第六章：比特币网络

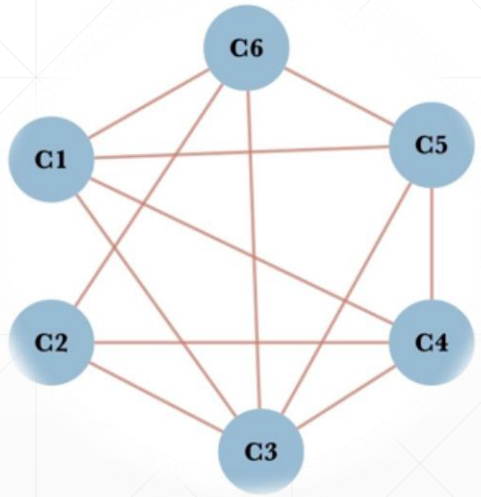
裴奇

2018.1

P2P网络架构

● 定义

P2P分布式网络架构是指位于同一网络中的每台计算机都彼此对等，各个节点共同提供网络服务，不存在任何“特殊”节点。每个网络节点以“扁平（flat）”的拓扑结构相互连通。



● 特点

1. 可靠性
2. 去中心化
3. 开放性

- 比特币被设计为一种点对点的数字现金系统。它的网络架构既是这种核心特性的反映，也是该特性的基石。
- 比特币去中心化控制是设计时的核心原则，它只能通过维持一种扁平化、去中心化的P2P共识网络来实现。

节点类型及分工

- **比特币网络**: 按照比特币P2P协议运行的一系列节点的集合。
- 每个比特币节点都是**路由**、**区块链**、**挖矿**、**钱包服务**的功能集合。

标准客户端（比特币核心）：拥有最完整功能的节点

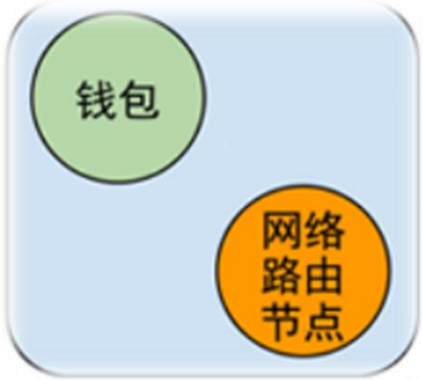
- **网络路由节点**：参与验证并传播交易及区块信息，发现并维持与对等节点的连接
- **完整区块链**：能够追溯到创世区块的账本
- **矿工**：通过不断重复哈希运算来产生工作量证明
- **钱包**：私钥、地址和区块链数据的管理工具

节点类型及分工

全区块链节点 ：
是运行着全节点客户端的节点，也可作为边缘路由器

VS

简易支付验证（SPV）节点:又叫“轻量级钱包”



节点类型及分工

全节点	SPV节点
★ 有完整的、最新的**包含全部交易信息的比特币区块链拷贝**	★ **只包含区块头**，不包含区块中的交易信息
不需借助或信任其他系统即**可独立地对所有交易信息进行验证**	**不能独立验证交易的有效性**，需借助全节点验证交易存在性，但不能验证交易不存在
通过交易所在区块链中的**高度**来验证	通过参考交易在区块链中的**深度**来验证
需要**很大的磁盘空间**、并且同步比特币网络耗时2至3天。	**大小只有完整区块链的1/1000。**

- SPV节点没有一份关于所有交易的记录。针对SPV节点的**拒绝服务攻击**或**双重支付型攻击**
- SPV节点需要**随机连接到多个节点**，以增加与至少一个可靠节点相连接的概率
- 这种随机连接的需求意味着SPV节点也容易受到**网络分区攻击**或**Sybil攻击**

具有良好连接的SPV节点是足够安全的！！

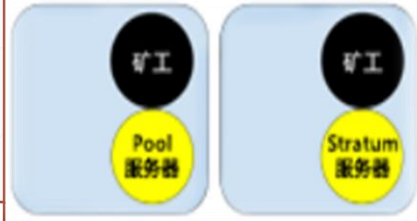
节点类型及分工

挖矿节点: 通过运行特殊硬件设备上的工作量证明（proof-of-work）算法，以相互竞争的方式创建新的区块。

独立矿工：（全节点）包含完整区块链、挖矿功能以及网络路由功能的节点

矿池协议服务器：将运行着其他协议的节点连接到P2P网络的网关路由器。

挖矿节点：（轻量级节点）通过矿池协议服务器连接进P2P网络并进行挖矿的节点。

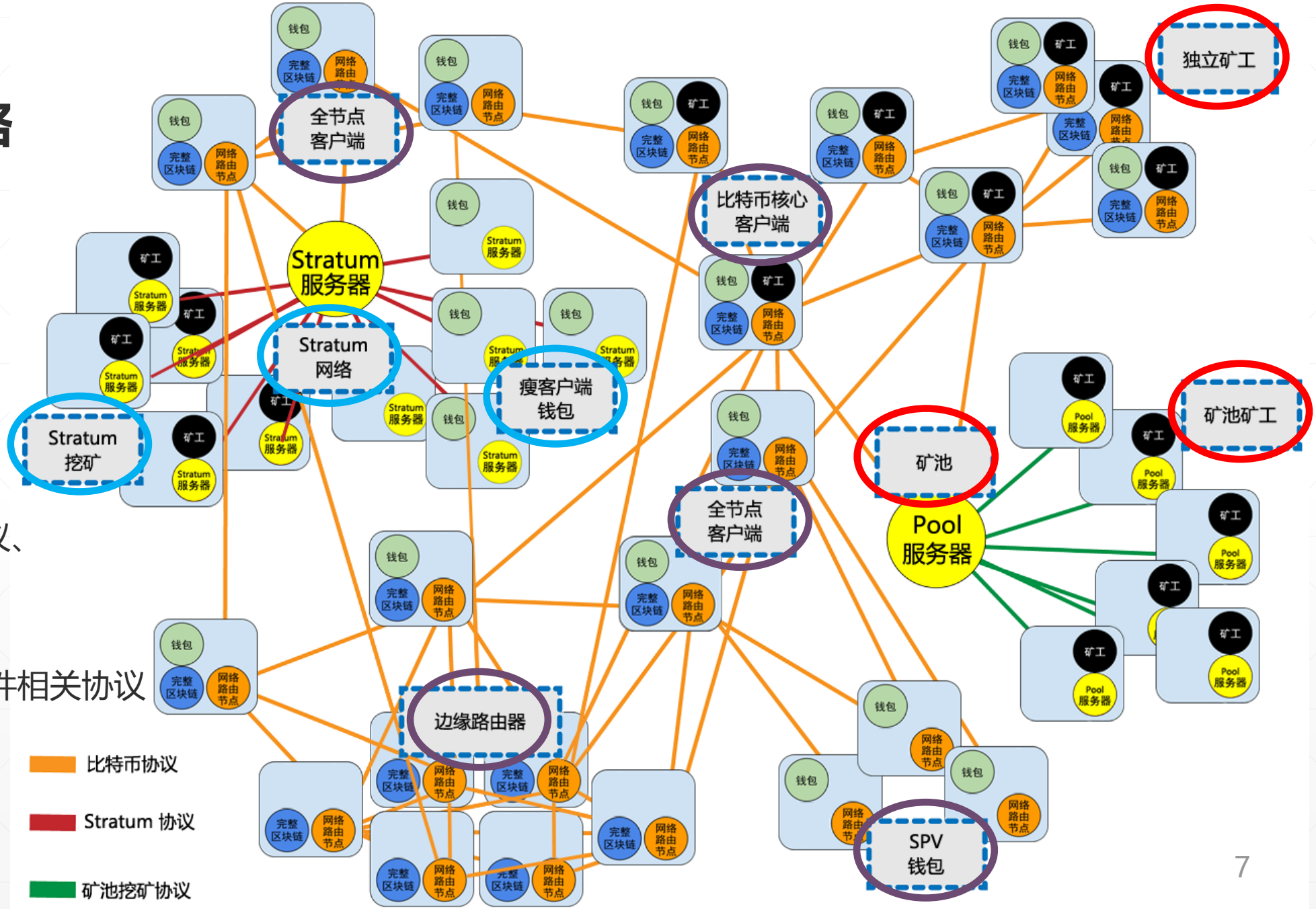


轻量（SPV）Stratum钱包（瘦客户端钱包）：包含不具有区块链的钱包，运行Stratum协议的网络节点。

扩展比特币网络

- **定义**

所有包含比特币P2P协议、
矿池挖矿协议、
Stratum协议以及
其他连接比特币系统组件相关协议
的整体网络结构。



比特币中继网络

背景：

- 当比特币p2p网路为多种类型节点服务的时候，它也为有专门需求的比特币**挖矿节点**带来了很高的**网络时延**。矿工从事**时间敏感的竞争**去解决工作量证明问题从而去拓展区块链。时延直接影响了收入盈利。

介绍：

- 比特币中继网络是一个用来**最小化区块在矿工之间传输时延**的网络。
- **原始的**比特币中继网络：该网络由世界各地的亚马逊Web服务基础架构上托管的几个专门的节点组成，并且连接大多数矿工和采矿池。
- 2016年被**FIBRE**（ Fast Internet Bitcoin Relay Engine ）代替——一个基于UDP的中继网络。
  1. 它可以通过发送额外的数据来弥补**数据包丢失**，从而消除延迟峰值；
  2. 实现**紧凑块**（compact block）优化，进一步减小数据传输量和传输时延；
- **Falcon**（提案中）使用"直通路由"（cut-through-routing）而不是"存储转发"来减少延迟。
  - 通过传播块的部分，而不是等待直到接收到完整的块。

网络发现与连接

- 当一个新节点启动的时候，为了加入网络，它需要发现网络中至少一个存在的比特币节点并进行连接。因为其他节点的地理位置是无关的以及比特币网络的拓扑结构不是地理上定义的，因此可以**任意的**选择存在的比特币节点。

- 寻找接入比特币网络的种子节点

  1. （默认）通过查询硬编码在比特币核心中的DNS种子，得到一个或几个可能接受此连接的全节点IP地址，选择其中的一个ip地址进行连接；
  2. 使用一个已知的对等节点的ip地址；例如： 用户在命令行指定 -seednode ，作为“引荐节点”。

Inbound：8 最多8个人连接我

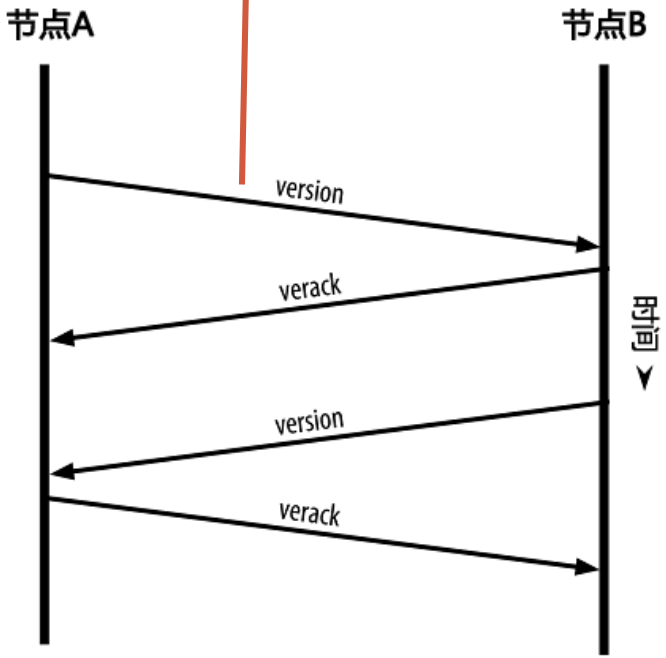
outbound ：125 可以去连接125个其他节点

网络发现与连接

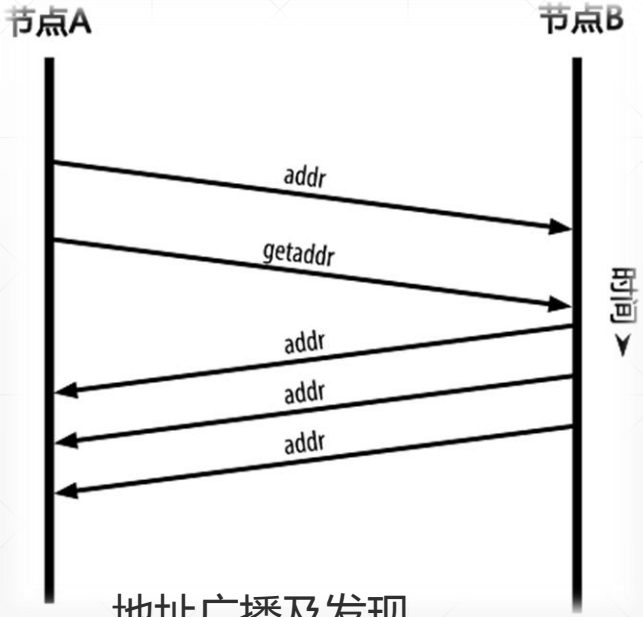
Version
1. PROTOCOL_VERSION 2. nLocalServices 3. nTime 4. addrYou 5. addrMe 6. subver 7. BaseHeight

首先检查协议版本是否一致

节点通常采用**TCP协议**、8333端口（该端口号通常是比特币所使用的，除8333端口外也可以指定使用其他端口）与已知的对等节点建立连接。



对等节点之间的初始"握手"通信



地址广播及发现

- 新接入的节点将一条包含自身IP地址的addr消息发送给对等节点；
- 新接入的节点可以向对等节点发送getaddr消息，要求它们返回其已知对等节点的IP地址列表；

网络发现与连接

- 一个节点必须连接多个不同的对等节点从而建立多种加入比特币网络中的的路径。
- 节点有加入有离开➔路径是不可靠的
  1. 当失去旧连接时持续去发现新的节点
  2. 协助新的节点启动

- 启动完成后，节点会记住它最近成功连接的对等节点；因此，当重新启动后它可以迅速与先前的对等节点网络重新建立连接。如果先前网络的对等节点对连接请求无应答，该节点可以使用种子节点进行重启动。
- 如果节点持续某个连接长达90分钟（心跳?）没有任何通信，它会被认为已经从网络中断开。

根据变化的节点及网络问题进行动态调整，不需经过中心化的控制即可进行规模增、减的有机调整。

Message header：所有在网络协议中的消息都使用相同的消息头格式

Bytes	Name	Data Type
4	start string	char[4]
12	command name	char[12]
4	payload size	uint32_t
4	checksum	char[4]

是硬编码的常量，出现在比特币网络上传送的消息的开始，
用来识别消息的来源网络

是一个二进制的字符串，确定了有效载荷中的消息类型，
如果字节数不够用0补充

有效载荷中的字节数

以小端形式表示的，对有效载荷进行两次SHA256，取前4个字节
如果载荷是空的（e.g.verack、getaddr消息），那么就是0x5df6e0e2
(SHA256(SHA256(<empty string>))).

例：Verack 消息的消息头

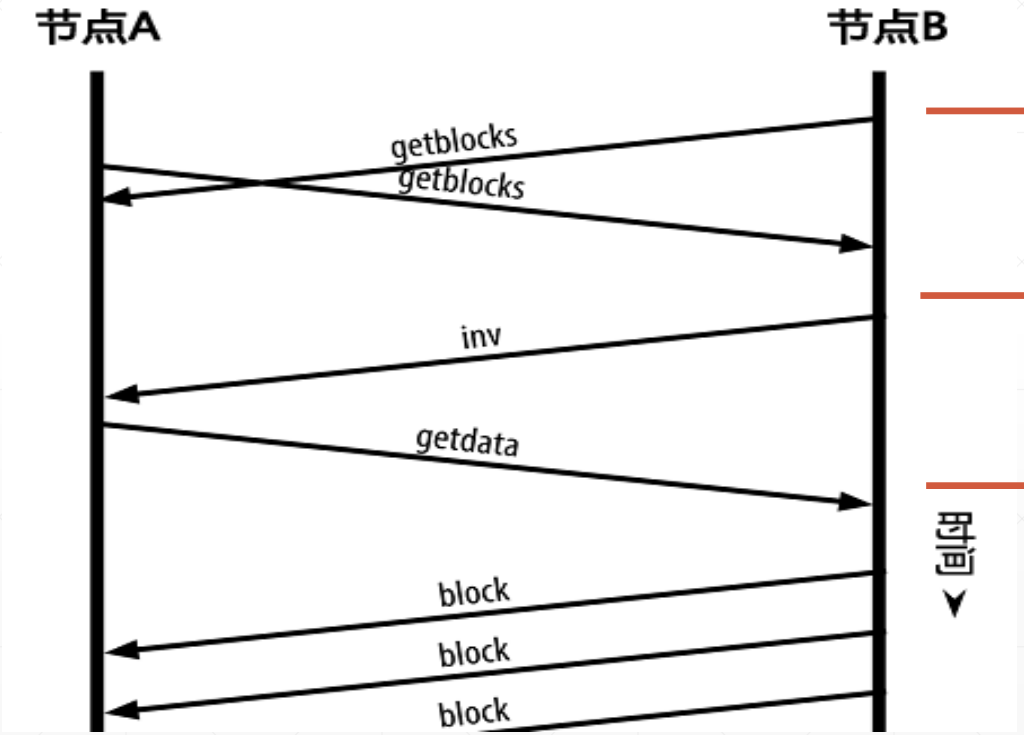
f9beb4d9 ..... Start string: Mainnet

766572616 36b000000000000 ... Command name: verack + null padding

00000000 ..... Byte count: 0

5df6e0e2 ..... Checksum: SHA256(SHA256(<empty>))

全节点交换“库存清单”——Initial Block Download (IBD)



节点通过从对等节点读取区块来同步区块链

getblocks消息用于**第一次启动**或者**断开连接的节点**去请求它缺失的块。

inv（inventory）消息把对等节点所需**区块的哈希值**回复出去，getblock的response

缺少这些区块的节点便可以通过各自发送的getdata消息来请求得到全区块信息

全节点交换“库存清单”——Initial Block Download (IBD)

Field Size	Description	Data type	Comments
4	version	uint32_t	the protocol version
1+	hash count	var_int	number of block locator hash entries
32+	block locator hashes	char[32]	block locator object; newest back to genesis block (dense to start, but then sparse)
32	hash_stop	char[32]	hash of the last desired block; set to zero to get as many blocks as possible (500)

getblocks

Message header

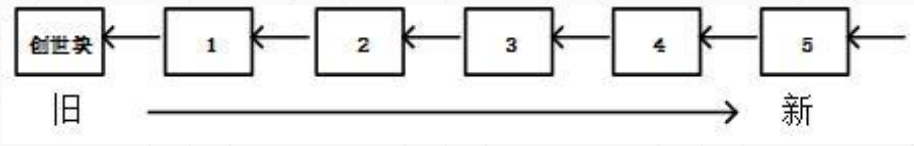
71110100 ..... Protocol version: 70001
02 ..... Hash count: 2

d39f608a7775b537729884d4e6633bb2
105e55a16a14d31b0000000000000000 ... Hash #1

5c3e6403d40837110a2e8afb602b1c01
714bda7ce23bea0a0000000000000000 ... Hash #2

0000000000000000000000000000000000000000000000000000000000000000 ... Stop hash

Block header hashes



Field Size	Description	Data type	Comments
?	count	var_int	Number of inventory entries
36x?	inventory	inv_vect[]	Inventory vectors

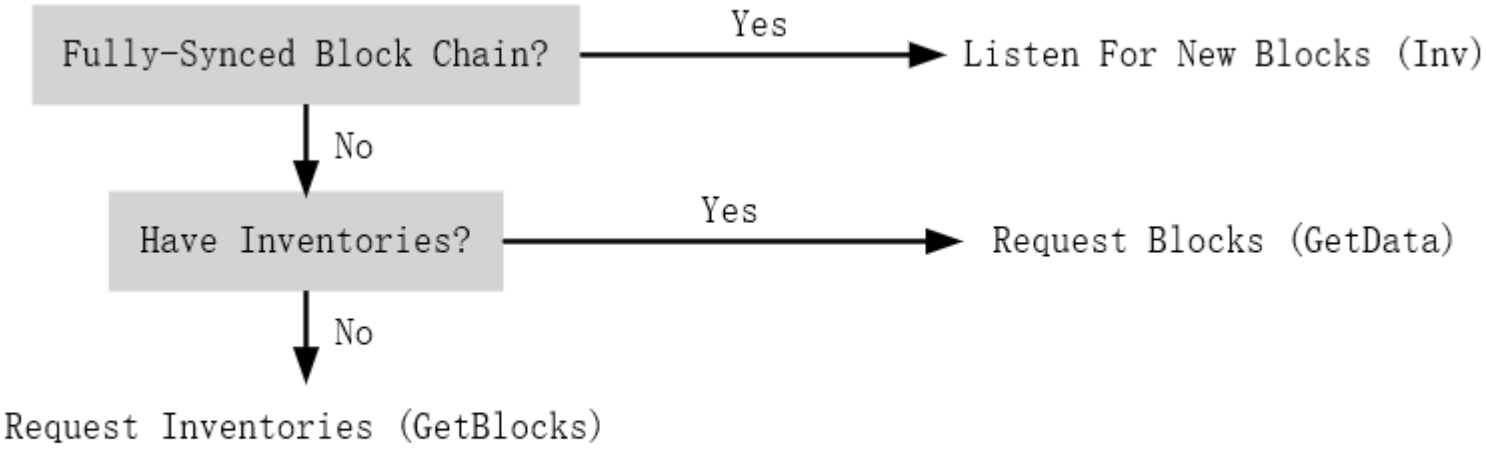
Inv message

Field Size	Description	Data type	Comments
4	type	uint32_t	Identifies the object type linked to this inventory
32	hash	char[32]	Hash of the object

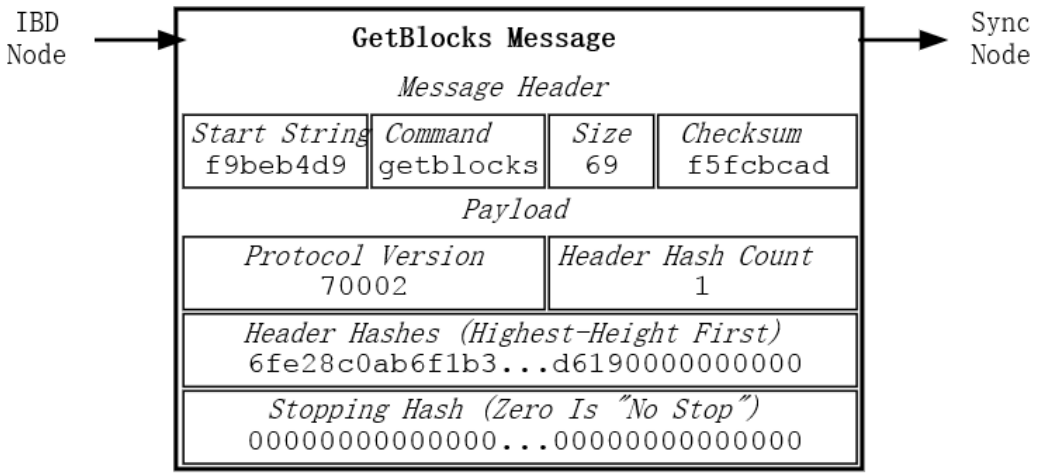
Inventory vectors

- getdata消息只能请求在inv消息中出现的区块
- block消息传输一个序列化的区块：
  - 80bytes blockheader，txn_count，txns

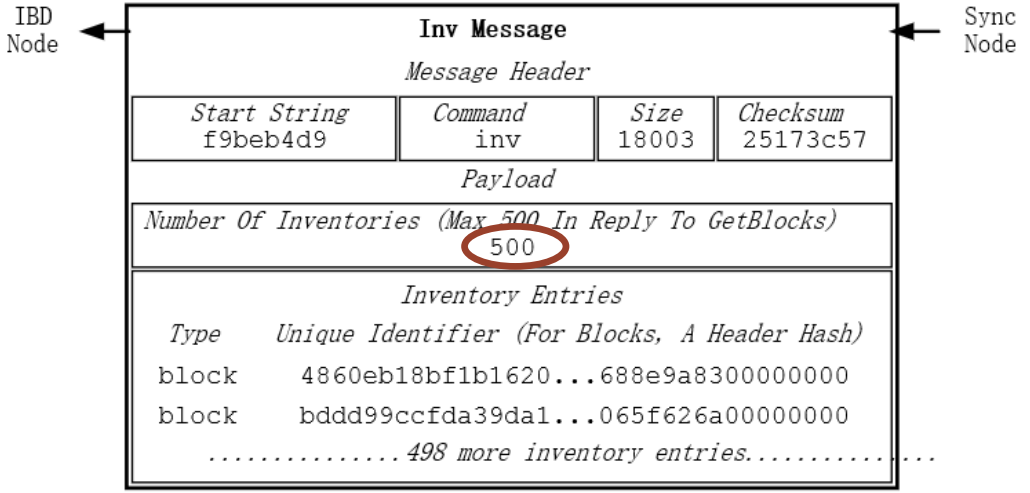
Blocks-First(BF)



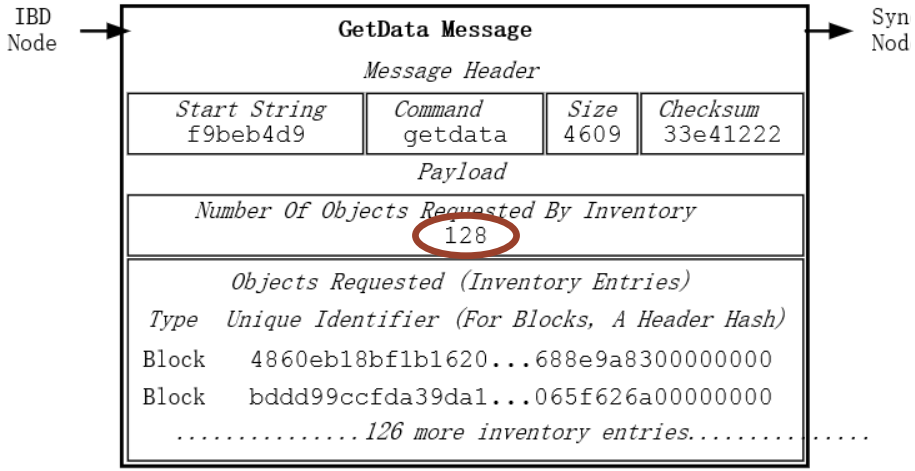
Overview Of Blocks-First Initial Blocks Download (IBD)



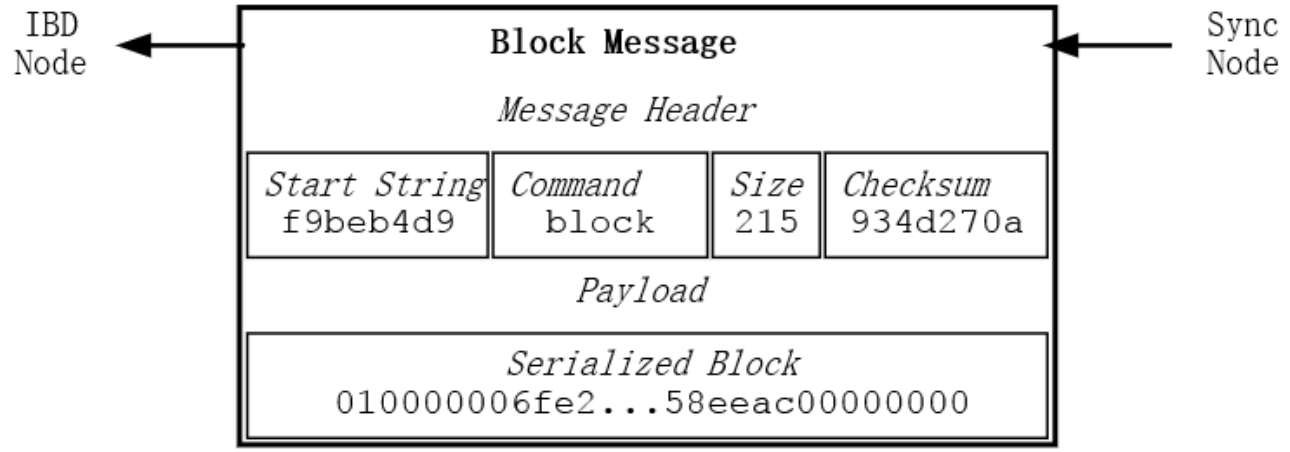
First getblocks message sent from Initial Blocks Download (IBD) node



First inv message reply sent to Initial Blocks Download (IBD) node



First getdata message sent from Initial Blocks Download (IBD) node



First block message sent to Initial Blocks Download (IBD) node

Blocks-First

- 最主要的**优点**是非常的简洁
- 最主要的**缺点**是IBD节点仅依赖于单一的同步节点

1：只依赖一个节点 → 速度受限制

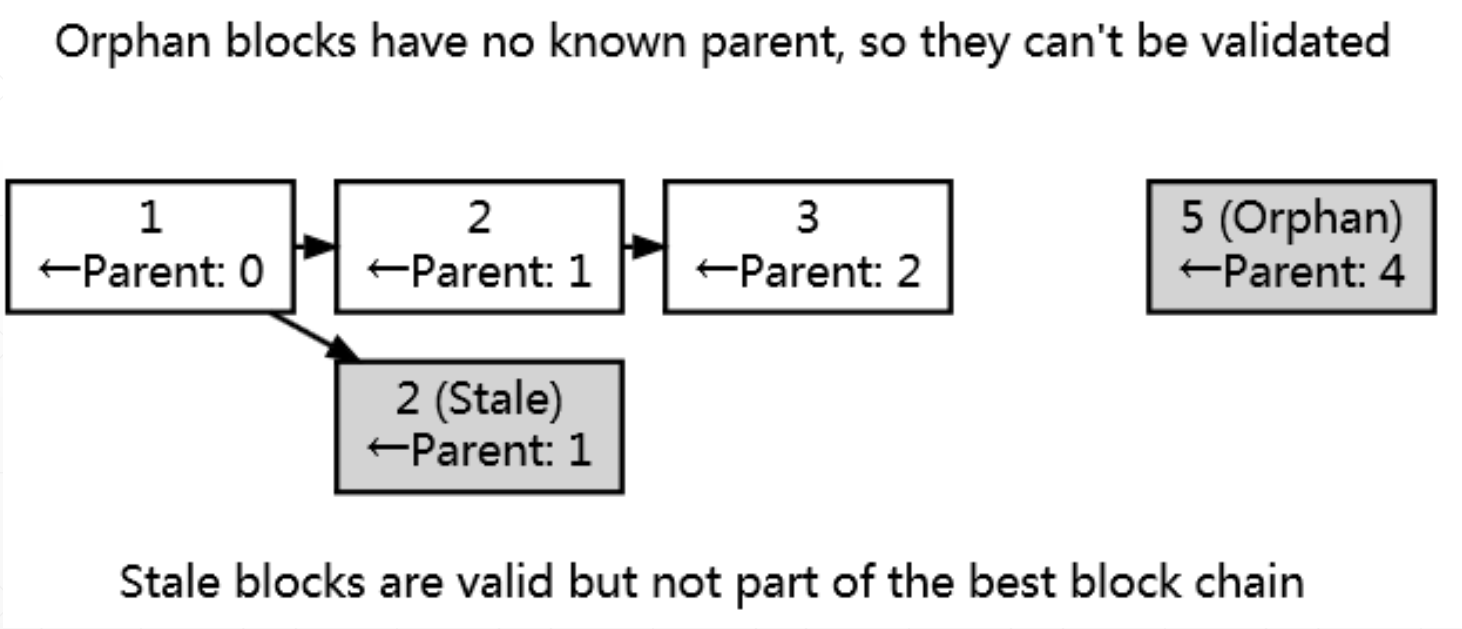
2：下载的不是最优区块链 → 下载重新开始

3：无用的区块占用磁盘 → 磁盘填充攻击（如果传送的是分叉链，也会先存下来，占用内存）

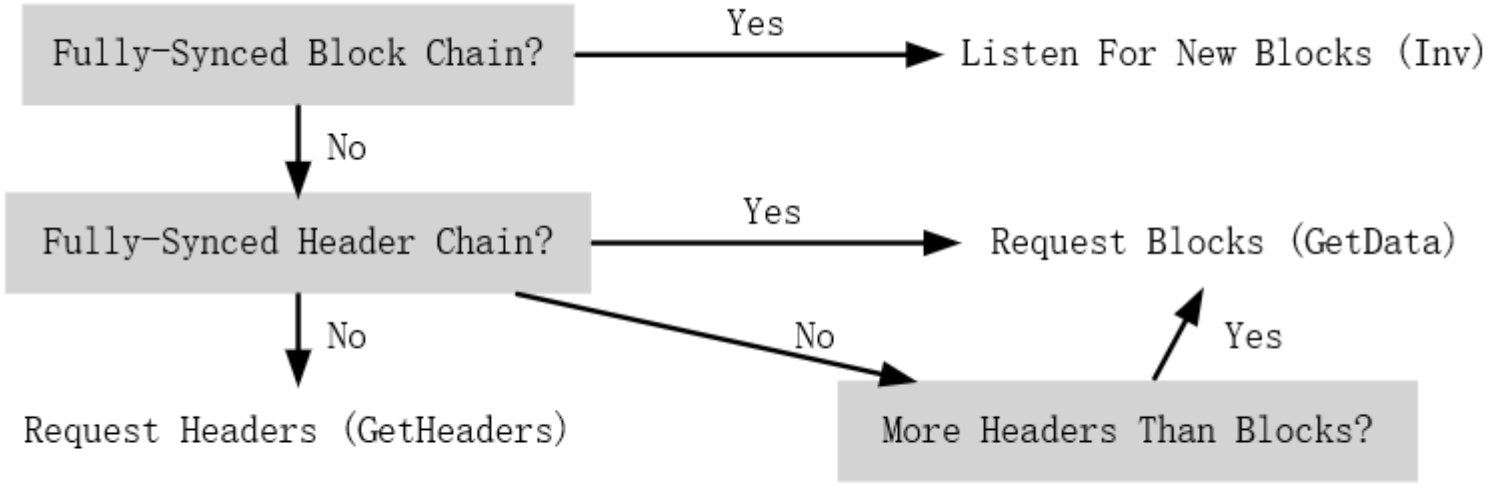
4：对等节点恶意不按顺序发送，**孤立块**增多 → 高内存使用率

孤立块

此区块的区块头的前一区块hash字段所引用的区块还没有接收到。换句话说，孤立块没有为人所知的父母。(stale block，他们知道父母，但他们不是主链区块链的一部分)。



Headers-First (HF)

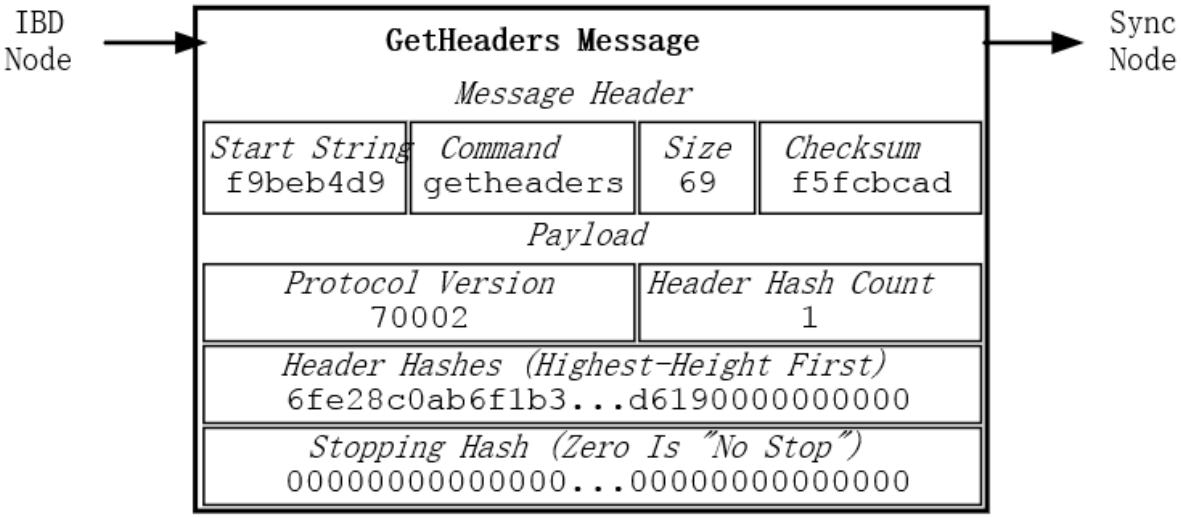


Overview Of Headers-First Initial Blocks Download (IBD)

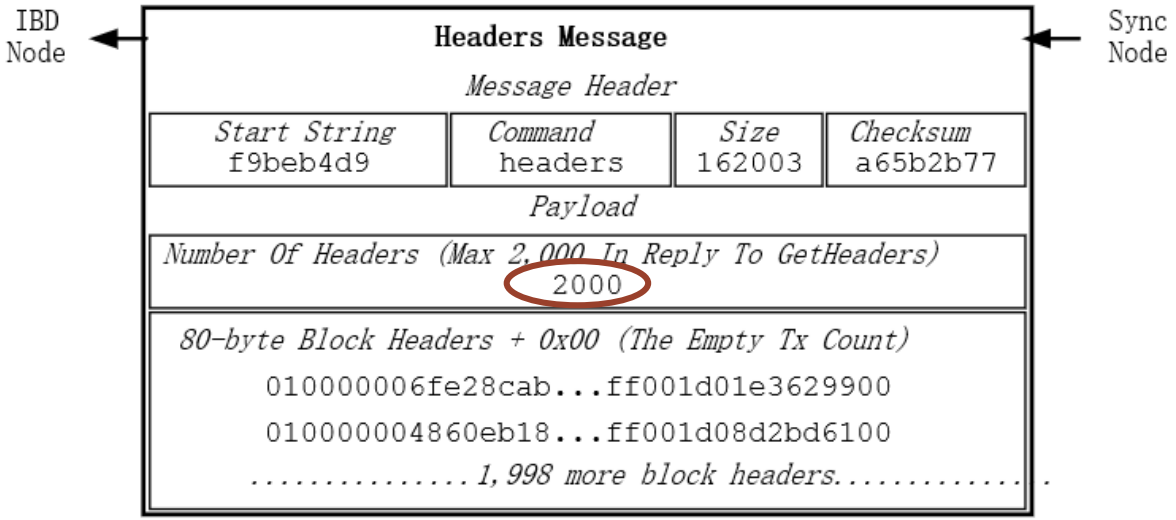
Field Size	Description	Data type	Comments
4	version	uint32_t	the protocol version
1+	hash count	var_int	number of block locator hash entries
32+	block locator hashes	char[32]	block locator object; newest back to genesis block (dense to start, but then sparse)
32	hash_stop	char[32]	hash of the last desired block header; set to zero to get as many blocks as possible (2000)

getheaders message

Headers-First



First getheaders message sent from Initial Blocks Download (IBD) node



First headers message reply sent to Initial Blocks Download (IBD) node

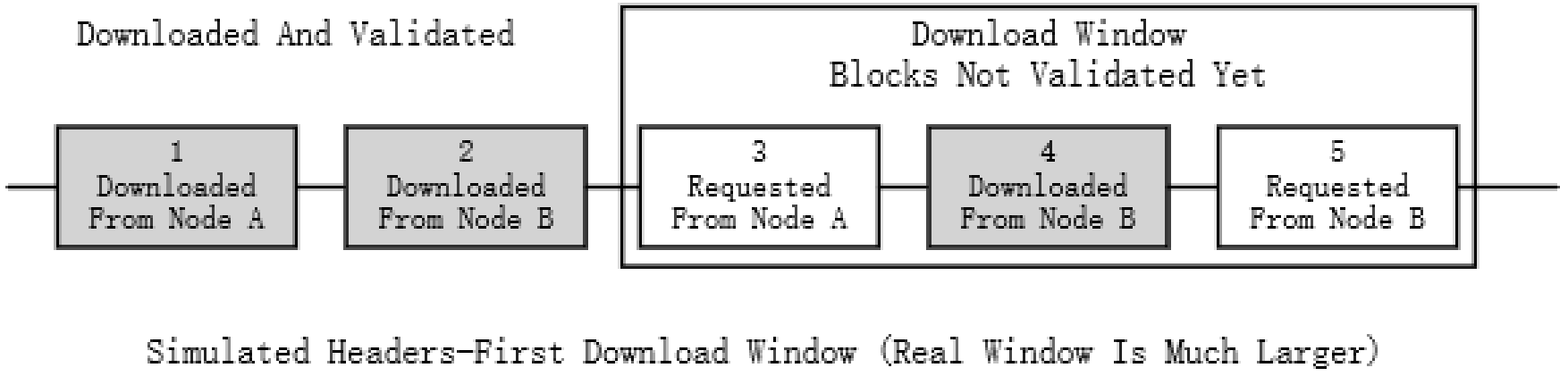
局部验证区块头确保所有的域都遵循共识规则，当IBD节点局部验证完区块头时，可以**并行的**做两件事

1：下载更多的区块头，直到headers message中的区块头hash少于2000个（意味着对等节点没有更多的区块头可提供）

2：下载区块，使用getdata消息向**任意一个全节点**请求

Headers-First

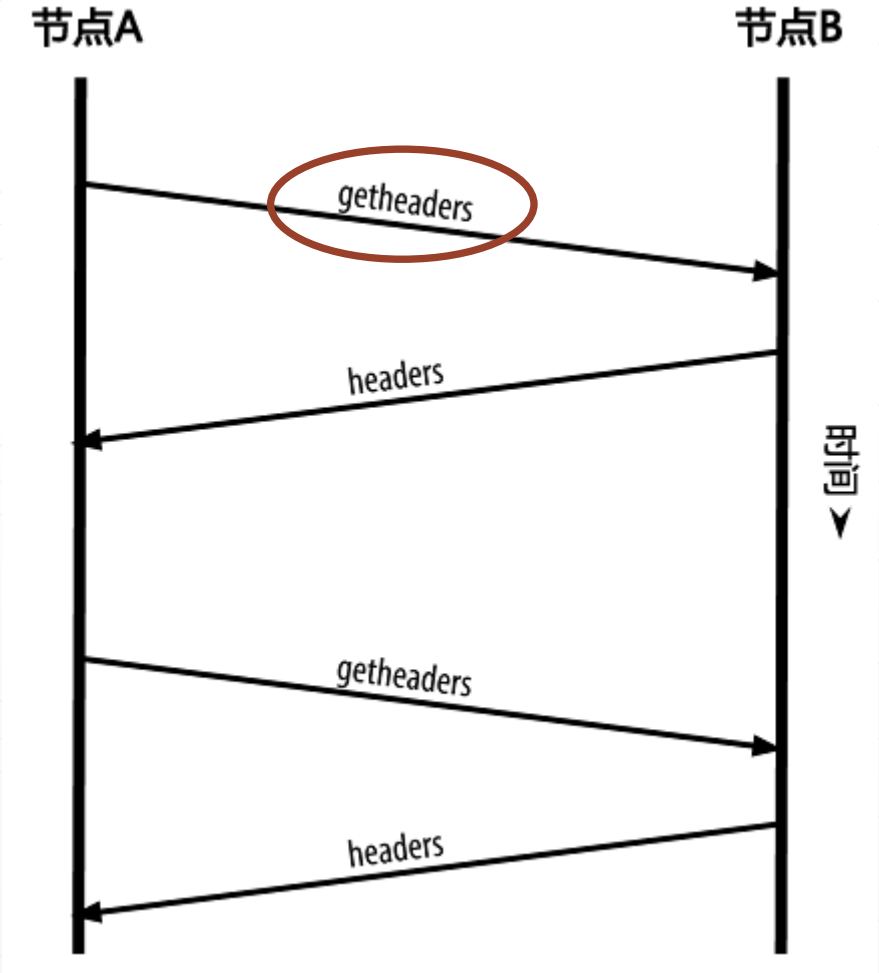
同一时间只能向每个对等节点请求最多16个区块，而且此节点最多从8个outbound对等节点下载。意味着在初始同步区块的时候同时只能请求最多128个块。



Simulated Headers-First Download Window (Real Window Is Much Larger)

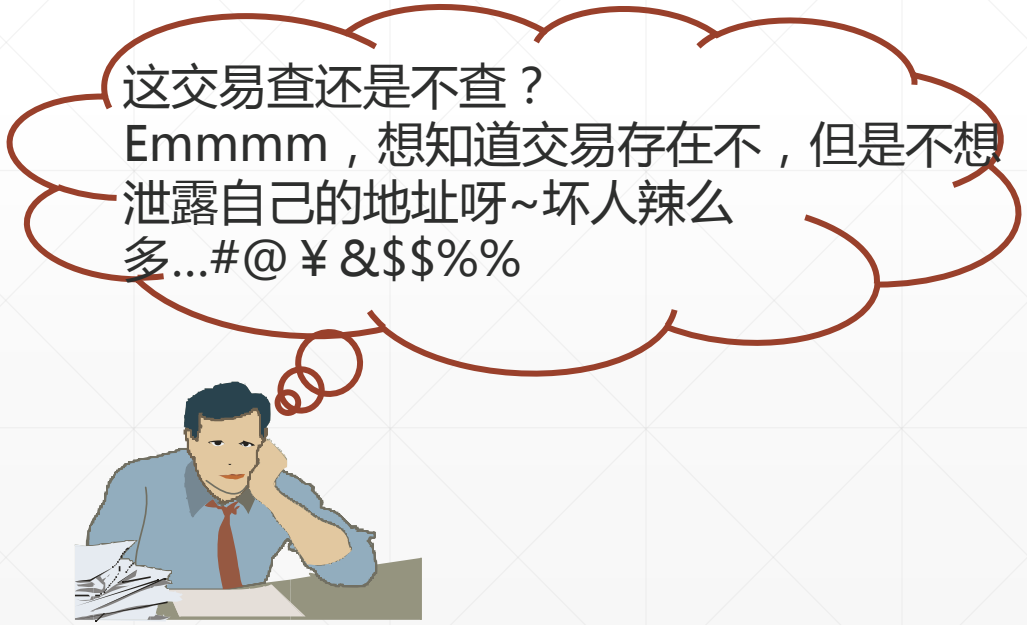
- 比特币核心的HF模式使用了一个1024区块的移动窗口去最大化下载速度
- 当一个对等节点对此窗口的移动拖延时间超过2秒就会断开此连接，尝试去连接一个更快的节点。

SPV节点交换“库存清单”



- SPV节点需要读取**特定交易**从而选择性地验证交易时，向对等节点发送一条getdata（获取区块？还是获取交易）的请求

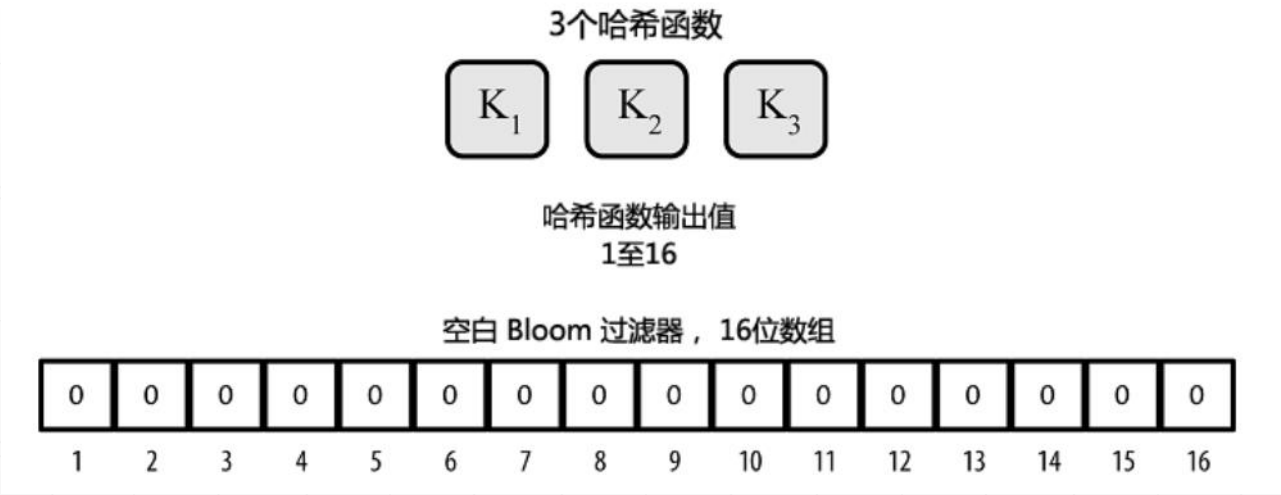
- SPV节点对特定数据的请求可能无意中透露了钱包里的地址信息，产生了**隐私风险**



Bloom过滤器

简介

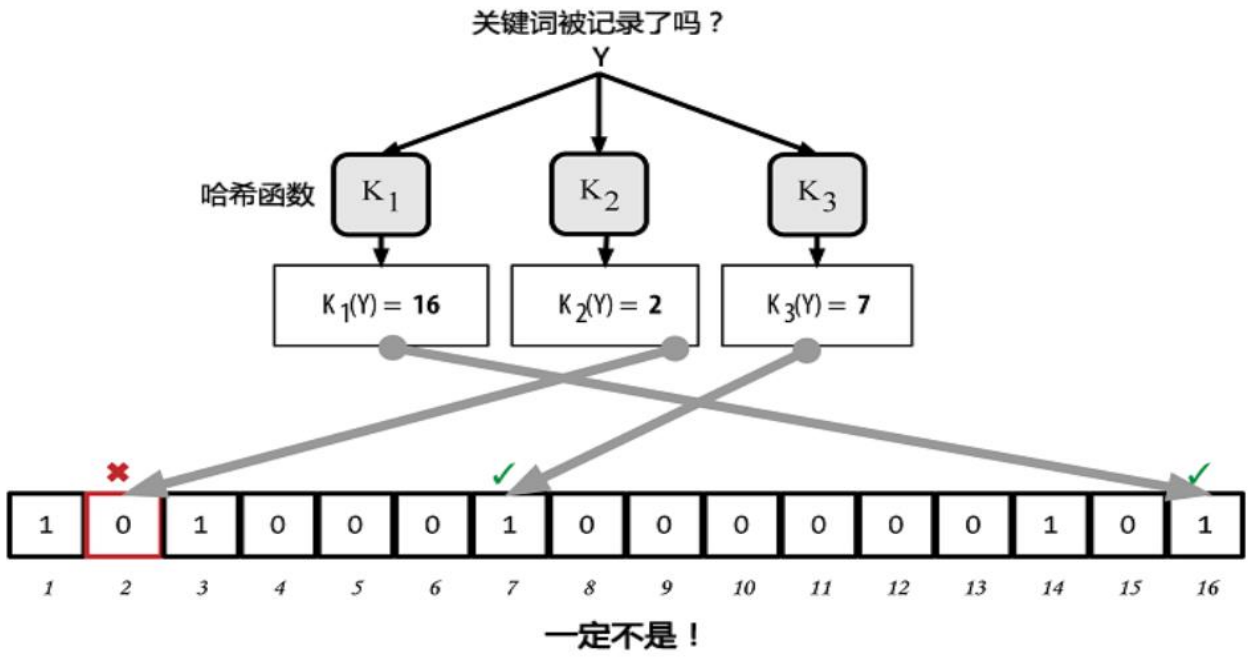
- 1970年由布隆提出的一种空间效率很高的随机**数据结构**，利用**位数组**很简洁地表示一个集合，并判断一个元素是否属于这个集合。
- Bloom过滤器能够**准确判断**一个元素**不在**集合内，但只能判断一个元素**可能在**集合内，是一个允许用户描述特定的关键词组合而不必精确表述的**基于概率**的过滤方法。

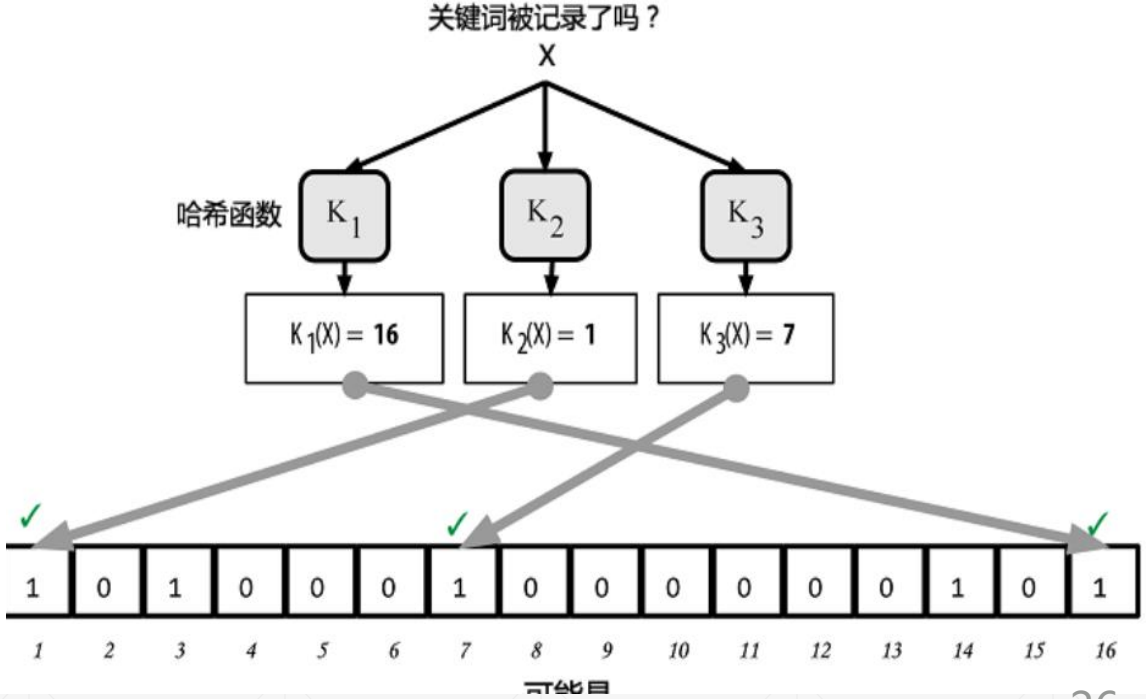


Bloom过滤器

实现

- 由一个可变长度（N）的二进制数组（N位二进制数成一个位域。max：36,000 bytes）和数量可变（M max：50）的一组哈希函数组成。
- 这些哈希函数的映射值始终在1和N之间，该数值与二制数组位置相对应。
- Bloom过滤器的准确性和私密性能通过改变长度（N）和哈希函数的数量（M）来调节。





SPV节点如何使用Bloom过滤器

1. SPV节点初始化一个不会匹配任何关键词的“空白”Bloom过滤器；每位都为0
2. SPV节点创建一个包含钱包中所有地址信息的列表，并创建一个与每个地址相对应的交易输出相匹配的搜索模式；（正常地址1，脚本哈希地址以3开头。搜索模式：，这里也是只用地址的前几位，避免泄露信息）
3. SPV节点把每一个搜索模式添加至Bloom过滤器里，这样只要关键词出现在交易中就能够被过滤器识别出来；
4. SPV节点向对等节点发送一条包含需在该连接中使用的过滤器的filterload消息，然后使用getdata请求交易相关信息；（inv消息有5种类型，type填MSG_FILTERED_BLOCK，32个字节的标识符是区块头hash值（为了方便对方快速找到某个区块里的交易？），inv消息类型被设置为filtered load ，）
5. 对等节点会用收到的Bloom过滤器来匹配交易的输出，只有符合的交易会传送至SPV节点；
6. 对等节点为回应来自SPV节点的getdata信息，会发出一条**merkleblock消息**，只含有和过滤器匹配的区块的**区块头信息**，以及与每个交易匹配的**merkle路径**，随后会发送一条相匹配的**交易的tx消息**
7. SPV节点抛弃不需要的数据，使用符合要求的数据去更新自己的UTXO集合和钱包余额；

MerkleBlock消息

Bytes	Name	Data Type
80	block header	block_header
4	transaction count	uint32_t
Varies	hash count	compactSize uint
Varies	hashes	char[32]
Varies	flag byte count	compactSize uint
Varies	flags	byte[]

01000000 ..... Block version: 1
82bb869cf3a793432a66e826e05a6fc3
7469f8efb7421dc88067010000000000 ... Hash of previous block's header
7f16c5962e8bd963659c793ce370d95f
093bc7e367117b3c30c1f8fdd0d97287 ... Merkle root
76381b4d ..... Time: 1293629558
4c86041b ..... nBits: 0x04864c * 256**(0x1b-3)
554b8529 ..... Nonce

07000000 ..... Transaction count: 7
04 ..... Hash count: 4

3612262624047ee87660be1a707519a4
43b1c1ce3d248cbfc6c15870f6c5daa2 ... Hash #1
019f5b01d4195ecbc9398fbf3c3b1fa9
bb3183301d7a1fb3bd174fcfa40a2b65 ... Hash #2
41ed70551dd7e841883ab8f0b16bf041
76b7d1480e4f0af9f3d4c3595768d068 ... Hash #3
20d2a7bc994987302e5b1ac80fc425fe
25f8b63169ea78e68fbaaefa59379bbf ... Hash #4

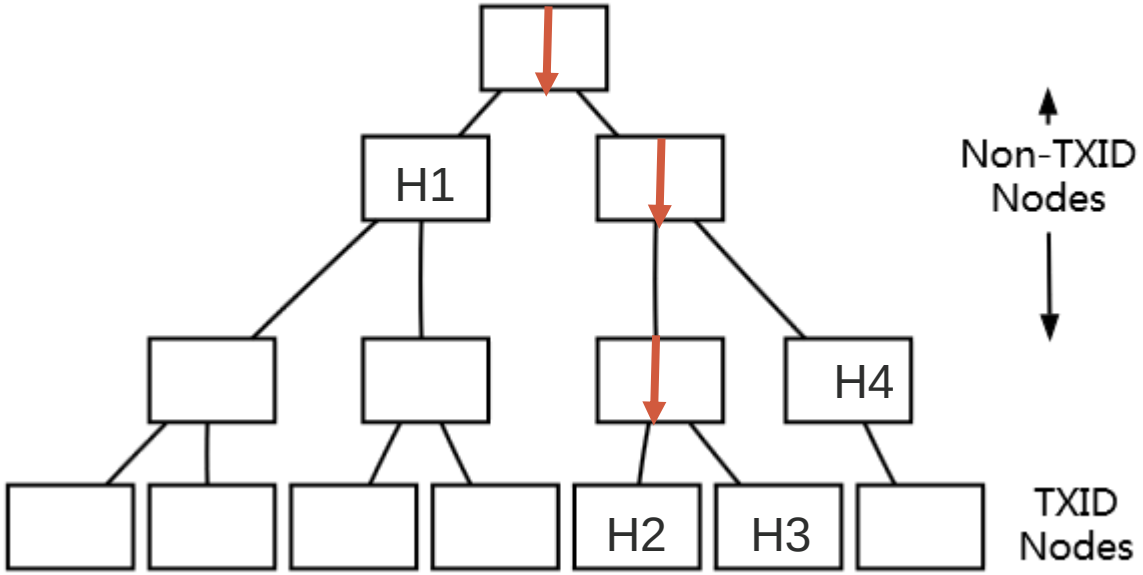
01 ..... Flag bytes: 1
1d ..... Flags: 1 0 1 1 1 0 0 0

Block header

SPV节点如何使用merkle路径验证交易存在性

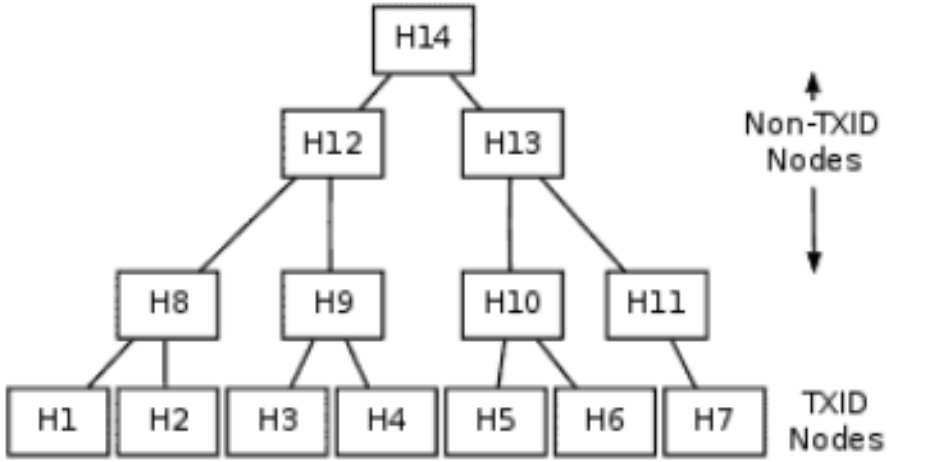
	TXID节点	非TXID节点
flag=0	使用hash 不是匹配节点	使用hash
flag=1	是匹配节点 使用hash	处理左孩子，此节点不做处理

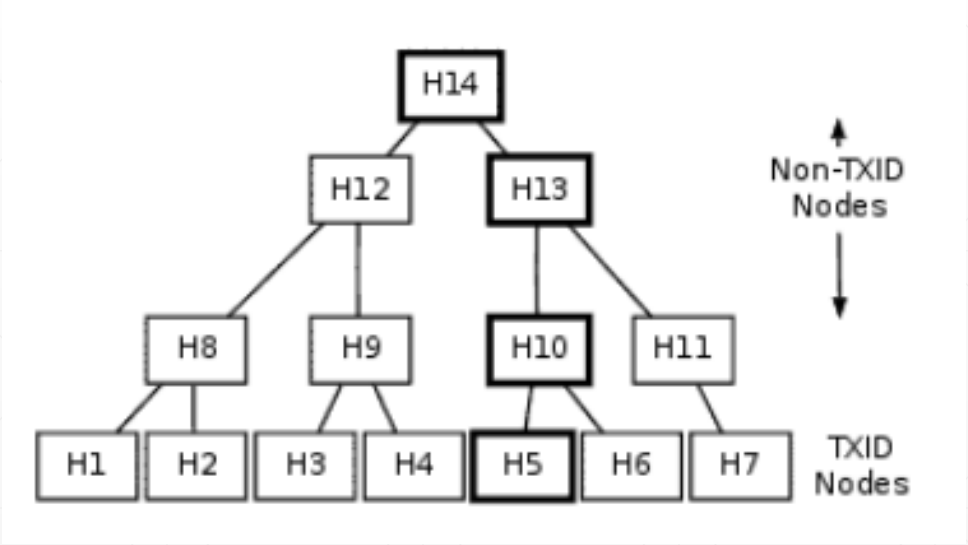
Flags: 1 0 1 1 1 0 0 0
Hashes: H1 H2 H3 H4



全节点如何行创建匹配交易的merkle路径

	TXID节点	非TXID节点
匹配交易或者匹配交易的祖先节点	flag->1 hash	flag->1 左孩子
不匹配交易	flag->0 hash	flag->0 hash





Flags: 1 0 1 1 1 0 0 0补位

Hashes: H_{12} H_5 H_6 H_{11}

加密和身份认证的连接

背景：

- 大多数比特币的新用户以为节点间的网络通信是加密的，但是事实上在最初的比特币通信实现上完全是in the clear，这对于**SPV节点**又是一个比较大的隐私问题。因此提出两种方法去提升P2P网络的隐私性和安全性。

1. Tor Transport（ The Onion Routing network ）

洋葱路由-→匿名网络

通过**随机网络路径**对数据进行加密和封装，具有**匿名、不可追踪、隐私性**

- 比特币核心提供了一些配置的选项从而允许用户通过tor网络来运行比特币节点和进行流量的传输。
- 此外还提供了tor隐藏服务允许其他运行tor的节点直接通过tor网络和自己的节点连接。

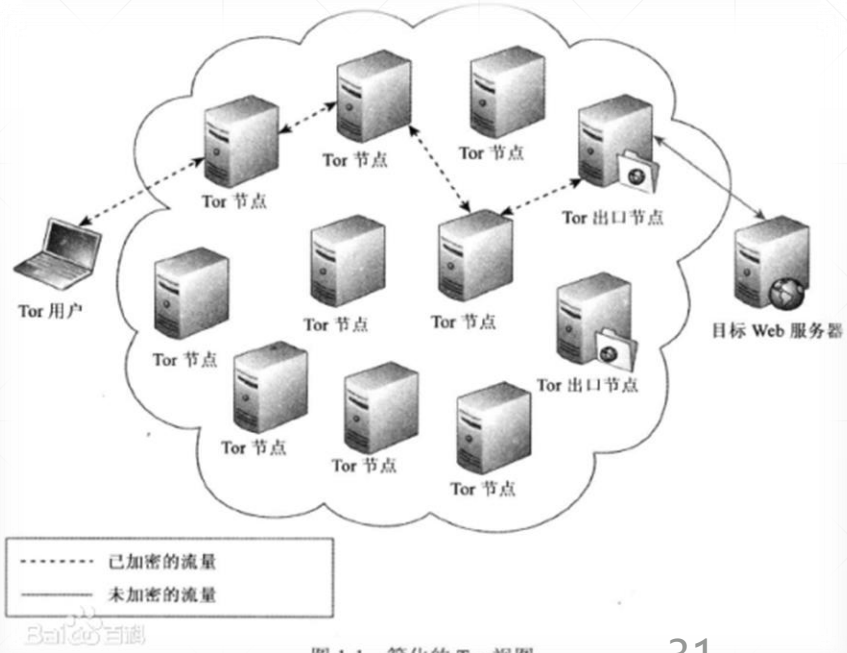
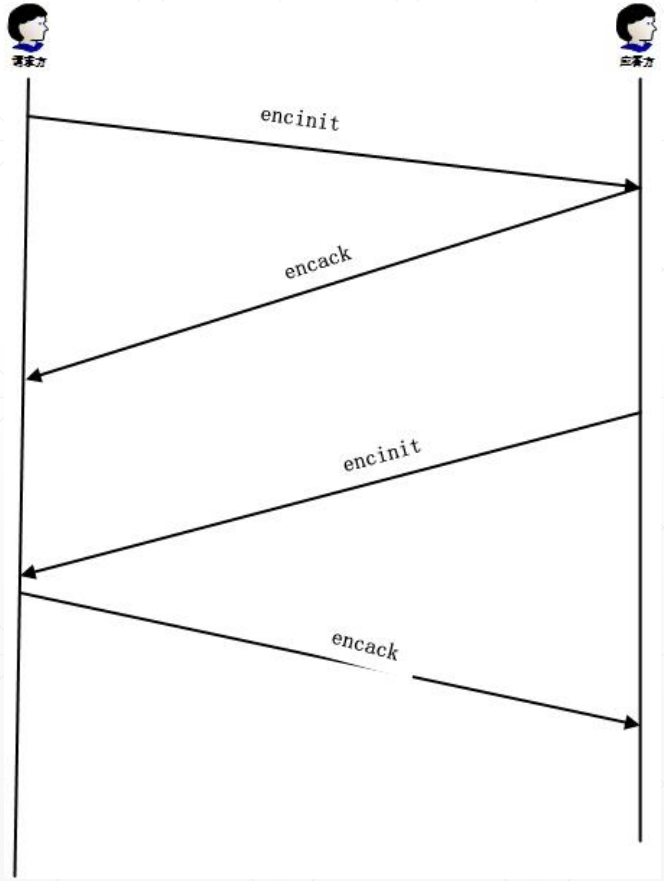


图 1-1 简化的 Tor 视图

2. Peer-to-Peer Authentication and Encryption

Encryption

- 对等通信进行加密将使分析特定的用户目标变得比当前更加困难。



Field Size	Description	Data type	Comments
33bytes	ephemeral-pubkey	comp.-pubkey	The session pubkey from the requesting peer
1bytes	symmetric key cipher type	int8	symmetric key cipher type to use

Possible symmetric key ciphers types

Number	symmetric key ciphers type
0	chacha20-poly1305@openssh.com

ChaCha20-Poly1305是由**ChaCha20流密码**和**Poly1305消息认证码**（MAC）结合的一种应用在互联网安全协议中的**认证加密算法**

Field Size	Description	Data type	Comments
33bytes	ephemeral-pubkey	comp.-pubkey	The session pubkey from the responding peer

2. Peer-to-Peer Authentication and Encryption

1. 用ECDH 密钥交换算法，通过双方的临时公钥生成ecdh_secret；
2. 用HKDF衍生出对称密钥对；
   - HKDF extraction PRK = HKDF_EXTRACT
     (hash=SHA256, salt="bitcoinecdh", ikm=ecdh_secret|cipher-type).
   - Derive Key1 K_1 = HKDF_EXPAND(prk=PRK, hash=SHA256, info="BitcoinK1", L=32)
   - Derive Key2 K_2 = HKDF_EXPAND(prk=PRK, hash=SHA256, info="BitcoinK2", L=32)

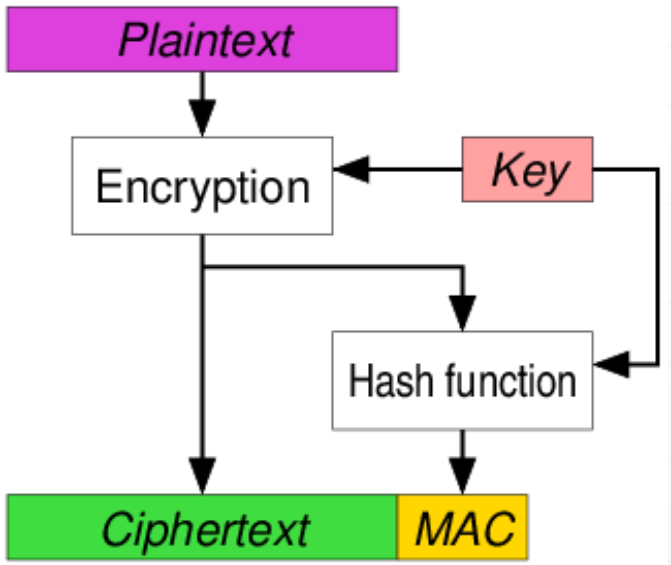
   K_1仅被用来加密消息有效载荷的长度从而避免通过暴露长度信息泄露相关信息。

   K_2 必须与poly1305一起使用以建立一个AEAD。

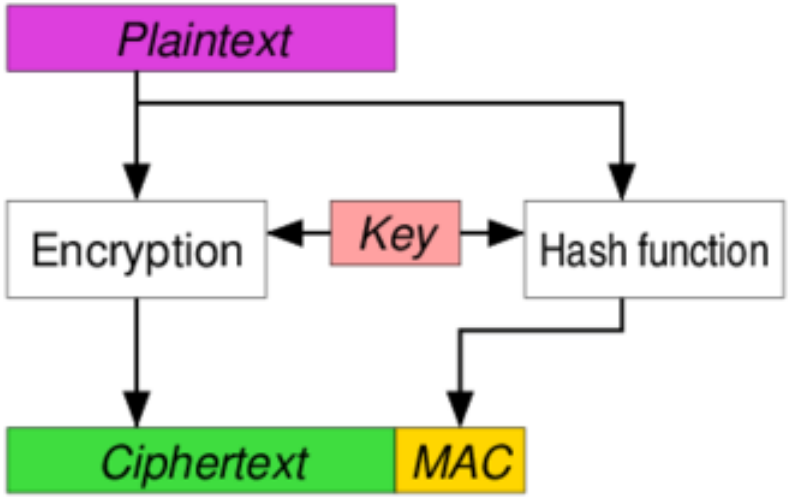
2. Peer-to-Peer Authentication and Encryption

AEAD(authenticated encryption with associated data)

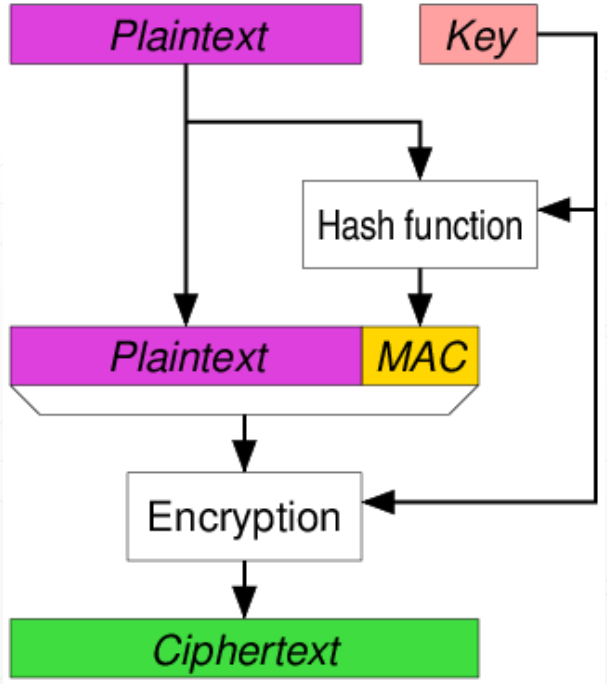
通过相关数据进行身份加密是一种加密形式，同时提供数据的**机密性**、**完整性**和**真实性**保证。



EtM



E&M



MtE

2. Peer-to-Peer Authentication and Encryption

Authentication
给对等节点提供一种身份认证的方法，保证节点的所有权，从而允许对等节点访问一些额外的或者有限的节点服务。

身份认证

其他概念介绍

交易池	交易孤立池	UTXO池
未确认（未被打包进区块）交易的临时列表。追踪记录那些被网络所知晓、但还未被区块链所包含的交易（收到的还未被打包进区块的交易）	一个交易的输入引用某未知的交易，此孤立交易就会被暂时储存在孤立交易池中直到某未知交易（如父交易）的信息到达	区块链中所有未支付交易输出（UTXO）的集合
初始化时为空，不断的从网络上接收到的交易来填充		初始化时不为空，包括到genesis块的utxo
只包含**未确认交易**		只包含**已确认交易**
存储在本地内存而不是持久存储，随着接连不断到来的交易动态填充		存储在本地内存/作为一个包含索引的数据库表安置在永久性存储设备中。
代表的是单个节点的本地视角。取决于节点的启动时间或重启时间，不同节点的两池内容可能有很大差别		代表网络的共识，因此，不同节点间UTXO池的内容差别不大

警告消息：是比特币的"紧急广播系统"，比特币核心开发人员可以借此功能给所有比特币节点发送紧急文本消息。这一功能是为了让核心开发团队将比特币网络的严重问题通知所有的比特币用户。例如一个需要用户采取措施的的严重bug。

Thanks for your attention !

Q&A