分布式系统大作业 – 去中心化的聊天系统

21307289 刘森元

0. 项目环境

Python 3.11.6

1. 项目背景

传统的聊天系统如微信等是一种中心化系统设计，数据集中存放.

本项目目的是设计一种去中心化的聊天系统，将聊天数据分散存储在各个客户端上.

2. 理论基础

2.1. Peer-to-Peer

对等式网络 (Peer-to-Peer)，是去中心化，依靠用户群交换信息的互联网体系.

本项目通过使用 P2P 技术实现用户间的信息交换.

2.1.1. NAT 穿透

考虑到实际的用户场景，目前大多数机器分布在大大小小的 NAT 内部网络中，若要实现 P2P 的信息交流，就需要一台 Announce 来进行信息的传递，比较知名的架构有 BitTorrent.

本项目简易的通过一台 Announce 来进行信息传递，注意，Announce 在这里仅传递 Peer 之间的 NAT Socket 信息，而不参与实际的聊天内容传递.

2.2. 聊天决策

基于 P2P 的背景，项目中采取 Announce 独裁的方式进行聊天管理，通过决策转发 Socket 来实现聊天室功能，即

- Peer 向 Announce 请求聊天室内其他 Peer 的 Socket
- Announce 判断聊天室内当前存在哪些 Peer，并响应请求
- Peer 收到响应后进行信息的转发

由于没有收到非聊天室内其他 Peer 的 Socket，并且基于 NAT 网络环境下 Socket 经常变动的特点，非聊天室内的 Peer 相对于聊天室内 Peer 是透明的.

一对一聊天则为聊天室的特殊情况.

2.3. 系统一致性 & 失效容错措施

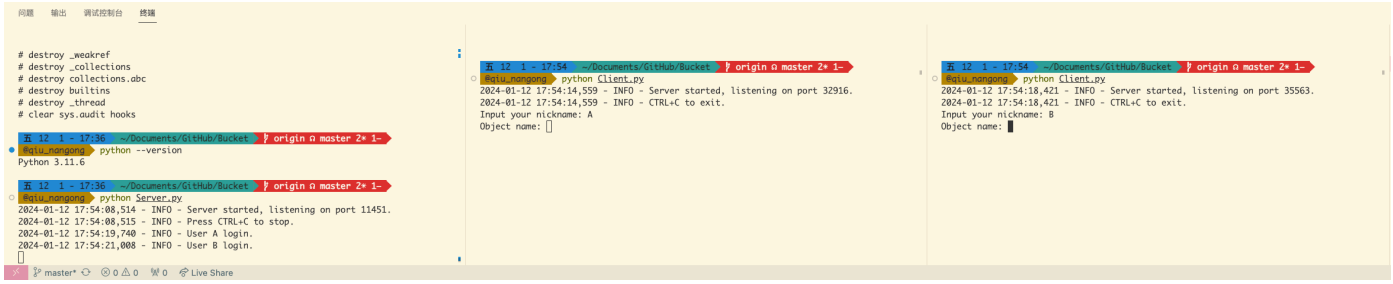
项目中有一个关键决策是

- 当信息发送目标不在 Announce 中时，禁止该信息的传递.
- 每个 Peer 定时向 Announce 发送 Heartbeat，超时认为该 Peer 下线
- 使用 gRPC 来实现信息传递，其中 gRPC 已经保证信息传递的正确性，避免了拜占庭类的实效

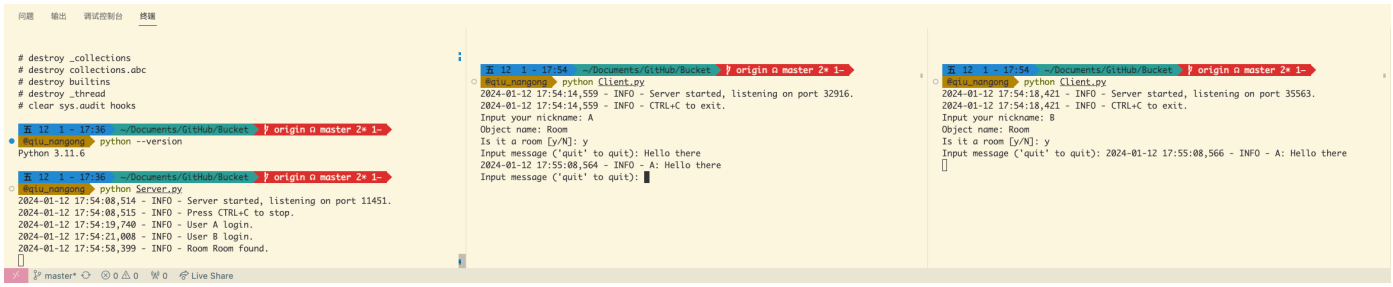
3. 项目展示

3.1. 基本功能

Peer 上线，Announce 进行广播



Peer 进入 Chatroom，并发送信息



Peer Ctrl+C 强制退出，Server 检测 Peer 下线

其中通过 timestamp 可见响应时间不会超过 10ms

3.2. 性能测试

由于性能测试 Peer 过多，不在此贴图展示.

在一台 Linux 服务器上使用脚本并行运行若干 Peer 进行并发测试，有结果如下

Peer 数量	平均响应时间	响应比例
4	3.382 ms	100%
16	4.284 ms	100%
128	7.599 ms	100%

可见随着 Peer 数量上升，响应时间随之上升.

分析可知，Announce 进行查表转发 Socket 的过程处理时间导致的.

4. 项目参考

Peer-to-peer - Wikipedia

NAT traversal - Wikipedia

gRPC Docs