# Recreate bitcount：交易的传播

## 声明

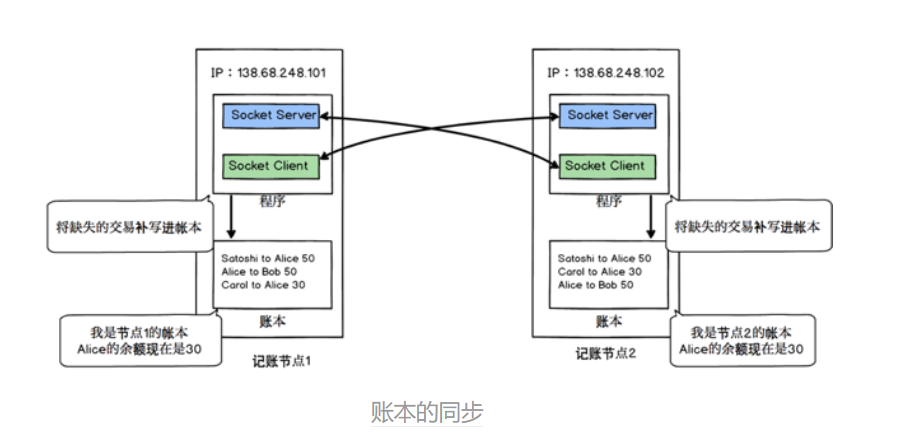
由于只有一台电脑，虚拟机的ip设置静态有些麻烦，无法按照作业要求在同一网段进行交易的传播（主要是bamboo网络坏掉了，无法把主机ip同步到虚拟机的静态ip）

## 需求文档

在代码readme.md中

*# 文件分析*\* UTXO.txt :记录UTXO，以硬件形式记录，每隔5分钟记录更新一次  
  
*# 交易的传播*\* 对于每个节点，使用多个固定的IP作为活跃节点，发送区块到这些节点作为交易的传播  
\* 受到区块，进行节点的更新（不进行验证）  
\* 受到交易（单个区块），进行独立的验证（UTXO 和 交易缓冲池 和 数据有效性验证）  
  
  
*## 每个节点存储的信息*BTC网络并不是收到一条广播就立刻更新系统的状态，而是有区块以及内存池的设计。  
  
网络中的（全功能）节点各自维护一个状态的副本，并在某一时刻就系统的状态达成最终一致。  
  
在某一个时刻，所有BTC节点都维护着一个记录UTXO的账本，并有一个接收未确认交易的内存池（Mempool）  
  
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  
  
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  
  
\* UTXO（存在硬盘中，文件） UTXO还是交易16进制字符串  
\* Menmpool（存在内存中，每隔一段时间进行验证-》清空-》更新UTXO）  
\* 区块  
  
收到boardcost->立刻更新到Menpool->验证boardcost的交易有效性->更新自己的UTXO  
  
*# 基本流程*\* 1,交易产生(交易：16进制的字符)  
\* 2,交易的广播(加入Menmpool->经过一段时间->写入UTXO)  
  
  
*# server 端 和 client 端 作用分析*\* 对于每个节点，都有一个server和一个client部分，server接收其他节点的交易,client发送本节点的交易  
  
*## server port 分析*\* 接收所有区块，缺失的区块进行有效性分析，写入chain  
\* 接受的ip地址只有固定的多个Ip  
\* 还是定时更新，不能实时更新  
\* 重复性判定：是否是本机的已有区块，是接收之后的第一个判定  
  
  
*# UTXO*\* 所有的机器都有某笔旧交易存在UTXO中  
\* UXTO 是每个特定交易 的每个输出 里面是 公钥和金额 直接连接，没有其他东西  
\* UTXO 是不可分离的  
\* 每有一笔交易进入，减去输入，加上输出  
  
*# 本机更新完毕之后*\* 发送新的区块到特定的活跃节点（发过的区块不再发送）  
\* 收到别的交易，判定缺失，缺失的交易写入tx\_block.txt  
  
  
*## UTXO anb tx\_block chain 区别分析*\* 更新 \_\_block chain\_\_ 就是更新 \_\_UTXO\_\_

## 安全模型



每个机器有一个socket server 端口和一个socket server 端口，按照上图进行更新区块

## 靶机IP

设置活跃地址，固定死IP地址在代码中.

保证两台机器网络畅通。

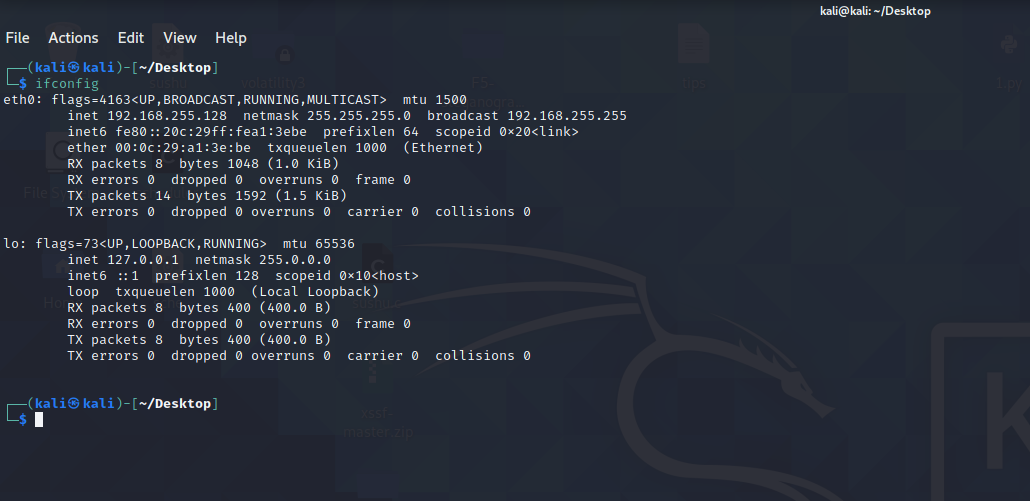


图 1虚拟机IP

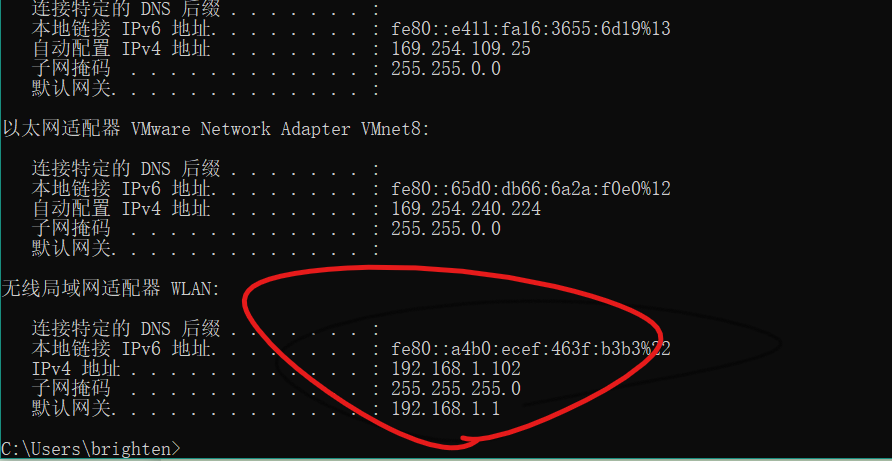
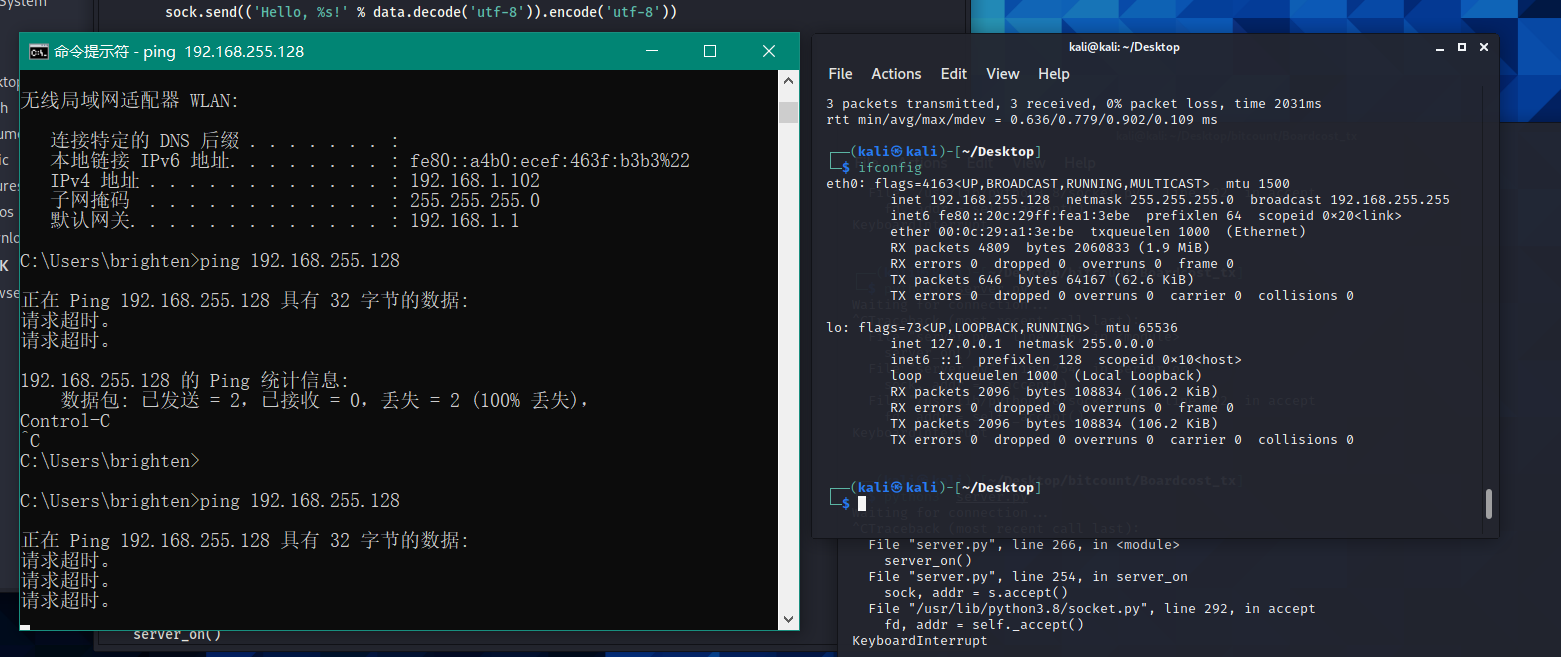


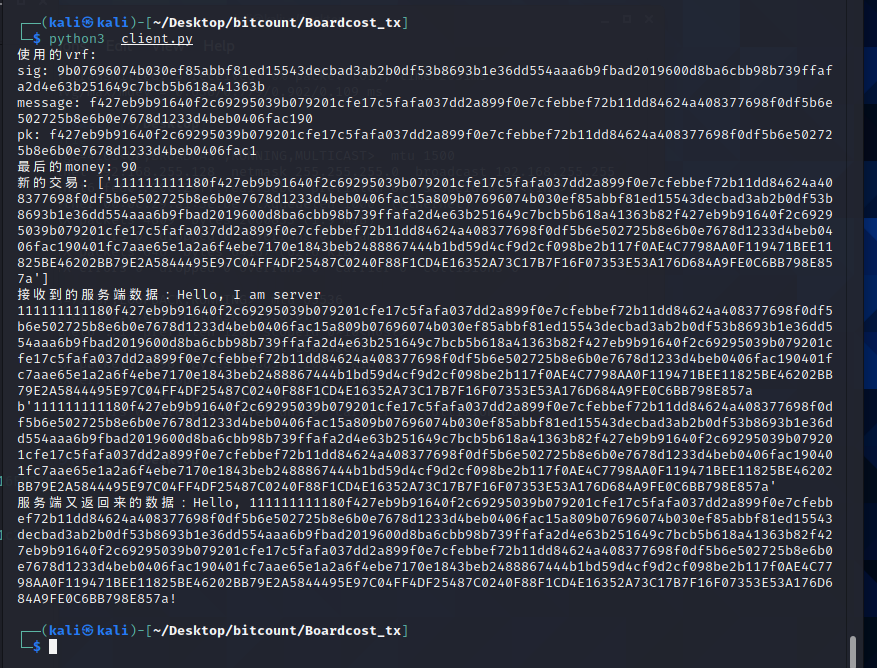
图 2本机IP



虚拟机可以Ping 主机，但是主机不能ping 虚拟机。

那么交易就只能从虚拟机同步到主机了。

虚拟机传播交易到主机



主机的server端口



由于已经在UTXO和本机的节点中存在，所以交易无法上链



图 3UTXO

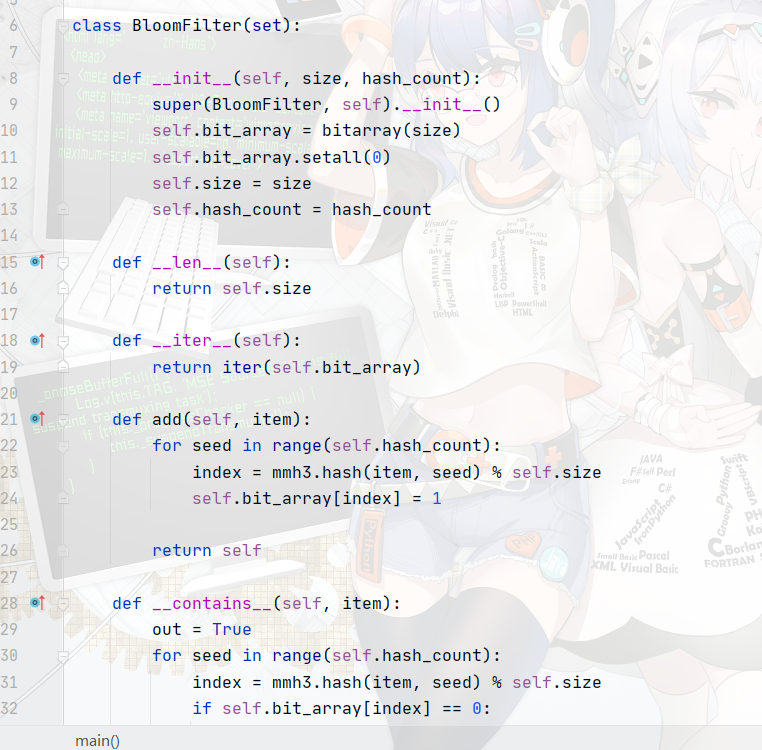


图 4block chain

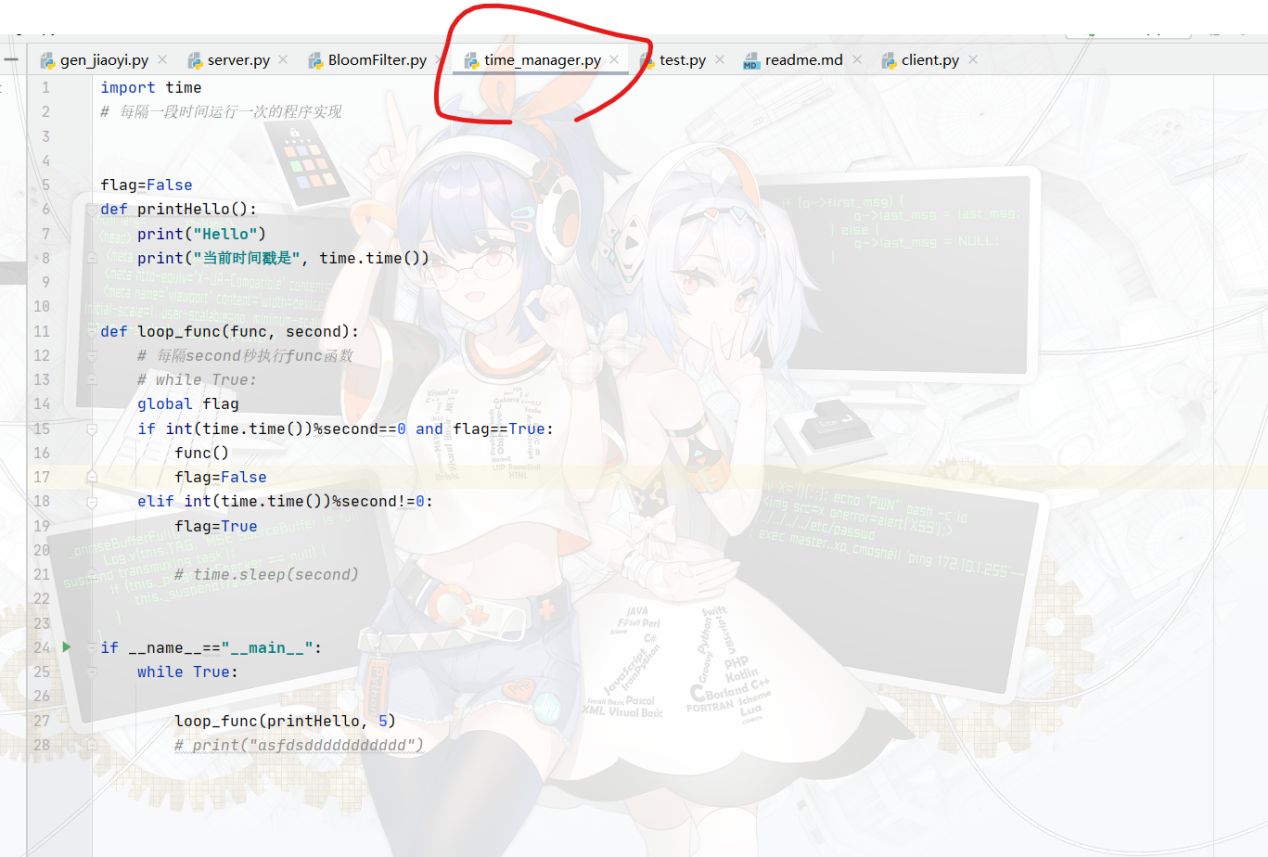
## 花里胡哨的功能

### Bloom filter类

用于检测是否已经出现



### Time manage类



用于每隔一段时间运行清空内存池，更新UTXO 池

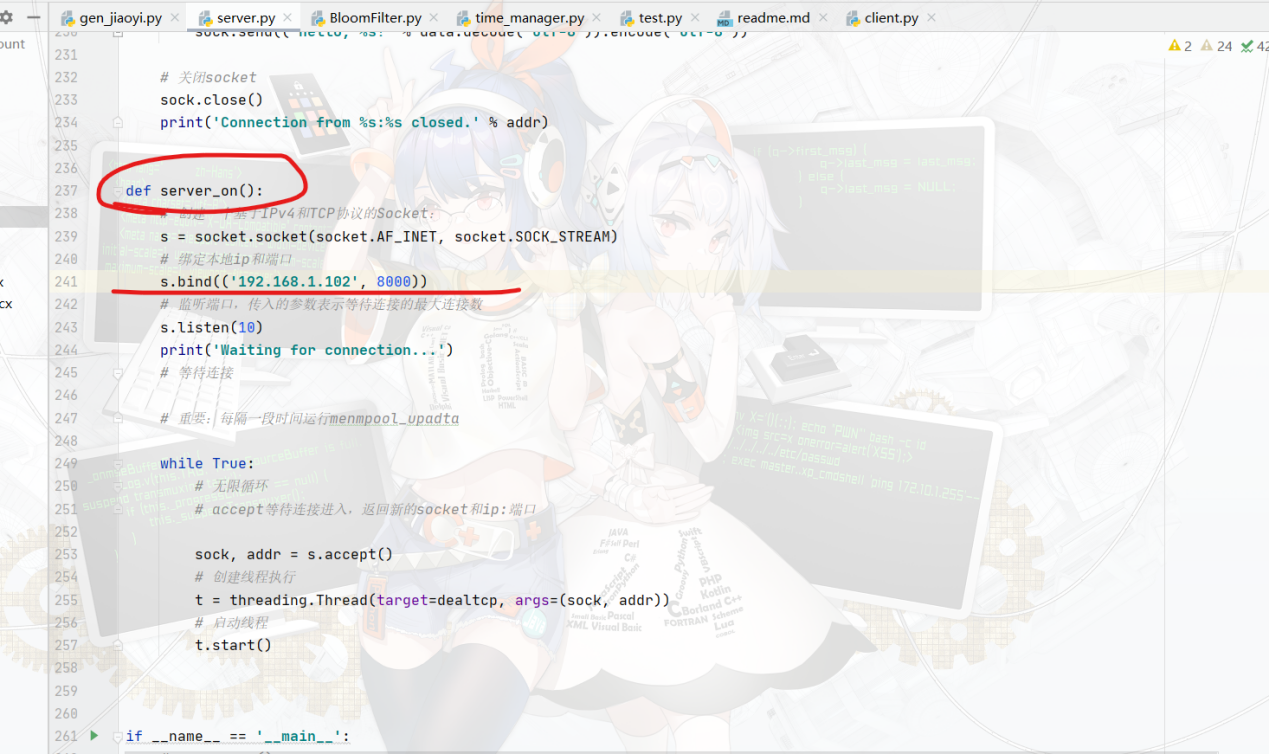
在运行时候，单独运行

### Client 端口

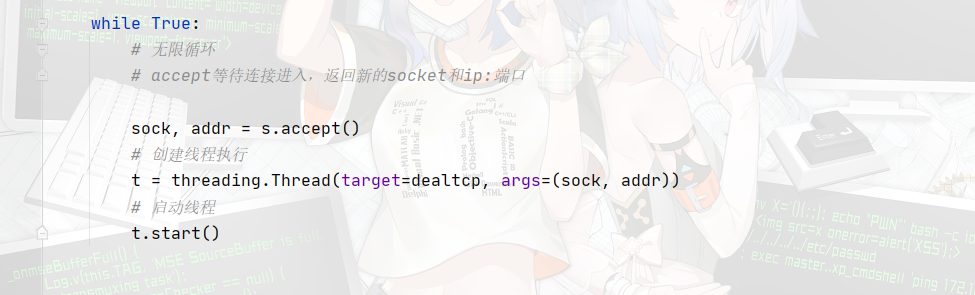


活跃主机号，写死为本机和虚拟机的ip地址

### Server 端口



设置本机的ip和端口号



写一个死循环，server端口持续打开，接收client的消息

不写了，花里胡哨的