P2P网络底层框架实现

概要：

- 本项目使用python开发了一个完整的p2p网络框架，为上层的app设计提供了底层的网络支持，实现了节点发现，连接，节点管理等一系列p2p网络功能。程序主要使用python底层的套接字和多线程编程，涉及的库均为python提供的标准库。

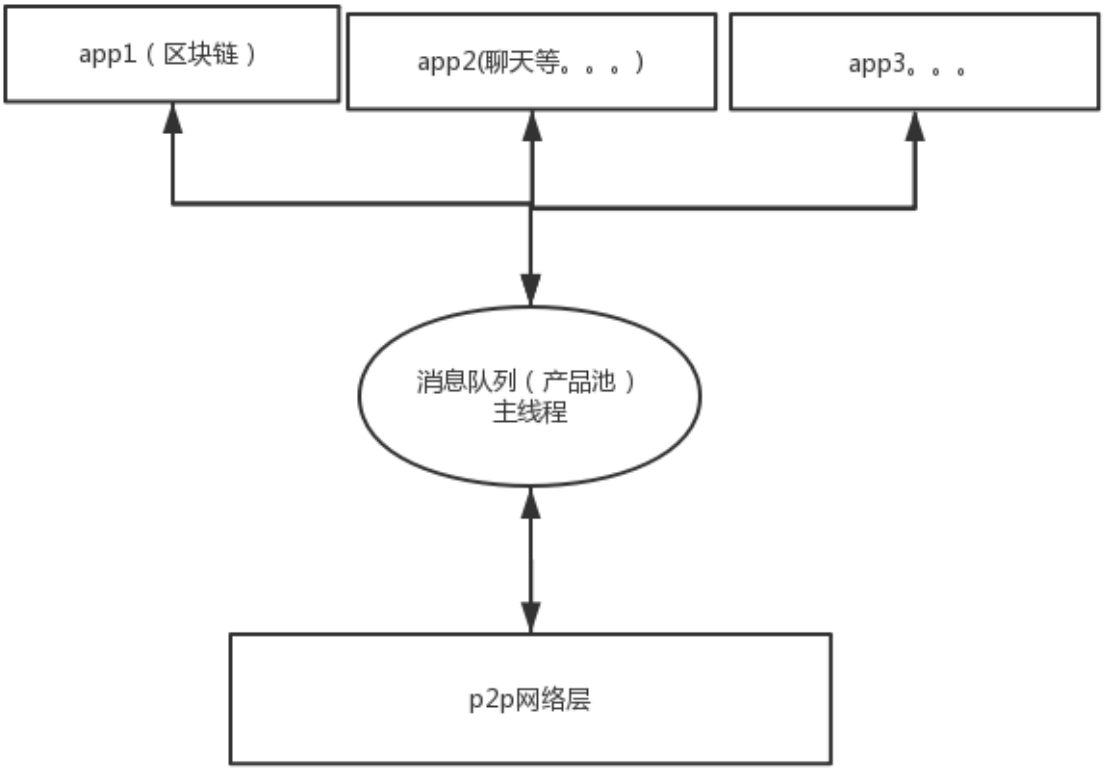
背景：

- 随着区块链技术的发展，去中心化技术的优势逐渐凸显出来。去中心化带来了对过去问题全新的解决方案（信任问题等等），并且人们也正在不断开拓新的去中心化应用，本程序旨在利用python实现去中心化的底层架构，即p2p网络框架的设计和实现。
- p2p网络的目标：
  - 所有节点运行同一套协议（甚至是完全相同的代码），能够互相发现和连接。
  - 每个节点 作为一个服务端和多个客户端。
  - 能够同态管理自己连接到的，以及连接自己的节点（处理连接断开）。
  - 与上层app进行对接。
- Why python：
  - python虽然执行效率低，但是开发效率较高。
  - 代码简洁，可读性强。
  - 因为本项目的结果是一个底层的p2p网络，所以需要较好的扩展性以供上层的app（比如区块链应用，点对点聊天等）运行，本程序已经预留了上层app的设计接口。

程序结构：

整体结构：

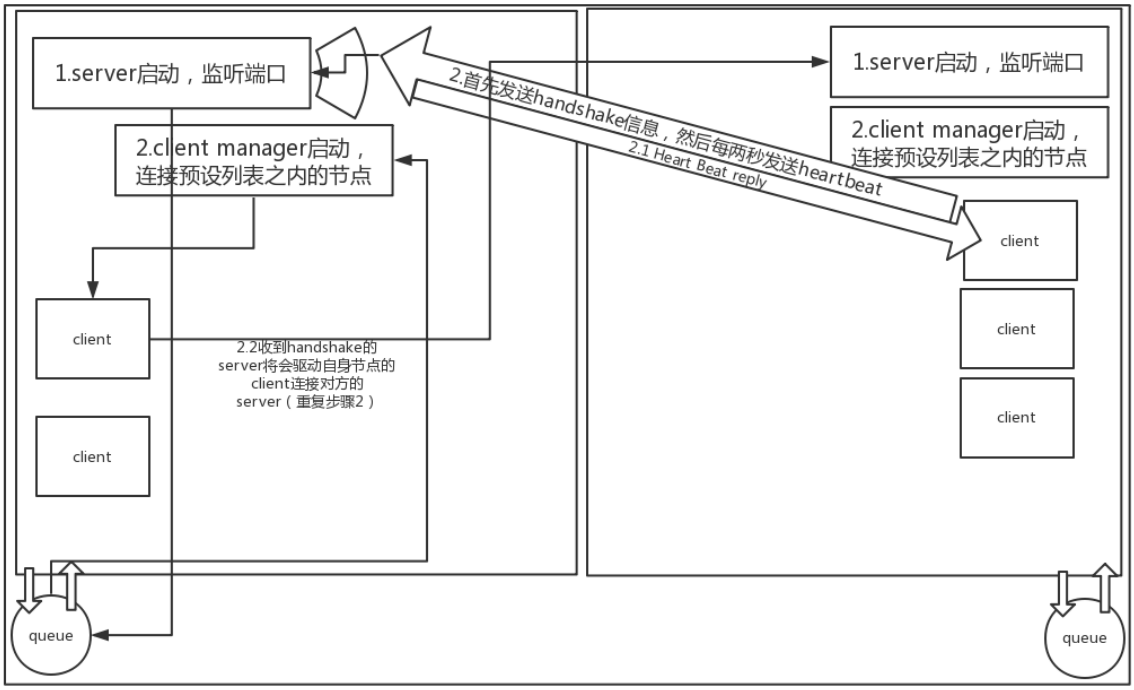
-



- 由于网络层为目前程序中唯一的层级，因此网络层的主要代码运行在主线程中（不影响后续app的运行，并且带来的好处是程序能先准备好网络层以供app调用）。
- 程序总体采用了 生产-消费者 的设计模式，使用在python多线程中应用较广的queue对象作为产品池子，程序主线程末端是一个无限循环，将不断从queue对象中消费产品（queue中没有对象时主线程阻塞）并调用相应的handler进行处理。
- queue中的产品主要是msg，目前的消息内容主要如下图所示（其中提到的client和server均为同一节点的不同part，因为在p2p网络中，每一个节点既是服务端（唯一）又是多个客户端（多个））。
  -

app	type	operation	content	说明
Network	heartbeat	/	/	clients发送的心跳
Network	heartbeat reply	/	/	server回复的心跳
Network	new_peer	to	sock 对象	client连接到新节点
Network	new_peer	from	sock 对象	server收到新连接
Network	lost_peer	to	sock 对象	client断开连接
Network	lost_peer	from	sock 对象	server断开连接
Network	handshake_peer	from	"<serverip>:<server port>"	本节点client向对方server建立连接时将会把自身的server地址发给对方节点，对方节点收到后将会驱动对方的client连接到本地的server

网络层结构：



p2p network（节点间）

- **网络层的运行过程：**
  - 1.首先扫描本机端口，寻找合适的端口作为服务端口。
  - 2.客户端管理器 开启多线程 连接到预设的其他节点的服务器端口（预设列表中去除上一步用于自身服务器的端口）
  - 服务端接受到连接后开启多线程处理客户端请求，同时驱动自身的客户端管理器向对方节点建立连接（也就是建立双向连接，过程如图中细的黑色箭头）
  - 收到连接将把对方客户端口(socket对象)加入clients_from字典，发送的连接将把对方服务端口(socket对象)加入clients_to字典。连接断开时将会对应删除。
- **server端逻辑：**
  - 遍历预设端口列表，寻找合适的本机端口开启监听，将找到的端口返回备用
  - accept阻塞服务端进程，当客户端连接时，调用新线程监听客户端消息，同时向全局的queue对象中传递`new_peer from`消息（将由客户端管理器（同时也是网络层消息的handler）进行处理）。
  - 监听线程设置5秒超时，（客户端心跳频率为两秒一次），超时将断开连接，向全局queue中传`lost_peer from`。
  - 接收的消息为心跳（`heartbeat`）时，将立刻回复客户端消息`heartbeat_reply`，方便客户端所在的节点维护clients字典。
  - 接受的消息为`handshake`时(通常在server接受到新连接时，即与第二条同时收到)，将消息传递给全局queue，并利用下述的client manager驱动client连接到对方的server。
- **客户端管理器(also the handler of network-related msgs)逻辑**：
  - server端启动结束后创建管理器对象，驱动多线程连接预设列表中的节点的server。
  - **client 端逻辑：**
    - 受到client manager的控制，向对方节点的server创建连接。

        - 创建连接时开启心跳发送线程和消息监听线程，同时向对方server发送`handshake`消息，该消息中包含了本节点的server地址，方便server所在的对方节点进行双向连接。
            - 前者发送两秒一次的心跳 消息`heart_beat`。
            - 后者接受对方节点的server传来的所有消息，包括`heart_beat_reply`
    - 关于msgs的处理（**handler of network-related msgs的逻辑**）：
        - `lost_peer from`和`lost_peer to`将会分别从全局变量`clients_from`和`clients_to`中删除对应的节点。
        - `new_peer from`和`new_peer to`将会分别从全局变量`clients_from`和`clients_to`中添加对应的节点。
        - 关于`handshake`的处理：如果对方的服务器端口不在`clients_to`的值当中，将命令client连接对方的服务器