海联金汇 HYUNION HOLDINGS
股票代码：002537

UMF 联动优势 UNION MOBILE FINANCIAL TECHNOLOGY
海联金汇旗下企业

科技赋能普惠金融

比特币探针实践分享

技术发展部 - 王宇

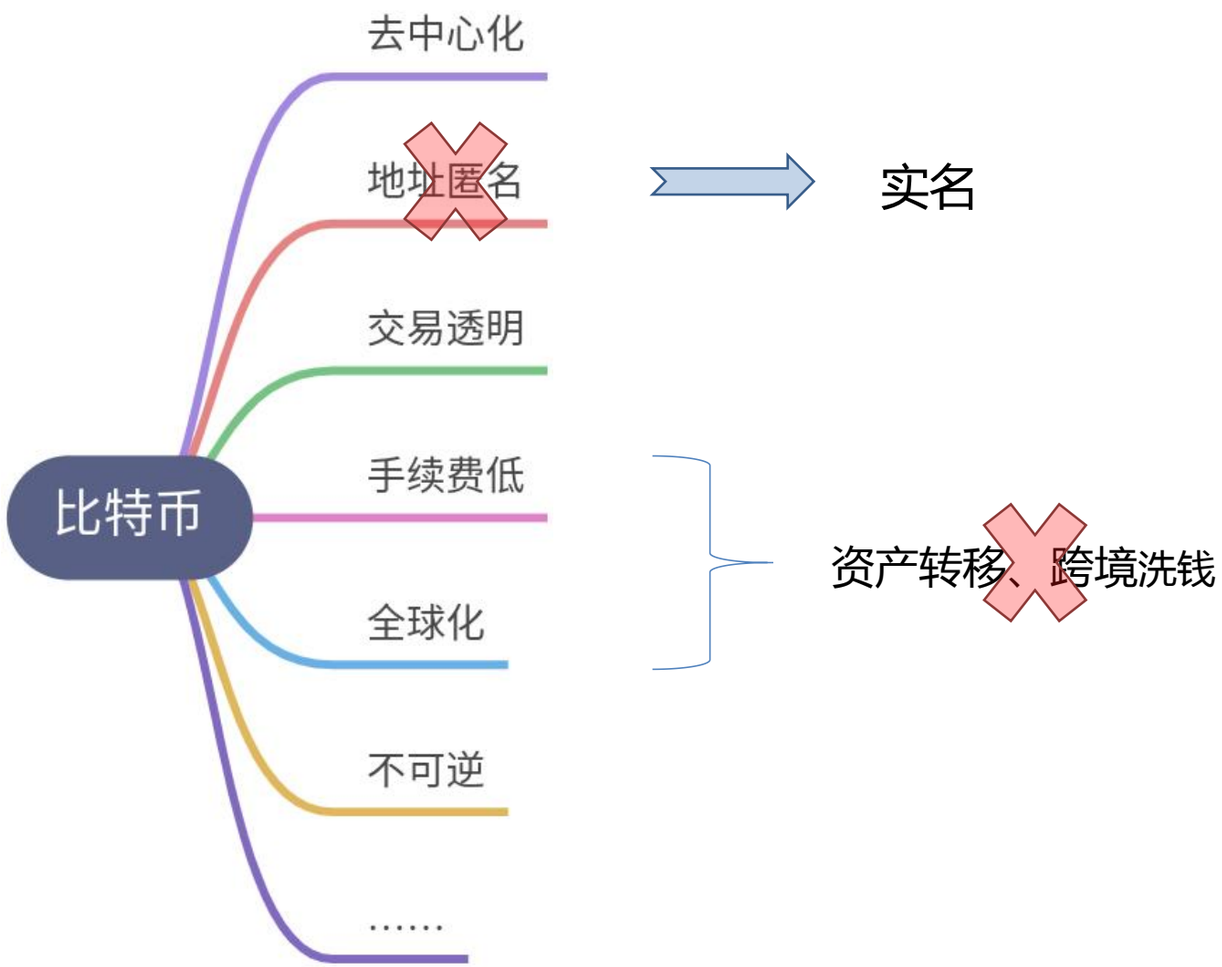
2020-06-04

MAKE FINTECH EVERYWHERE

概述

比特币有很多特性

目标



比特币探针的两个小目标：

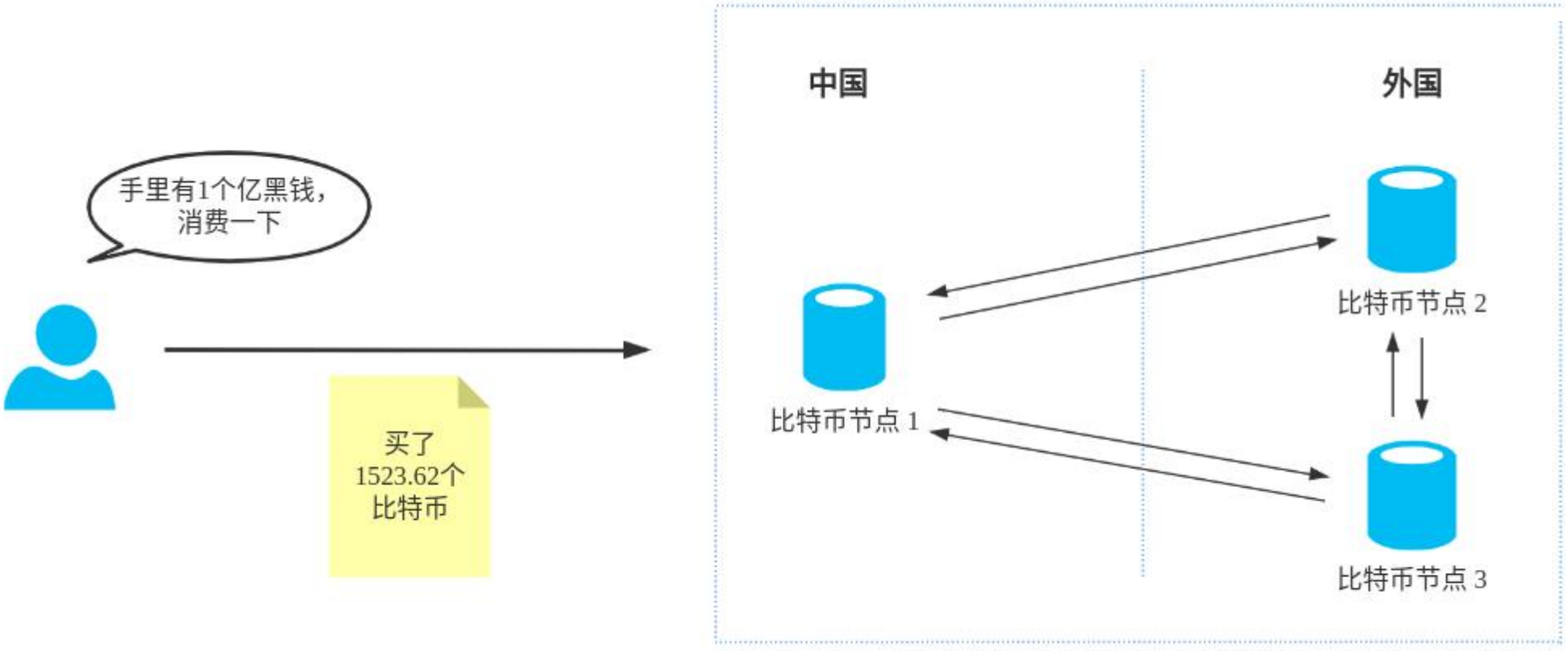
- 挑战比特币的匿名性
- 从交易中识别出跨境交易

今天分享的两部分内容：

- 两个小目标的实现思路
- bitnodes的代码实现

场景

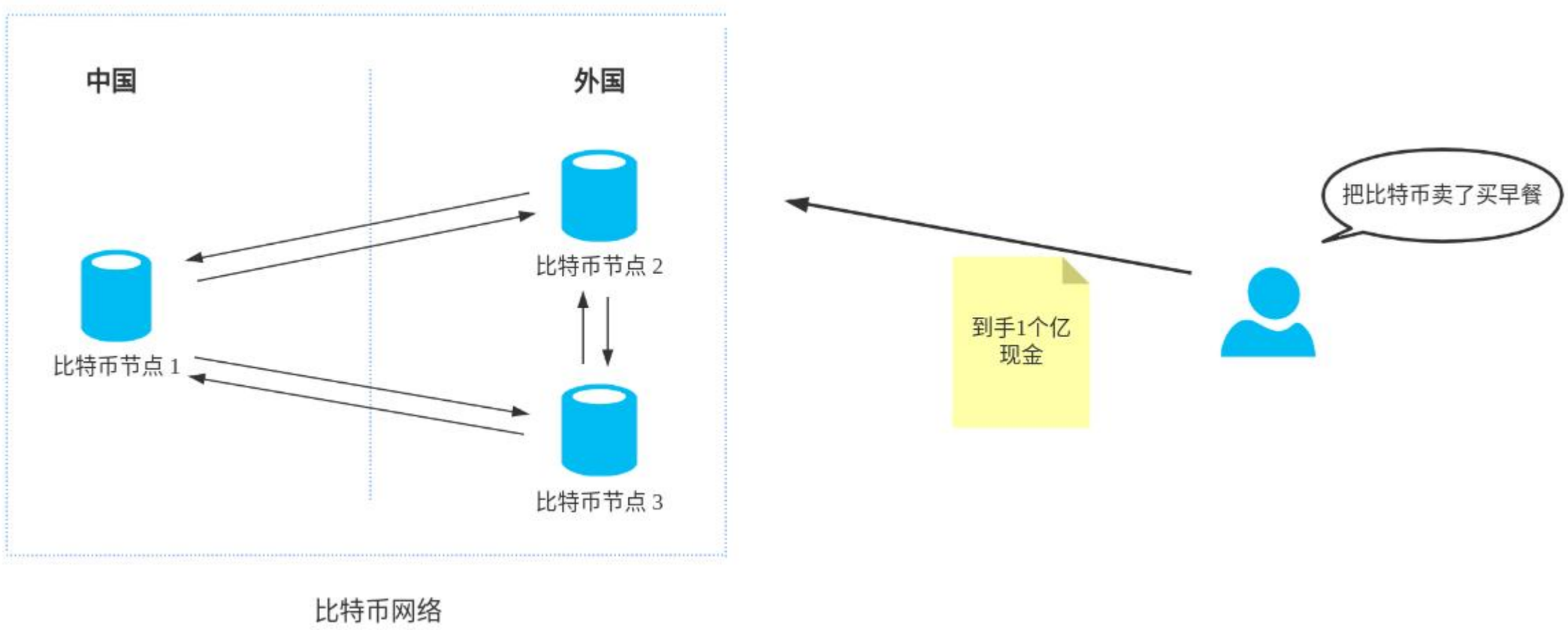
6月1号 儿童节



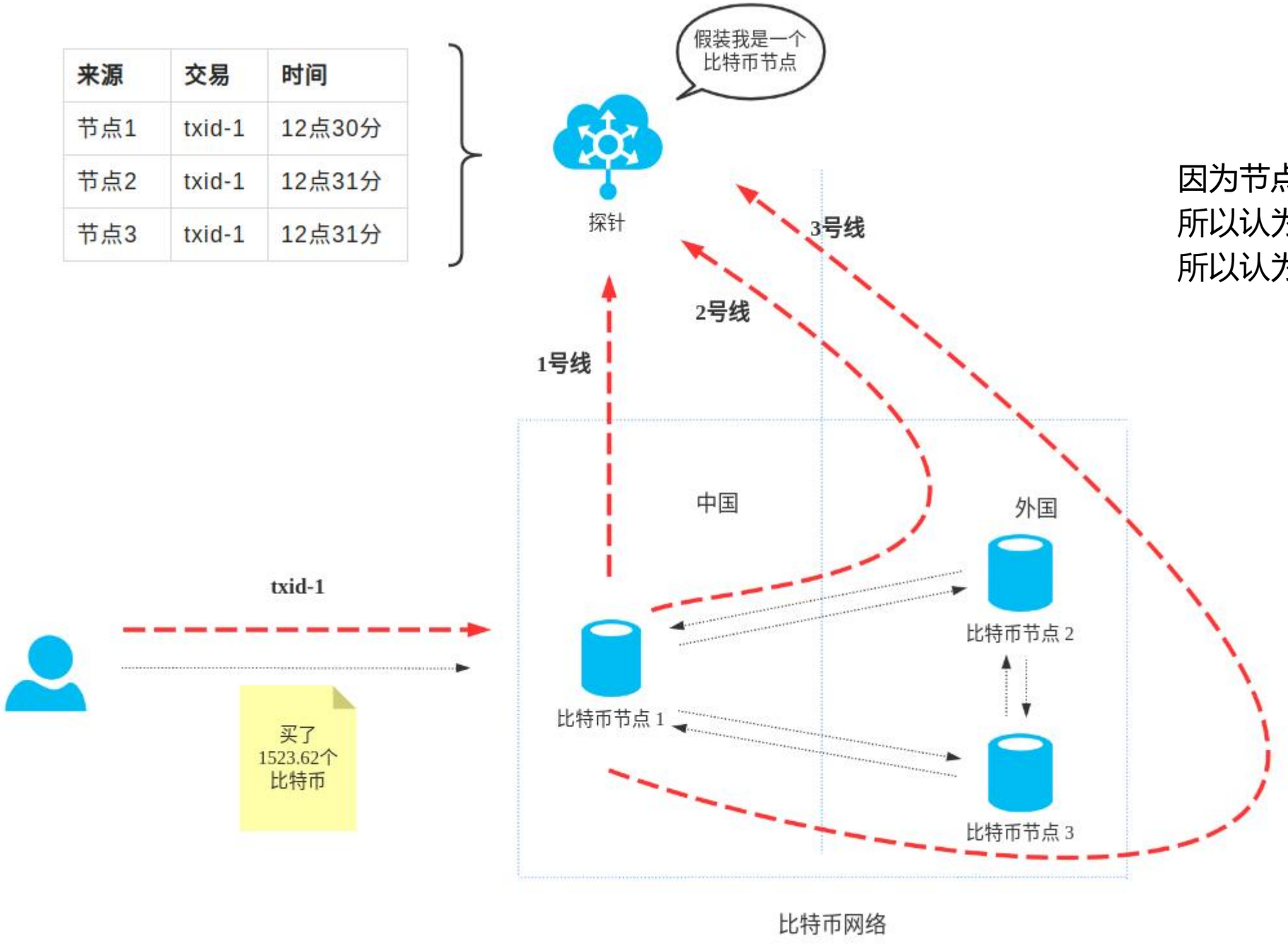
比特币网络

场景

6月2号 儿童节第二天



来源	交易	时间
节点1	txid-1	12点30分
节点2	txid-1	12点31分
节点3	txid-1	12点31分

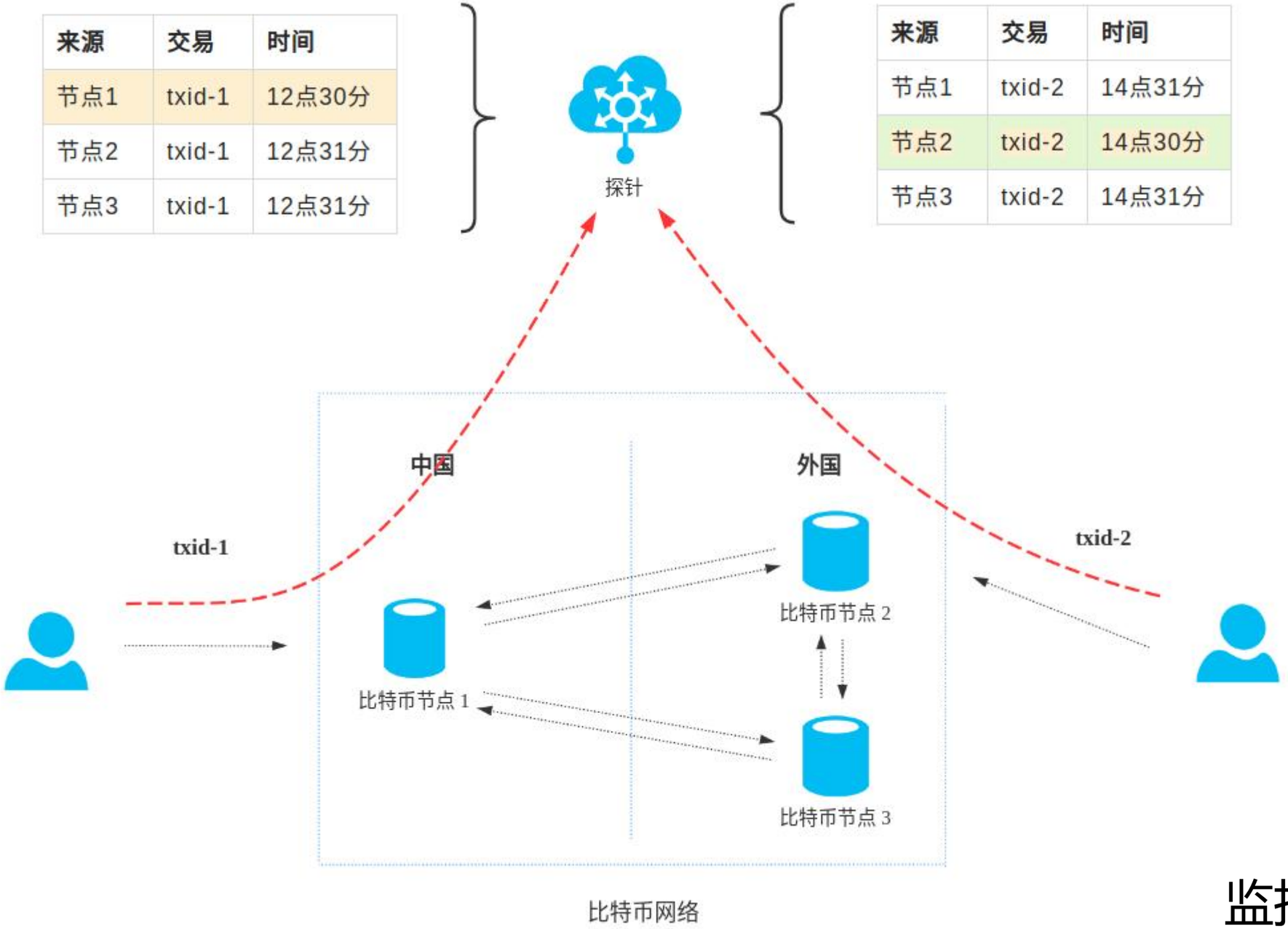


因为节点1在中国
所以认为交易的发起地点是中国
所以认为交易的发起人在中国

监控到节点数据

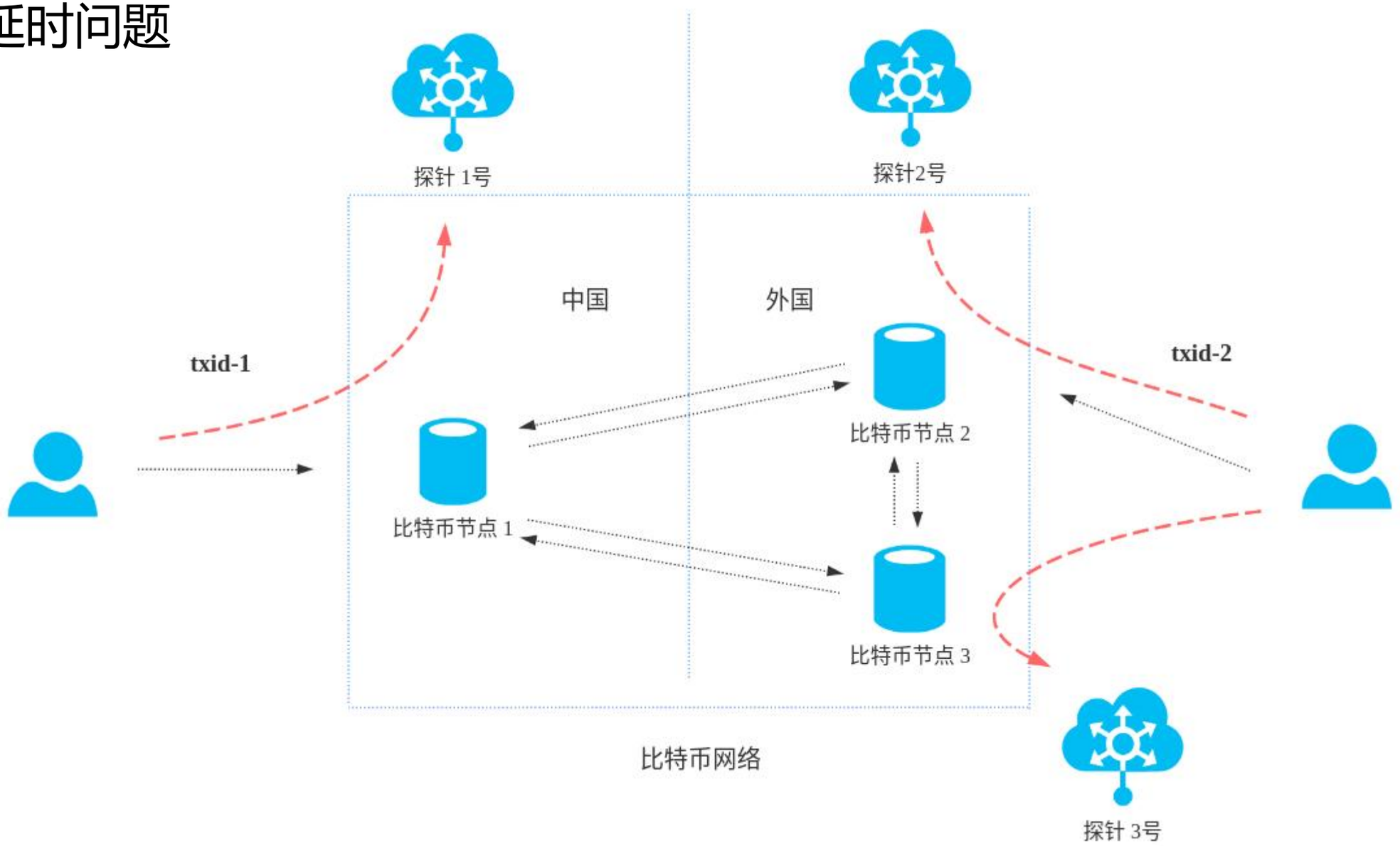
来源	交易	时间
节点1	txid-1	12点30分
节点2	txid-1	12点31分
节点3	txid-1	12点31分

来源	交易	时间
节点1	txid-2	14点31分
节点2	txid-2	14点30分
节点3	txid-2	14点31分



监控到节点数据

网络延时问题



整体思路

买比特币的交易

来源	交易	时间
节点1	txid-1	12点30分

中国

买方	卖方	买回来
address-1	无所谓	1523.62 BTC

中国 —— 节点1 —— txid-1 —— address-1

卖比特币的交易

来源	交易	时间
节点2	txid-2	14点30分

外国

买方	卖方	卖出去
无所谓	address-1	1523.62 BTC

外国 —— 节点2 —— txid-2 —— address-1

整体思路

跨境交易的几种可能：

单笔交易 —— 国内转国外

两笔交易 —— 国内转国内，国外转国外
- 连续两笔交易
- 不连续两笔交易

多笔交易 —— ...

整体思路

全网广播的交易数据大概是：

估算1：

一天 20万 笔交易，一个探针监控 3000 个节点，一共 6亿 条数据

估算2：

一个块确定下来的交易数量 2000 多，平均 2000/10分钟/60秒 = 30笔交易/ 秒

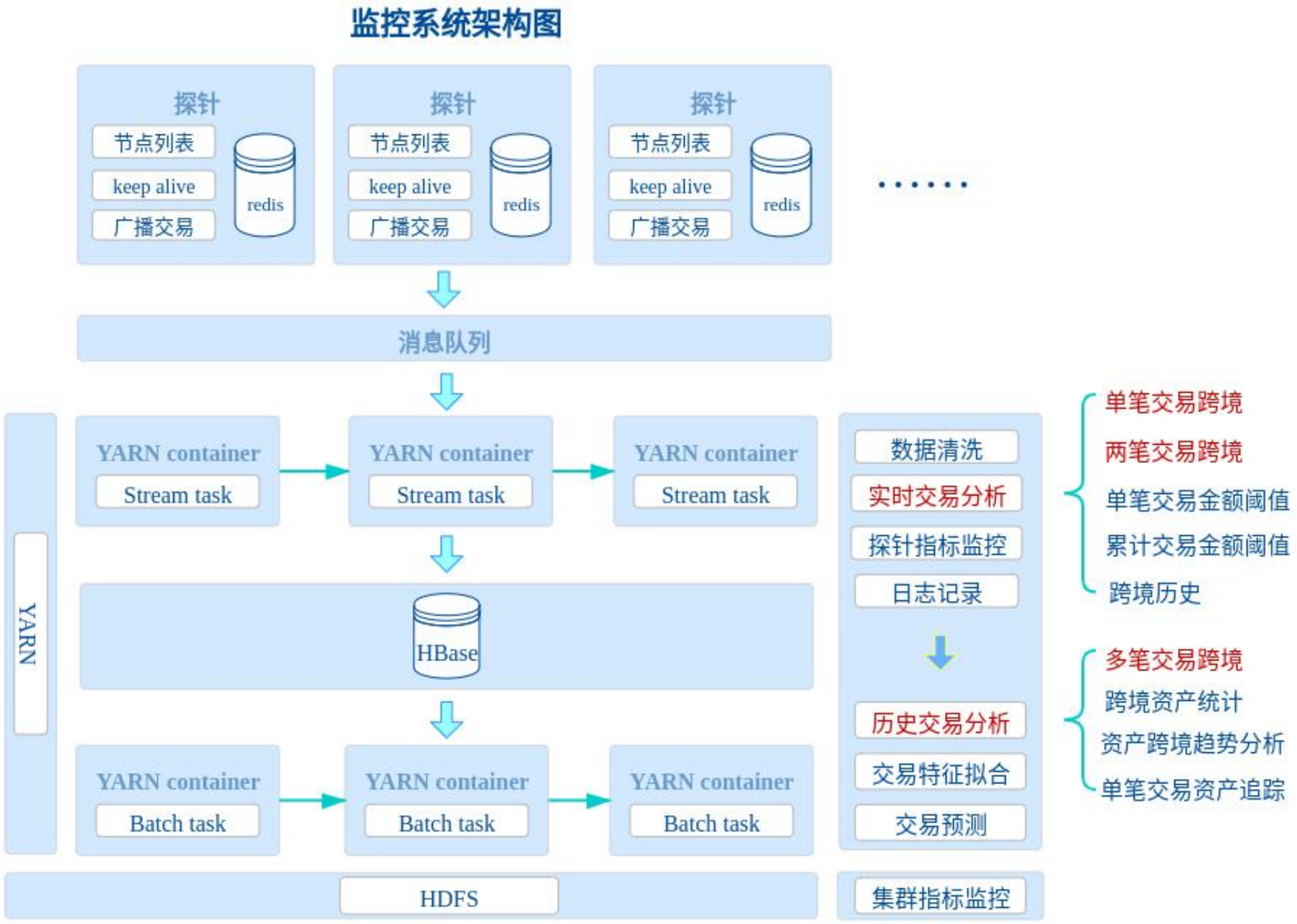
3000 个节点，一个探针每秒会收到 3000x30 = 90000 笔交易广播

一个探针一天收到的数据量是 90000 x 60 x 60 x 24 = 72 0000 0000 笔交易

测试：

我们机器上的探针 48 小时收到 2000万 次交易广播，14万 笔交易

整体思路


监控系统架构图
探针
节点列表
keep alive
广播交易
redis
探针
节点列表
keep alive
广播交易
redis
探针
节点列表
keep alive
广播交易
redis
.....
消息队列
YARN
YARN container
Stream task
YARN container
Stream task
YARN container
Stream task
HBase
YARN container
Batch task
YARN container
Batch task
YARN container
Batch task
HDFS
数据清洗
实时交易分析
探针指标监控
日志记录
历史交易分析
交易特征拟合
交易预测
集群指标监控
单笔交易跨境
两笔交易跨境
单笔交易金额阈值
累计交易金额阈值
跨境历史
多笔交易跨境
跨境资产统计
资产跨境趋势分析
单笔交易资产追踪

bitnodes

7个核心脚本：

- procotol.py —— 核心工具类，消息序列化、反序列化
- crawler.py —— 获取节点列表和块高度
- ping.py —— 维护和节点的通信
- resolve.py —— 解析出节点的国家等信息
- export.py —— 导出数据到文件
- seeders.py —— 生成.zone配置文件
- pcap.py —— 分析广播消息

protocol

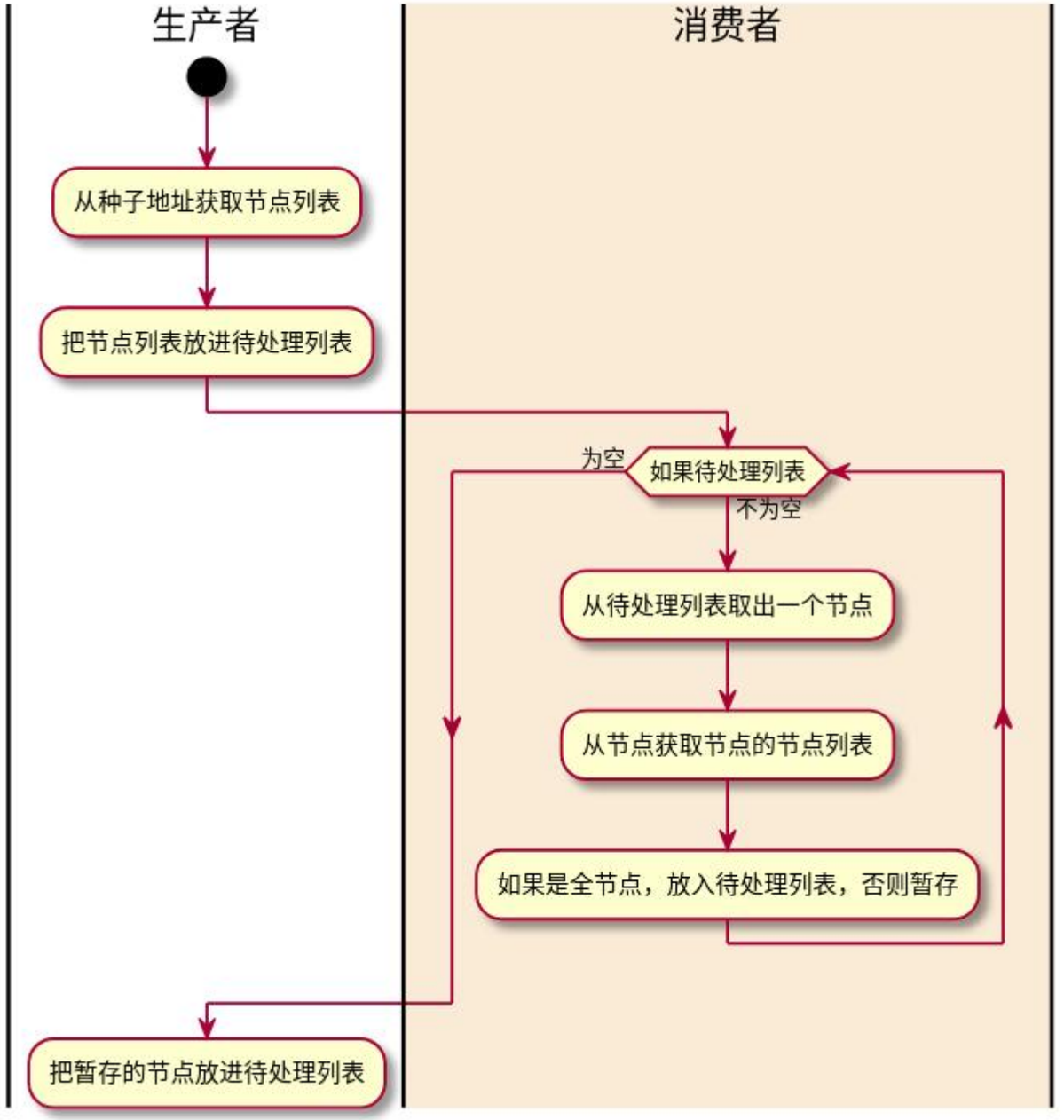
比特币协议的消息类型、报文格式：


```
0000   F9 BE B4 D9 76 65 72 73  69 6F 6E 00 00 00 00 00   ....version.....
0010   55 00 00 00 9C 7C 00 00  01 00 00 00 00 00 00 00   U....|..........
0020   E6 15 10 4D 00 00 00 00  01 00 00 00 00 00 00 00   ...M............
0030   00 00 00 00 00 00 00 00  00 00 FF FF 0A 00 00 01   ................
0040   20 8D 01 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
0050   00 00 00 00 FF FF 0A 00  00 02 20 8D DD 9D 20 2C   .......... ... ,
0060   3A B4 57 13 00 55 81 01  00                       :.W..U...
```

```python
def serialize_version_payload(self, to_addr, from_addr):
    payload = [
        struct.pack("<i", self.protocol_version),
        struct.pack("<Q", self.from_services),
        struct.pack("<q", int(time.time())),
        self.serialize_network_address(to_addr),
        self.serialize_network_address(from_addr),
        struct.pack("<Q", random.getrandbits(64)),
        self.serialize_string(self.user_agent),
        struct.pack("<i", self.height),
        struct.pack("<?", self.relay),
    ]
    return ''.join(payload)
```

crawler

整体流程：



crawler

version消息返回了节点的块高度，getaddr消息返回了节点的节点列表
crawler这个文件最后生成的数据格式：

```
1 [address, port, services, height]
2 ["83.248.118.103", 8333, 1033, 630403]
```

比特币节点的地址有三类：

ipv4： 10.10.144.72

ipv6: 2001:DB8:2de::e13

onion: http://4mtu5pl6yp3fmvny.onion

Value	Name	Description
1	NODE_NETWORK	This node can be asked for full blocks instead of just headers.
2	NODE_GETUTXO	See BIP 0064
4	NODE_BLOOM	See BIP 0111
8	NODE_WITNESS	See BIP 0144
1024	NODE_NETWORK_LIMITED	See BIP 0159

crawler

1. crawler需要多少生产者和消费者？

程序默认是 3 个生产者，2000个消费者
crawler获取到的主要信息是块高度，所以要保证在一个块的时间内遍历所有节点
实际过程中待处理列表里的节点数量在 25000 左右

2. 多线程消费同一队列，如何保证不重复消费？

ping

ping脚本主要维持和所有比特币节点的连接

向节点发送的消息类型：

ping　判断节点是否可连通

inv　发送最新的块信息给其他节点

addr　发送节点列表给其他节点，3小时不发其他节点就会忽略你

其他：

目前节点数6000左右，可以用10000个线程去处理

源程序把节点地址用CIDR的格式存下来了，但是没有使用

ping脚本最终结果：

维护和6000个节点的连接，并且把处于维护状态的节点存到一个列表里

resolve export

resolve脚本

从 maxmind 数据库拿数据，拿到ip对应的国家信息

遍历 ping 脚本在维护的处于连接状态的节点列表，匹配信息

export脚本

导出 resolve 的结果到文件

其他：

源程序用了 Redis 的 channel，有没有替代方案？

```
[[
 "35.167.136.109",              | node
  8333,
  70015,                        | version
  "/Satoshi:0.17.1/",           | user_agent
  1589429340,                   | last_ping
  1037,                         | services
  630300,                       | 块高度
  "us-west-2.amazonaws.com",    | hostname
  "Boardman",                   | city
  "US",                         | country
  45.8491,
  -119.7143,
  "America/Los_Angeles",        | timezone
  "AS16509",                    | asn
  "AMAZON-02"                   | org
 ]]
```

seeder

```
$ORIGIN seed.bitnodes.io.
$TTL 60

@ 28800 IN  SOA ns1.bitnodes.io. hostmaster.bitnodes.io. (
        1590043904 ; SERIAL
        3600 ; REFRESH
        1800 ; RETRY
        604800 ; EXPIRE
        60 ; MINIMUM
        )

@ 1800  IN  NS  ns1.bitnodes.io.
@ 1800  IN  NS  ns2.bitnodes.io.
@ 1800  IN  NS  ns3.bitnodes.io.
@ 1800  IN  NS  ns4.bitnodes.io.
@ 1800  IN  NS  ns5.bitnodes.io.

ns1 1800  IN  A 88.99.167.175
ns1 1800  IN  AAAA  2a01:4f8:10a:37ee::2
```

seeder脚本根据 export脚本的结果，生成了很多 .zone 文件

这些文件对我们来说没什么用

pcap

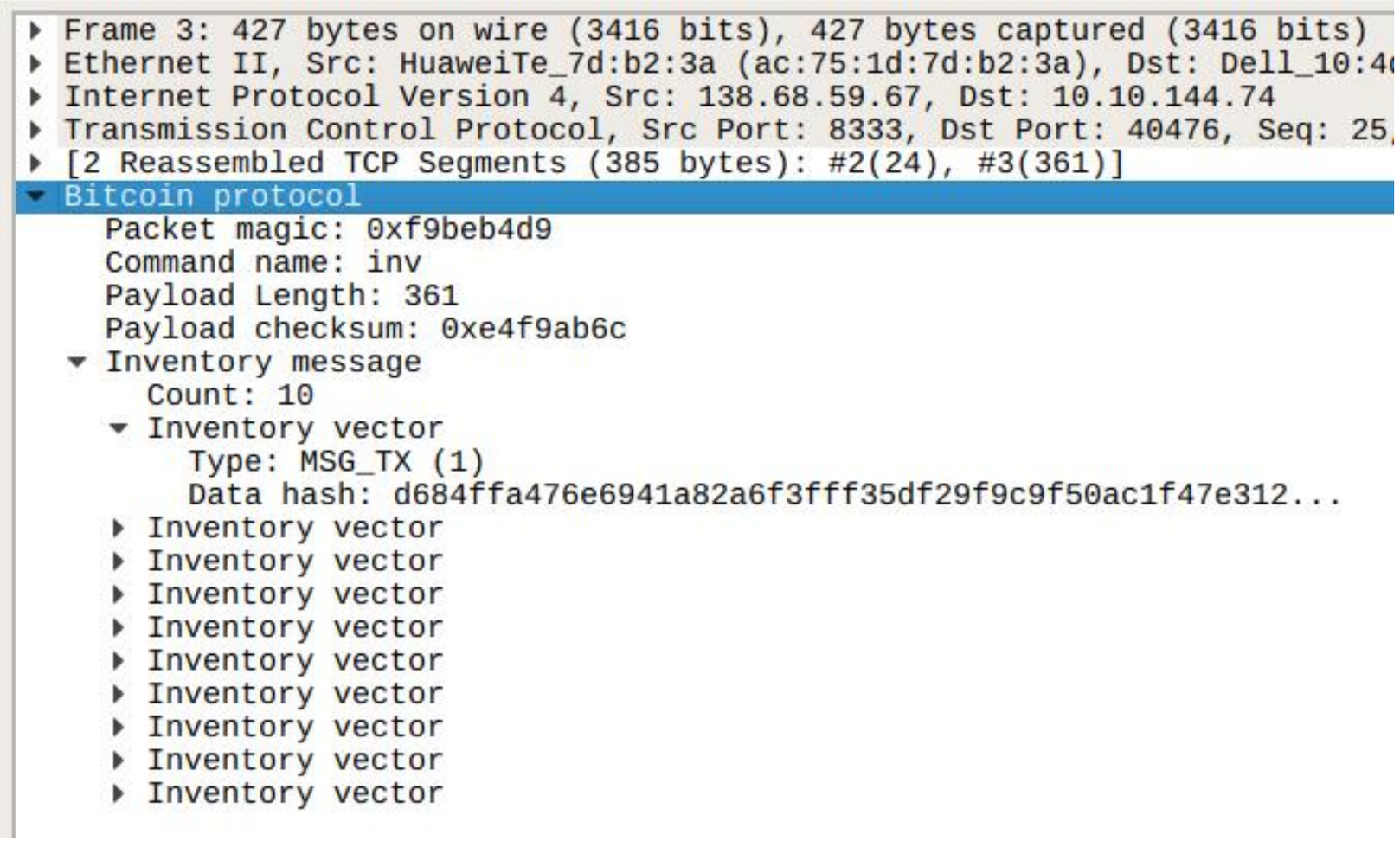
补充一下，程序刚启动的时候，就会开始用 tcpdump 抓包

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	138.197.153.151	10.10.144.74	TCP	463	8333 → 54479
2	0.001050	138.68.59.67	10.10.144.74	TCP	90	8333 → 40476
3	0.001097	138.68.59.67	10.10.144.74	Bitcoin	427	inv
4	0.001586	10.10.144.72	10.10.144.74	TCP	79	1030 → 49154
5	0.001596	80.79.114.34	10.10.144.74	TCP	90	8333 → 37124
6	0.001801	10.10.144.72	10.10.144.74	TCP	75	1030 → 41346
7	0.001813	80.79.114.34	10.10.144.74	Bitcoin	139	inv
8	0.001832	91.122.46.120	10.10.144.74	TCP	175	8333 → 49860
9	0.001868	34.221.229.176	10.10.144.74	Bitcoin	415	inv
10	0.002366	188.165.193.113	10.10.144.74	TCP	90	8333 → 50873
11	0.002379	188.165.193.113	10.10.144.74	Bitcoin	103	inv
12	0.002878	47.52.56.169	10.10.144.74	TCP	90	8333 → 59459
13	0.002926	47.52.56.169	10.10.144.74	Bitcoin	319	inv
14	0.002934	193.58.196.212	10.10.144.74	TCP	90	8333 → 33455
15	0.002980	193.58.196.212	10.10.144.74	Bitcoin	355	inv
16	0.003260	10.10.144.72	10.10.144.74	TCP	70	1030 → 44855
17	0.003647	91.106.188.229	10.10.144.74	TCP	103	8333 → 59872
18	0.003655	76.84.79.211	10.10.144.74	TCP	66	8333 → 58726
19	0.003661	108.30.14.21	10.10.144.74	TCP	90	8333 → 42395
20	0.004002	108.30.14.21	10.10.144.74	Bitcoin	931	inv
21	0.004126	10.10.144.72	10.10.144.74	TCP	74	6102 → 18530
22	0.004136	173.249.2.209	10.10.144.74	TCP	90	8333 → 48801
23	0.004143	10.10.144.72	10.10.144.74	TCP	74	6104 → 18530
24	0.004183	173.249.2.209	10.10.144.74	Bitcoin	211	inv
25	0.004779	10.10.144.72	10.10.144.74	TCP	74	6106 → 18530
26	0.004796	108.2.174.127	10.10.144.74	TCP	90	8333 → 43309
27	0.004827	204.12.207.91	10.10.144.74	TCP	90	8333 → 54586
28	0.004842	204.12.207.91	10.10.144.74	Bitcoin	74	pong
29	0.004844	82.244.37.15	10.10.144.74	TCP	60	8333 → 46464
30	0.004868	54.72.249.230	10.10.144.74	TCP	90	8333 → 36858
31	0.004949	54.72.249.230	10.10.144.74	Bitcoin	895	inv
32	0.005319	51.89.173.199	10.10.144.74	TCP	90	8333 → 46829

```
sudo tcpdump -i [INTERFACE]
  \ -w %s.[INTERFACE].pcap
  \ -v -n -G 2 -B 65536
  \ -Z [USERNAME] 'tcp'
  \ > [INTERFACE] 2>&1 &

 sudo tcpdump -i lo
  \ -w %s.lo.pcap
  \ -v -n -G 2 -B 65536
  \ -Z [USERNAME] 'tcp'
  \ > lo 2>&1 &

参数含义：

-i     哪个网口
-w     保存成cap文件
-v     产生详细的解码信息
-n     不对地址进行数字表示到名字表示的转换
-G     按照秒数的格式命名-w的文件
-B     buffer size
-Z     sudo启动后，监听指定用户的包

lo是本地回环接口
```

pcap

数据包的具体内容：



Value	Name
0	ERROR
1	MSG_TX
2	MSG_BLOCK
3	MSG_FILTERED_BLOCK
4	MSG_CMPCT_BLOCK

用 getdata 命令可以根据hash拿到具体的内容

其他

其他问题：

1. 用什么方式拿具体的交易内容合理
   用解析全节点日志的方式
   直接发 getdata 请求的方式
   rpc接口

2. 用 getdata 请求的方式，是不是能做关于比特币数据的 to C 的界面？

3. 如何判断一笔交易是第一次出现？

4. 多节点怎么同步数据、日志怎么记录、系统架构怎么设计？

海联金汇 HYUNION HOLDINGS

股票代码：002537

UMF 联动优势 UNION MOBILE FINANCIAL TECHNOLOGY

海联金汇旗下企业

科技赋能普惠金融

谢谢！

Thank you !

MAKE FINTECH EVERYWHERE