机性
  - ✳ 很长时间后才取得共识，*1*小时
  - ✳ 随着时间的增加，对某一块的共识的概率越来越大

# 课后作业



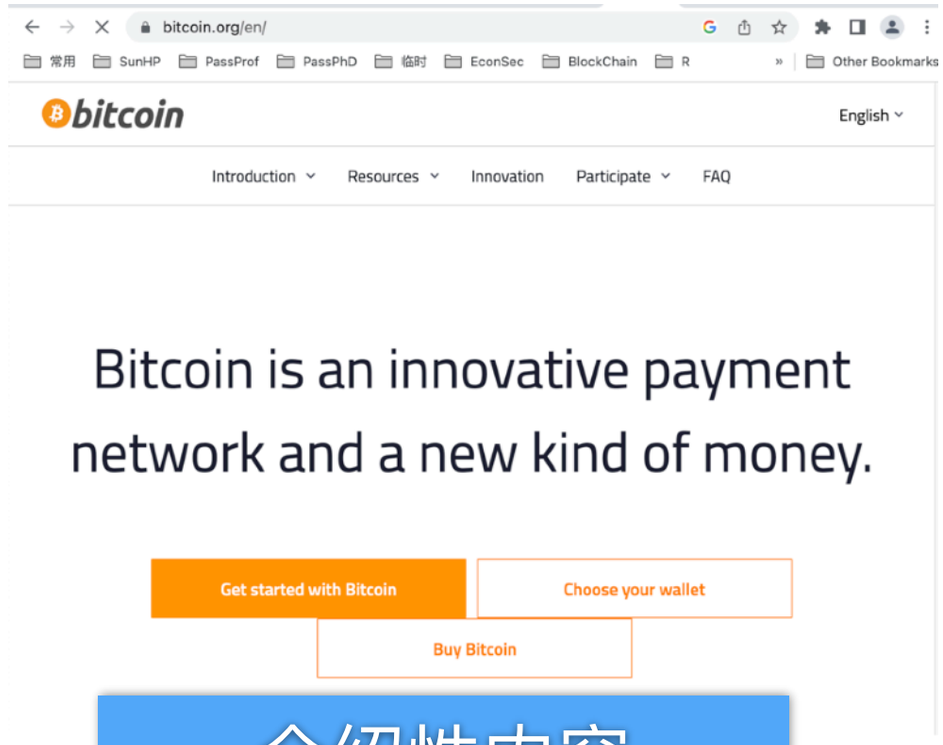
# Homework 课后阅读建议


阅读全部章节

*https://www.8btc.com/book/281955*

*https://github.com/bitcoinbook/bitcoinbook*

*https://www.bitcoin.org/*




介绍性内容
Introduction
White Paper

开发方面内容
Developer Guides、Reference、
Examples、Learning Resources

# 谢谢！

孙惠平
sunhp@ss.pku.edu.cn