UDP & FTP Report

Program 1 in lecture Computer Network

School of Software Jingao Xu 2013013310

Section1：实验环境 Environment

- Hardware：

CPU：Intel(R) Core(TM) i7-3720QM CPU @ 2.30GHz

Memory: 16GB

- Software：

Operation System: MAC OS

Programming language：Python(UDP) C(TCP/FTP)

Section2：UDP实现

- UDP概述：

Socket 客户端主要步骤如下：

创建一个socket —> 与服务器连接 —> 发送package包 —> 输入文字 —> 等待服务器应答 —> 接受关闭命令Socket关闭

Socket 服务器端主要步骤如下：

创建一个 Socket —> 接收连接的请求 —> 接受处理命令 —> ctrl+c关闭Socket

- UDP问题回答：

如何利用UDP实现两个客户端的聊天：可以考虑利用上层协议来保证可靠传输：如果客户端使用UDP协议发出消息后，服务器收到该包，需要使用UDP协议发回一个应答包。这样可以保证消息的传输

是否可以利用UDP进行文件传输？如果可以，如何实现：UDP是一个不可靠的传输连续数据的协议。它主要存在两个问题：接收包的次序随机和丢包。为解决收包次序问题可以在每次传输时定义一个序列号，这可以让接收端按照正确的顺序储存这些包而不必管他们到达的先后顺序。为解决丢包问题可以让接收端每次正确地接收到一个包数据之后向发送端发出接收请求，这个办法可以消除丢包的可能性。但这两种办法都会使传输变得非常缓慢，也是利用UDP付出的代价。

Section3：FTP实现

- **FTP概述：**

Socket 客户端主要步骤如下：

创建一个 Socket —> 与服务器连接 —> 命令输入和应答 —> QUIT命令关闭 Socket

Socket 服务器端主要步骤如下：

创建一个 Socket —> 监听 —> 接收连接的请求 —> 接受处理命令 —> ctrl+c关闭Socket

注：本*FTP*利用*fork()*函数实现了支持多个客户端连接服务器

- FTP程序运行方式：

服务器：在server目录下“make”，之后在终端输入“./server -port(可选) -root(可选)”。如果不输入“-port”“-root”默认为“/tmp”目录和21端口。

客户端：在client目录下“make”，之后在终端输入“./client host port”，如：“./client 59.66.139.82 9000”

- FTP端口：

FTP 使用 2 个端口，一个数据端口和一个命令端口。

在此FTP中，这两个端口一般是21 （命令端口）和 20 （PORT数据端口）。在本实验中在运行时输入“-p 端口号”来确定命令端口（如果不输入，默认为21端口），控制 Socket 用来传送命令，数据 Socket 是用于传送数据。每一个 FTP 命令发送之后，FTP 服务器都会返回一个字符串，其中包括一个响应代码和一些说明信息。

- FTP 命令：

FTP 每个命令都有 3 到 4 个字母组成，命令后面跟参数，用空格分开。

要下载或上传一个文件，首先要登入 FTP 服务器，然后发送命令，最后退出。这个过程中，主要用到的命令有 USER、PASS、REST、STOR、RETR、PASV、PORT、QUIT、ABOR以及CWD，CDUP，DELE，LIST，MKD，PWD，RMD，RNFR，RNTO。这次试验过程之中我已经实现了全部这些命令，其中，前半部分命令在文档之中已经有过介绍，下面对自己实现的后半部分命令进行简单地介绍：

CWD：改变工作目录,进入当前目录的下一层文件夹或者其他目录下得文件夹，若不存在文件夹，则报错。如：“CWD dirname”。目录的输入可以带/也可以不带，程序会进行自动处理。

CDUP：进入当前目录上一级目录。没有参数，如：“CDUP”。

PWD：返回当前目录。如“PWD”。

LIST：在已经设置PORT或PASV模式之后，可以返回当前目录的文件和文件夹（包含隐藏文件），否则提示先设定模式。没有参数，如“LIST”。

MKD：在当前工作区创建一个文件夹，但不改变目前的工作区，如果文件夹已经存在，会报错。含有参数，如：“MKD dirname”

DELE：在当前工作区删除一个文件，如果不存在该文件或者不够权限，会报错。命令含有参数，如：“DELE filename”

RMD：在当前工作区删除一个文件夹，如果不存在该文件夹或者权限不够，会报错。命令含有参数，如：“RMD dirname”

RNFR：重命名命令，需要和RNTO结合使用，选定一个要重命名的文件或者文件夹作为即将重命名的目标文件，命令含有参数，如：“RNFR dirname/filename”

RNTO：重命名命令，需要和RNFR结合使用，修改选定的文件夹或者文件为给定名称，命令含有参数，如：“RNTO dirname/filename”

Section4：总结与反思

- 程序特色：

在实验过程之中实现了大部分的ftp命令（扩展功能见Section3），并且通过科协的服务器测试了自己写的客户端实现的基本功能与扩展功能，用下载的transmit客户端软件检测了自己写的服务器实现的基本功能与扩展功能；在传输文件过程之中会发送文件的大小，并在退出的时候告知传输文件总大小；实现了注册用户登录功能，在.auth文件之中存储了一些用户名和密码，可以实名登陆。

- TCP与UDP

编写完两个练习，对TCP和UDP已经比较了解。

TCP提供的是面向连接、可靠的字节流服务。当客户和服务器彼此交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。TCP提供超时重发，丢弃重复数据，检验数据，流量控制等功能，保证数据能从一端传到另一端。
而UDP是一个简单的面向数据报的运输层协议。UDP不提供可靠性，它只是把应用程序传给IP层的数据报发送出去，但是并不能保证它们能到达目的地。由于UDP在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制（注：在UDP程序中加入超时等待判断）

- 反思

在本次实现中经历了一些波折。最初实现的FTP版本没有理解FTP的协议要求，认为自己的服务器能和自己的客户端正常通讯就好，很多错误代码以及判断信息返回（如230 message）的方式都不正确。这提醒我以后在完成任何开发过程中，都要先看清要求，避免之后对于代码改动带来的不必要工作。

体会到了协议的严谨性，最开始的时候发送消息都没有在后面加“\r\n”导致接收错误；没有用好“htonls”和“nohls”导致卡在socket连接不上，这让我知道了网络编程是一个严谨的过程，要遵守协议。

同时，在实现过程中，我没有想清楚考虑清结构，如：全局变量，函数调用等。直接导致后来一度思路混乱，为自己增添了很多麻烦，今后应当注意。