计算机网络第一次实验

2015013249 孙子平 软件学院

1 回答问题

1.1 UDP问题

1.1.1 如何写一个借助UDP的聊天软件使两个客户端互相聊天？

服务端在接收到其中某一个client消息的时候，向另一个用户转发这条消息，如果要绕过路由器，那可能要加个心跳包。（由于助教高亮了**clients**，就假定不能p2p，如果可以p2p，那没啥难度）

1.1.2 能否使用UDP传输文件？如果可以该如何做？

可以使用已有的TCP on UDP的方案，比如说使用OpenVPN，搞一个UDP隧道。OpenVPN会负责好隧道上的TCP可靠传输。（我才不会告诉你可以用OpenVPN 53号UDP端口搞代理实现校内免流量）

此外也可以自己实现一个协议，比如给一套接收方未收到就提示发送发重试，然后再分块checksum。不断反复知道checksum都过了。

1.2 TCP传送问题

1.2.1 如何传送大文件而不阻塞服务器？

文件传输的socket交给非阻塞或异步API即可。在Linux上就是使用select/poll/epoll。

弱弱地吐槽一句：我是真的实现了超高性能的大文件非阻塞式传输，手撸块状链表缓冲区+内存池优化性能（详见下文），写得欲仙欲死，助教一定要明察秋毫啊。很多人control用了epoll，但data是同步的；或者data是异步，但是是繁忙等待地轮询data是否传输完毕。

2 作业特性简介

2.1 UDP作业

使用Node.JS的异步API，可以避免Python版本的阻塞式丢包问题。并显示出各个包的响应时间和Timeout的包。

2.2 FTP服务端

服务端特性如下：

- 拥有一个CLI（command line interface），支持自动补全，可以用于踢掉客户端等等
- 可同时开多个服务端（每个服务端拥有相同的账号系统），如果需要多个不同的账号系统需要多个示例
- 支持开双栈（IPv4和IPv6）的服务端（支持EPSV和EPRT命令）
- 基于bcrypt的用户验证系统，每个用户有独立的被囚禁的根目录
- 通过输错密码等待3秒防止暴力尝试密码
- 对于signal适当的处理，干净地清理用户链接
- 超高性能的非阻塞式编程
- 极其鲁棒性的防御性编程，就算malloc失败了，程序还能正常跑（我甚至是把/dev/urandom胡乱地cat到自己服务端上测试的）
- 良好的日志输出（可以`--quiet`，`--verbose`和`--debug`）
- 额外实现的命令包括：SIZE, APPE, DELE, EPSV, EPRT

CLI依赖了GNU的readline库，可以自动补全，和使用历史命令。命令行参数添加`--cli`即可使用，也因此支持了配置文件，使用方法是`cat example.conf | ./server --cli --no-prompt`。命令包括开启及关闭服 端、添加及删除用户、踢掉客户端等等。使用`help`可以查看帮助

为了支持海量用户，用户验证部分使用了以名字为键的哈希表，密码使用bcrypt加密，避免了服务端存储明文密码的不安全性。输错密码的延时是通过非阻塞timerfd做到的（PS虽然实际上是非阻塞，但在客户端看来服务器想被阻塞了，即不处理完当前命令服务器，不会处理新的命令）。

signal使用了signalfd做到干净地处理（第一次收到Ctrl-C关闭所有服务端，但保留客户端的连接，等待客户全部退出再退出，第二次收到Ctrl-C直接全部关闭）。

每个系统调用和malloc之类的调用我都会检查返回值。错误处理代码甚至是这次2k+服务端代码的主要组成部分。

文件传输使用了带内存池的块状链表，每块为4K内存，一次用户传文件最多分配16K这样的块，总计64M的传输缓存（这样在发送文件的时候，一旦写就绪我就尽可能写，如果写不就绪，就把文件读入缓冲区，使下一次写更快），在大文件传输上性能碾压。这里我解释一下为什么不适用zero-copy的`splice`之类的API，最主要我原计划支持SSL/TLS，然而目前SSL/TLS无法直接对splice透明。

2.3 FTP客户端

客户端特性如下：

- 使用全新的Python3.5异步协程API，还玩生成器，就是不想让别人看懂代码
- 良好的signal处理，可以终止当前操作（如下载文件）或直接退出客户端（该功能是在主教的奇葩PASV模式之后加入的，我不想实现非标准，但我可以不让我的客户端被挂死）
- 仿GNU/Linux上ftp命令的交互方式
- 拥有上传下载的进度显示
- 可以支持上传的断点续传（使用SIZE + APPE命令）

- 额外实现的命令包括：HELP, SIZE, DELE和APPE

按Ctrl-C一次可以终止当前的操作。1秒内按Ctrl-C两次会直接退出客户端。

3 作业遇到的困难

本次作业堪称异步/非阻塞式IO大礼包，分别使用了3种语言的异步/非阻塞式编程。现在就叙述一下自己的感慨。

首先就异步而言，js是天生的，拥有较为良好的生态环境。而py虽然起步晚，但API自由度比js自由度更高。比如原生js的Promise是不支持Cancel的，但py的future就可以Cancel。但py生态环境差，先前实现的一些库也未能很好的与asyncio兼容。比如说py原生的图形界面库tkinter，其消息循环不能和asyncio的消息循环兼容（要么你一个启动另一个导致某个被挂住，要么你开双线程，都很不优雅），这也是我没有写图形界面的主要原因。（另一个原因是，文件夹浏览器都可以直接打开我服务端）

然后再说说C吧。只能用痛苦来形容C的非阻塞编程。就拿传文件来说，有读就不就绪、写就不就绪和缓冲区尚有空闲等多种情况。每种情况都需要细致的处理。此外如何使这个异步的服务器在客户端看来像是阻塞的，不完成当前命令不能继续处理，验证失败等待3秒，这都是很麻烦的。只能说这种用消息循环搞非阻塞式IO的程序，起逻辑根本不是顺序的，这造成了代码维护的困难。

所以才感慨`libuv`之类的库的牛逼，也更多地体会到了协程的好处。

最后吐槽一下助教给的代码好多bug，很多人被坑到了。（虽然我看都没看）