Project report

Final Project-FTP Proxy

Course Title: *Internet Applications*

Name: HU Jiaming *(2013212981)*

SHI Yuqi (2013212998)

Date: *June 15th, 2016*

Content

1. Overview

1.1 Goal of the project

- Design a FTP proxy program based on Linux command line terminal.
- The proxy can transfer files between client and server within either passive and active mode

1.2 Requirements of the project

- FTP proxy is able to set up separate control and data connections with FTP client and FTP server separately.
- FTP proxy is able to receive the commands from FTP client using control connection. The commands include: CWD, LIST/MLSD, MDIR, DELE, RNFR/RNTO, RETR, and STOR.
- Proxy can resolve the commands and modify if necessary (i.e. modify the parameters for PORT), and then forward the commands to the FTP server using control connection.
- FTP proxy is able to receive the FTP replies from FTP server using control connection. And it can resolve the replies and modify if necessary (i.e. modify the parameters in the replies for PASV), and then forward the replies to the FTP client.
- FTP proxy is able to set up data connections if required by either FTP client or FTP server.

- If in active mode, server would try to start data connection with proxy at first but in passive mode the client initiates this process.
- If FTP client wants to upload a file, FTP proxy can receive the file from FTP client and then upload the file to FTP server.
- If FTP client wants to download a file, FTP proxy can receive the file from FTP server and forward the file to FTP client. Meanwhile, the file will store in its cache.
- If the file that FTP client wants to download already exists in the cache, FTP proxy will not download the file from FTP server but send the file in the cache to FTP client. And no data connection will be set up between FTP proxy and FTP server. Otherwise, data connection will be set up between FTP proxy and FTP server.
- For data connection, implement both passive mode and active mode.
- Stable and friendly to users, and be able to handle error commands.

2. Requirements Analysis

2.1 Environment of the project

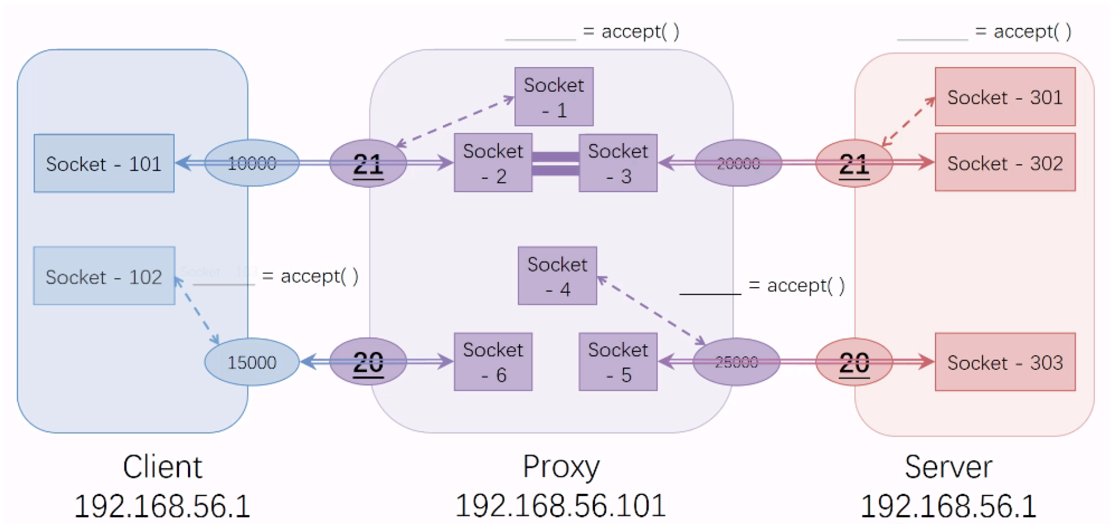
- C language
- Linux operation system (Ubuntu Server 12.04) act as proxy part
- Use gcc compiler and gdb debug tool
- Use CLI (Command Line Interface) as input & output
- FileZilla software as the client and server

2.2 Specific functional requirement

- Use Ubuntu act as client and connect it to a ftp server installed in my mac（192.168.56.1） and use Wireshark to analyze the FTP command and the reply packets of FTP protocol.
- FTP client use two connections to transfer either control information or data with server.
- Command link is should be established to pass the control packet information between client and server. Proxy should be able to pass the requirement from the client to server and modify some of them. (e.g. Calculate the port number when receive PORT message)
- Data connection should be established for transferring data and closed after one transmission.
- Following jobs should be accomplished:
    1. Client can upload the file to the server through proxy.
    2. Client can download the file from the server. The cache is introduced in this part, after one transmission, the file should be stored in the cache of the proxy. If client want to download again, no data connection is established between client and server, client can achieve the file from the cache.
    3. In the client, we can change the server’s directory, rename create and delete the file in the server.

3. Preliminary Design

Take the download process in the active mode as example:



3.1 Decomposition of functional modules

3.1.1 Establish Control Connection

- The proxy should bind and listen to the client
- After the proxy listen to the client requirement, it should create the control connection with client and server separately as long as the user name and password are matched.
- Synchronous I/O (including sockets) multiplexing for waiting user commands or server reply.

3.1.2 User Commands

- According to different server reply code, write the proxy reaction to different user's input
- Transfer commands by control connection and send to server.

3.1.3 Server Reply

- Convert server reply code to relevant type to hint user input.
- Data or command operation.

3.1.4 Establish Data Connection

- Download, upload and read or write files (jpg and doc).
- During the first download process, the file should be stored in the proxy cache.
- Close data connection after each transmission.

3.1.6 Proxy cache

- After the proxy receive the RETR command from the client, it will check if the file already exists in the proxy.
- If the file is in the proxy, proxy won't pass on the RETR command the server. Proxy transfers the file to client directly.

3.2 Relationship and Interface between the Modules

Control connection and data connection can be built upon TCP connection. User and server can interact in control connection and achieve data transfer in data connection with the proxy as an agent. Some modules are mainly used to convert requests and responses and others build connections. Besides, few modules are involved some unique functions to do detailed tasks such as calculating port. Interactions between user and server are completed by using some read and write buffers while connections are based on socket file descriptor and port number.

3.3 Design of Data Structure

3.3.1 Data connection mode:

passive mode and active mode

3.3.2 Data transfer mode:

Binary and ASCII

3.3.3 Structures:

struct sockaddr_in

struct in_addr

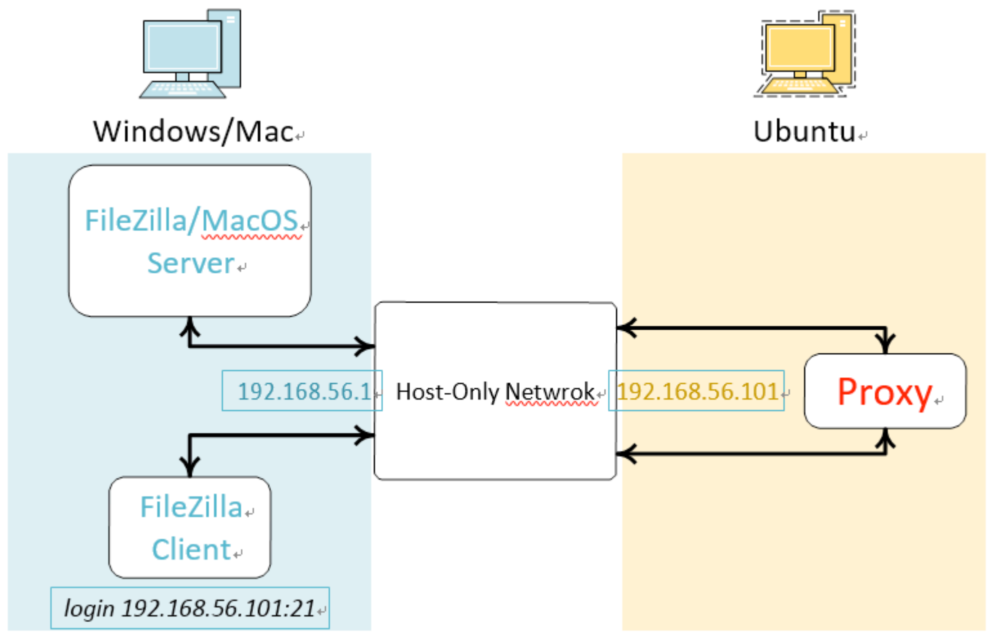
struct timeval

We create 6 sockets with the select() function to monitor if there is any changes in any socket.

4. Detailed Design

4.1 The structure of project

- We use our PC as the terminate of the FTP client and the FTP server, which IP is 192.168.56.1
- We use the Ubuntu as the FTP proxy, which IP is 192.168.56.101
- We use a Host-Only Network to connect client, server and the proxy.



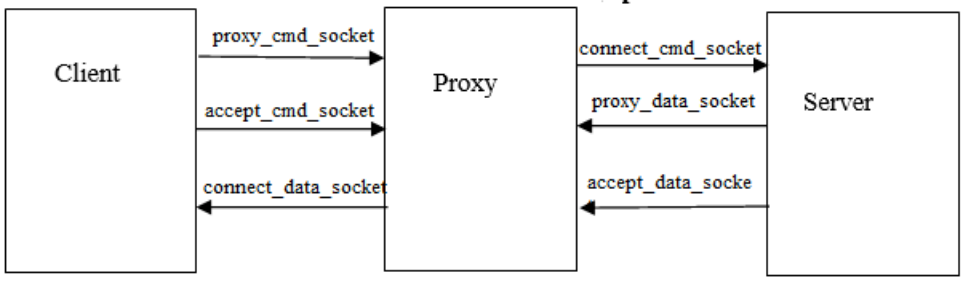
Diagrammatic sketch

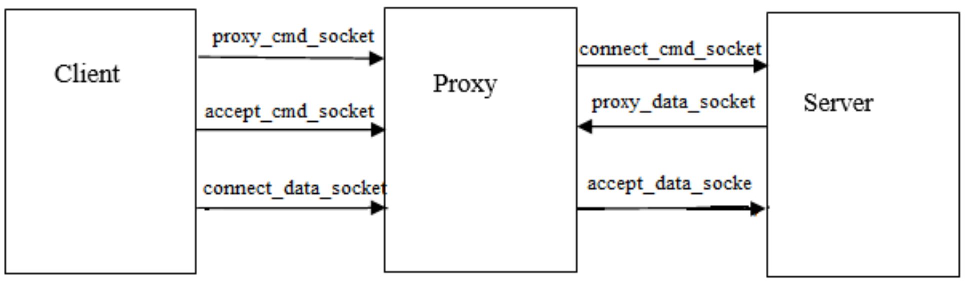
4.2 Design principle (6 sockets & select ())

4.2.1 Socket programming

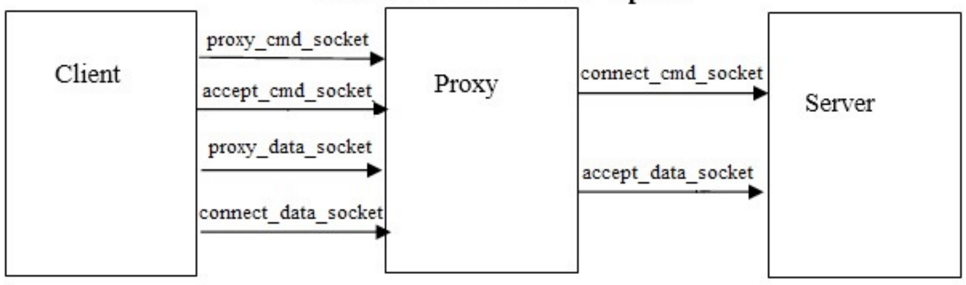
- proxy_cmd_socket (1): Listen the control connection
- accept_cmd_socket (2): Accept client require the control connection
- connect_cmd_socket (3): Connect server build the control connection
- proxy_data_socket (4): Listen the data connection.
- accept_data_socket (5): In active mode accept gets the requirement of building the data connection from the server. In passive mode, accept gets the requirement of data connection from the client.
- connect_data_socket (6): In active mode connect gets the requirement of building the data connection from the client. In passive mode, it gets the server requirement of data connection.

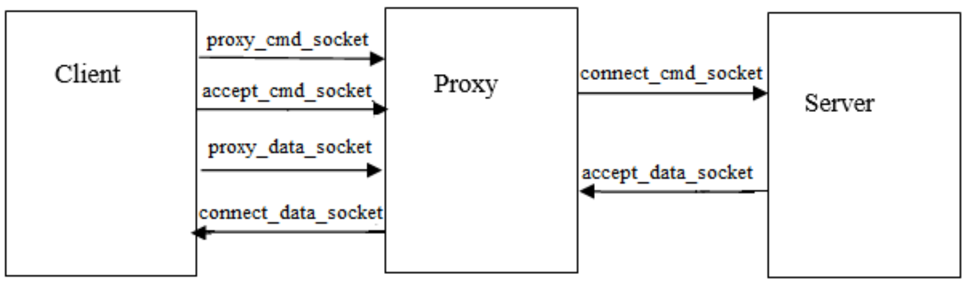
Active mode Download Upload





Passive mode Download Upload





4.2.2 Select() Function

int select (int maxfdp, fd_set *readfds, fd_set *writefds, fd_set *errorfds, struct timeval*timeout);

- struct fd_set：It can be understood as a set, which save the file descriptor. When you use the function select() you will wait here, until the file which is observed change its atate.
- In our programme we use the parameters struct timeval*timeout: which function is set the overtime.
- The type of the return is int: when the value is negative：select has error. When it comes to 0：it means overtime，according to struct timeval*timeout to judge whether li is overtime.

  When it comes to a active: it means it is doing some operation.

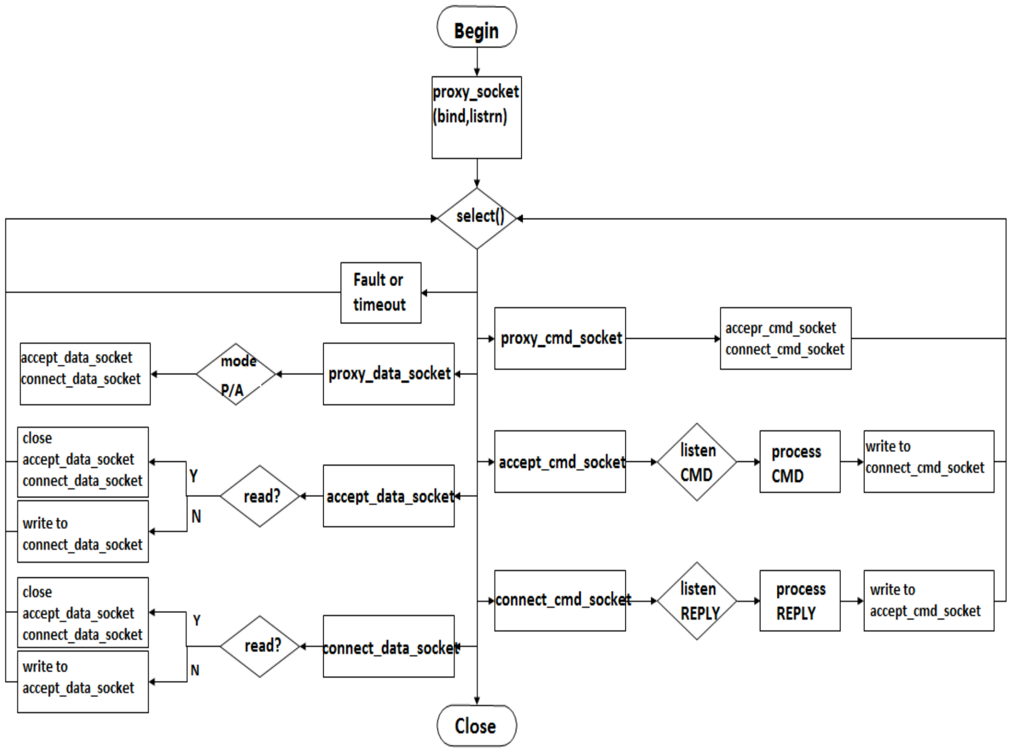
4.3 Flow Path of project

- When we start the project, we should build a socket to bind and listen port 21 and add it to the master set. Then we should start a select () program to circle monitor the operation and protect it from overtime.
- In the proxy_cmd_socket (1) when the socket hears the control connection, it will new two sockets accept_cmd_socket and connect_cmd_socket using the function acceptCmdSocket() and connectToServer(). Then we get the proxy IP dynamically and set the file descriptor into master set.

- In the accept_cmd_socket (2): If the socket cannot accept anything it will close. Otherwise it will accept client require the control connection. When the socket get the command we use function memcmp() to compare the first four letters to the “PORT” and “RETR” to distinguish the operation. Moreover we use a function strchr() to find the first space in the command, and get the data in the command like ”192.168.56.101.1.16 ”.we also should to make sure if the name of file has existed in the cache. If the command is “PORT” we should new a socket proxy_data_socket using the port member we calculate and add it into master set. Finally, we write the command to the server by connect_cmd_socket.
- In the connect_cmd_socket (3): If the socket cannot accept anything it will close. Otherwise it will connect server build the control connection. We should build on the data connection when it is on passive mode. If the command is started with “227” it will new a socket proxy_data_socket using the port member we calculate and add it into master set. Finally, we write the command to the client by accept_cmd_socket.
- In the proxy_data_socket (4): we should listen and build the data connection. If it is the passive mode, we should build accept_data_socket and connect_data_socket. In active mode we use getpeername() to get the server IP and connect it and bulid

accept_data_socket and connect_data_socket.

- In accept_data_socket (5): In active mode accept gets the requirement of building the data connection from the server. In passive mode, accept gets the requirement of data connection from the client. when it read something from the client we will write it to the connect_data_socket.
- When it comes to connect_data_socket (6): In active mode connect gets the requirement of building the data connection from the client. In passive mode, connect gets the requirement of data connection from the server. when it read something from the server we will write it to the accept_data_socket.



Begin
proxy_socket (bind,listrn)
select()
Fault or timeout
proxy_cmd_socket
accepr_cmd_socket connect_cmd_socket
accept_data_socket connect_data_socket
mode P/A
proxy_data_socket
accept_cmd_socket
listen CMD
process CMD
write to connect_cmd_socket
close accept_data_socket connect_data_socket
Y
read?
N
accept_data_socket
write to connect_data_socket
connect_cmd_socket
listen REPLY
process REPLY
write to accept_cmd_socket
close accept_data_socket connect_data_socket
Y
read?
N
connect_data_socket
write to accept_data_socket
Close

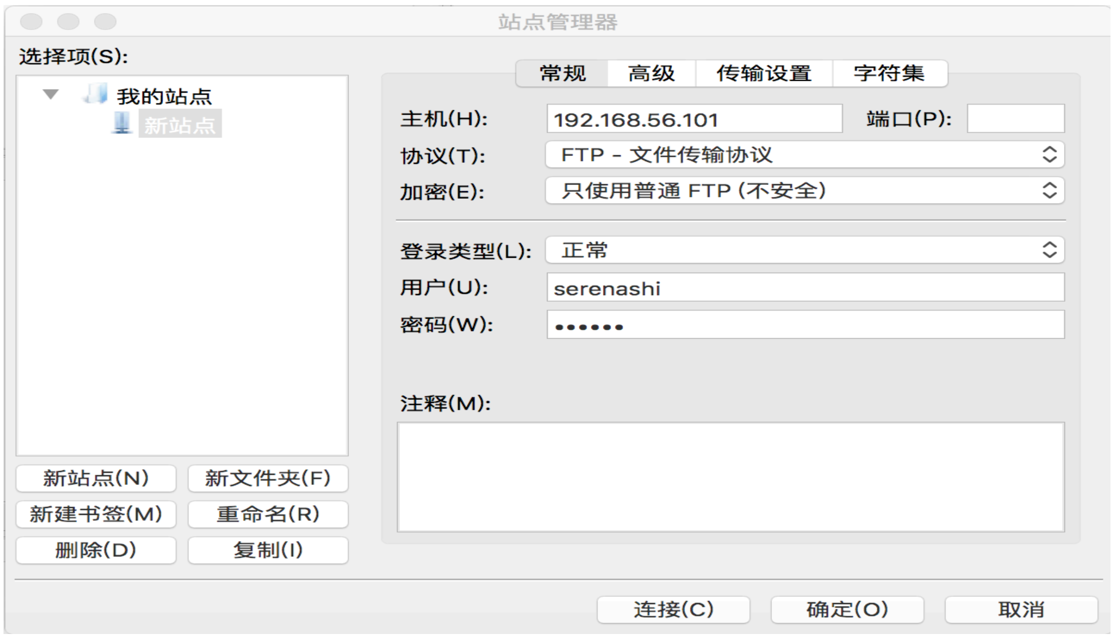
Project Flow Table

5.Results

Choosing download and upload process within passive mode as an example:

Firstly, we set up certain parameters to our FileZilla client

The IP address for the proxy is 192.168.56.101 and the user name and password should be set equal to the server’s computer password.



站点管理器

选择项(S):

我的站点

新站点

常规 | 高级 | 传输设置 | 字符集

主机(H): 192.168.56.101 端口(P):

协议(T): FTP - 文件传输协议

加密(E): 只使用普通 FTP (不安全)

登录类型(L): 正常

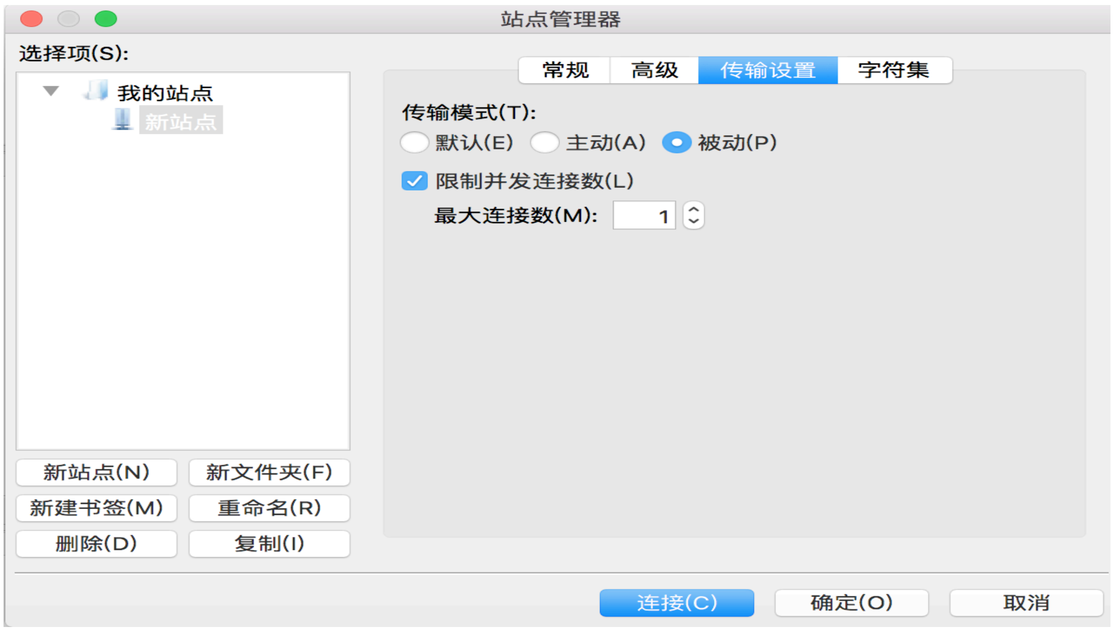
用户(U): serenashi

密码(W): ••••••

注释(M):

新站点(N) 新文件夹(F) 新建书签(M) 重命名(R) 删除(D) 复制(I)

连接(C) 确定(O) 取消



站点管理器

选择项(S):

我的站点

新站点

常规 | 高级 | 传输设置 | 字符集

传输模式(T):

默认(E) 主动(A) [✓ 被动(P)]

[✓] 限制并发连接数(L)

最大连接数(M): 1

新站点(N) 新文件夹(F) 新建书签(M) 重命名(R) 删除(D) 复制(I)

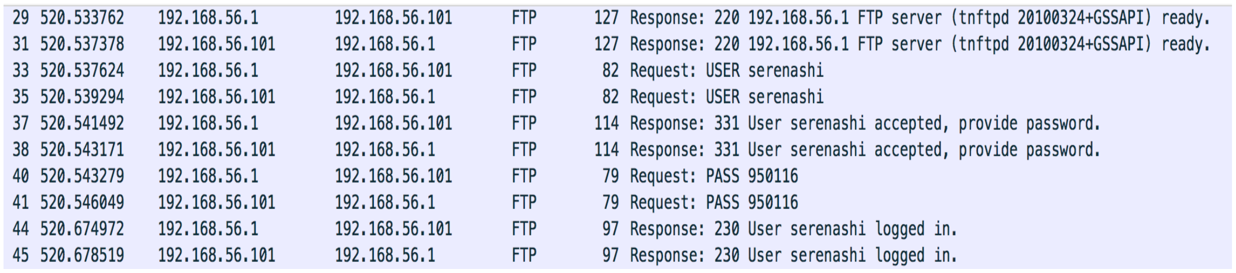
连接(C) 确定(O) 取消

And then open the server in the terminal using command:

sudo -s launchctl load -w /System/Library/LaunchDaemons/ftp.plist

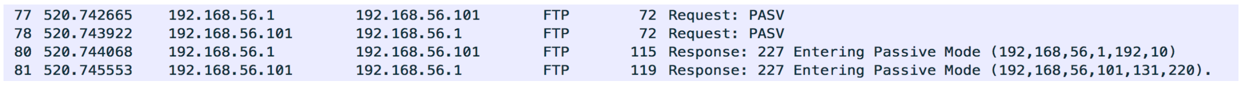
Start the proxy program in our Ubuntu and Using Wireshark to catch certain packets:

1. USER and PASS packets to check if user successfully logged in:



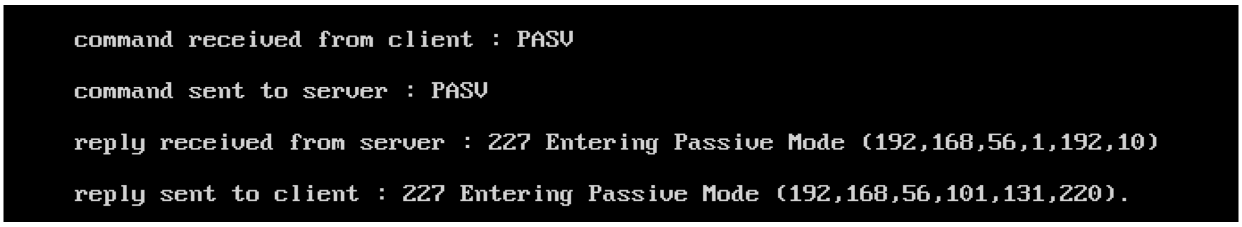
```
29 520.533762   192.168.56.1     192.168.56.101   FTP   127 Response: 220 192.168.56.1 FTP server (tnftpd 20100324+GSSAPI) ready.
31 520.537378   192.168.56.101   192.168.56.1     FTP   127 Response: 220 192.168.56.1 FTP server (tnftpd 20100324+GSSAPI) ready.
33 520.537624   192.168.56.1     192.168.56.101   FTP    82 Request: USER serenashi
35 520.539294   192.168.56.101   192.168.56.1     FTP    82 Request: USER serenashi
37 520.541492   192.168.56.1     192.168.56.101   FTP   114 Response: 331 User serenashi accepted, provide password.
38 520.543171   192.168.56.101   192.168.56.1     FTP   114 Response: 331 User serenashi accepted, provide password.
40 520.543279   192.168.56.1     192.168.56.101   FTP    79 Request: PASS 950116
41 520.546049   192.168.56.101   192.168.56.1     FTP    79 Request: PASS 950116
44 520.674972   192.168.56.1     192.168.56.101   FTP    97 Response: 230 User serenashi logged in.
45 520.678519   192.168.56.101   192.168.56.1     FTP    97 Response: 230 User serenashi logged in.
```

2. Passive mode:



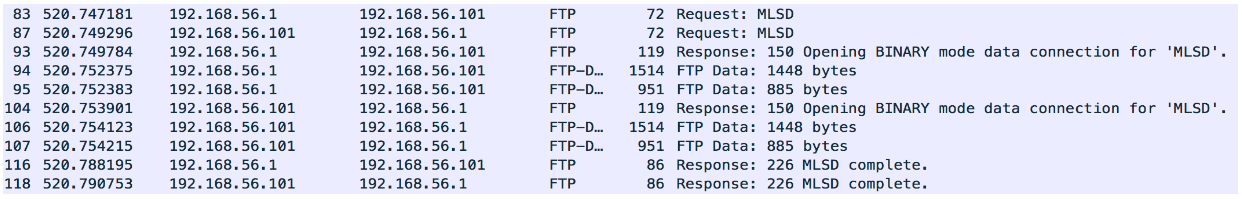
```
77 520.742665   192.168.56.1     192.168.56.101   FTP    72 Request: PASV
78 520.743922   192.168.56.101   192.168.56.1     FTP    72 Request: PASV
80 520.744068   192.168.56.1     192.168.56.101   FTP   115 Response: 227 Entering Passive Mode (192,168,56,1,192,10)
81 520.745553   192.168.56.101   192.168.56.1     FTP   119 Response: 227 Entering Passive Mode (192,168,56,101,131,220).
```

The message in the proxy:

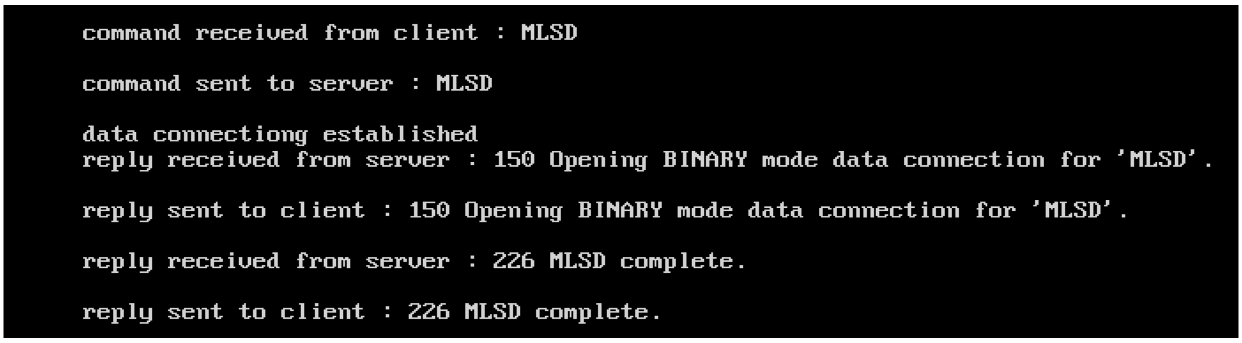


```
command received from client : PASV

command sent to server : PASV

reply received from server : 227 Entering Passive Mode (192,168,56,1,192,10)

reply sent to client : 227 Entering Passive Mode (192,168,56,101,131,220).
```

3. Pass the list

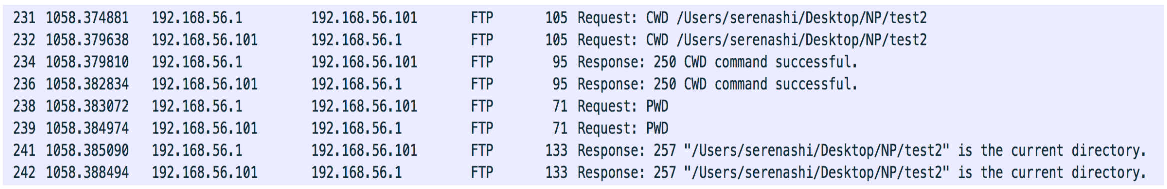


```
 83 520.747181   192.168.56.1     192.168.56.101   FTP      72 Request: MLSD
 87 520.749296   192.168.56.101   192.168.56.1     FTP      72 Request: MLSD
 93 520.749784   192.168.56.1     192.168.56.101   FTP     119 Response: 150 Opening BINARY mode data connection for 'MLSD'.
 94 520.752375   192.168.56.1     192.168.56.101   FTP-D…  1514 FTP Data: 1448 bytes
 95 520.752383   192.168.56.1     192.168.56.101   FTP-D…   951 FTP Data: 885 bytes
104 520.753901   192.168.56.101   192.168.56.1     FTP     119 Response: 150 Opening BINARY mode data connection for 'MLSD'.
106 520.754123   192.168.56.101   192.168.56.1     FTP-D…  1514 FTP Data: 1448 bytes
107 520.754215   192.168.56.101   192.168.56.1     FTP-D…   951 FTP Data: 885 bytes
116 520.788195   192.168.56.1     192.168.56.101   FTP      86 Response: 226 MLSD complete.
118 520.790753   192.168.56.101   192.168.56.1     FTP      86 Response: 226 MLSD complete.
```



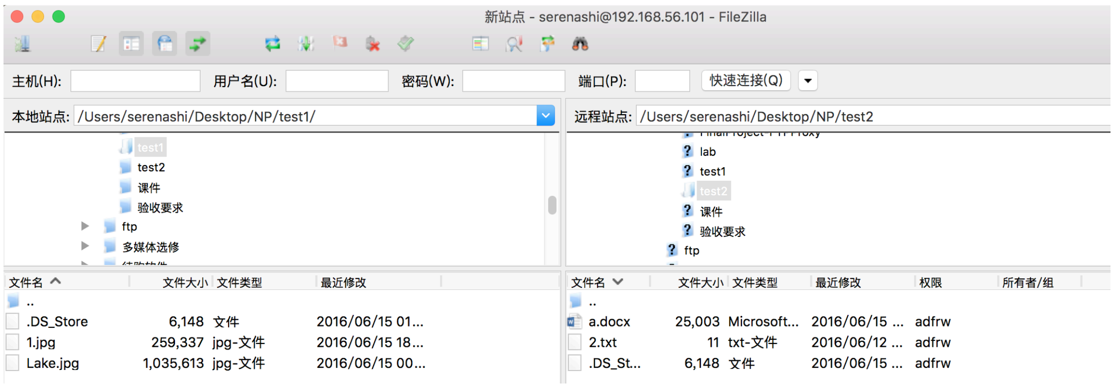
```
command received from client : MLSD

command sent to server : MLSD

data connectiong established
reply received from server : 150 Opening BINARY mode data connection for 'MLSD'.

reply sent to client : 150 Opening BINARY mode data connection for 'MLSD'.

reply received from server : 226 MLSD complete.

reply sent to client : 226 MLSD complete.
```

4. Change the directory and print the current directory



```
231 1058.374881  192.168.56.1     192.168.56.101   FTP   105 Request: CWD /Users/serenashi/Desktop/NP/test2
232 1058.379638  192.168.56.101   192.168.56.1     FTP   105 Request: CWD /Users/serenashi/Desktop/NP/test2
234 1058.379810  192.168.56.1     192.168.56.101   FTP    95 Response: 250 CWD command successful.
236 1058.382834  192.168.56.101   192.168.56.1     FTP    95 Response: 250 CWD command successful.
238 1058.383072  192.168.56.1     192.168.56.101   FTP    71 Request: PWD
239 1058.384974  192.168.56.101   192.168.56.1     FTP    71 Request: PWD
241 1058.385090  192.168.56.1     192.168.56.101   FTP   133 Response: 257 "/Users/serenashi/Desktop/NP/test2" is the current directory.
242 1058.388494  192.168.56.101   192.168.56.1     FTP   133 Response: 257 "/Users/serenashi/Desktop/NP/test2" is the current directory.
```

5. The initial state in FileZilla before we transfer the data



新站点 - serenashi@192.168.56.101 - FileZilla

主机(H): 用户名(U): 密码(W): 端口(P): 快速连接(Q)

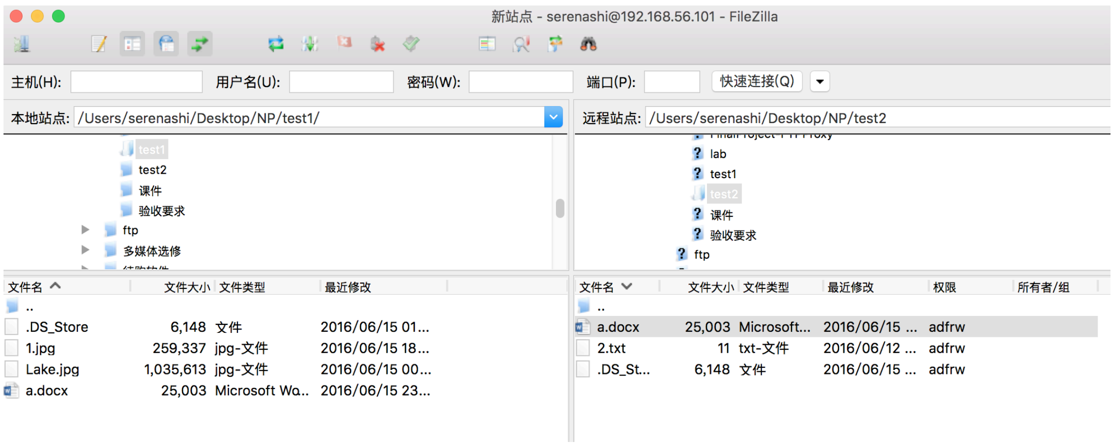
本地站点: /Users/serenashi/Desktop/NP/test1/

远程站点: /Users/serenashi/Desktop/NP/test2

文件名	文件大小	文件类型	最近修改
..			
.DS_Store	6,148	文件	2016/06/15 01...
1.jpg	259,337	jpg-文件	2016/06/15 18...
Lake.jpg	1,035,613	jpg-文件	2016/06/15 00...

文件名	文件大小	文件类型	最近修改	权限	所有者/组
..					
a.docx	25,003	Microsoft...	2016/06/15 ...	adfrw	
2.txt	11	txt-文件	2016/06/12 ...	adfrw	
.DS_St...	6,148	文件	2016/06/15 ...	adfrw	

6. When we download the file a.docx from the server to the client



新站点 - serenashi@192.168.56.101 - FileZilla

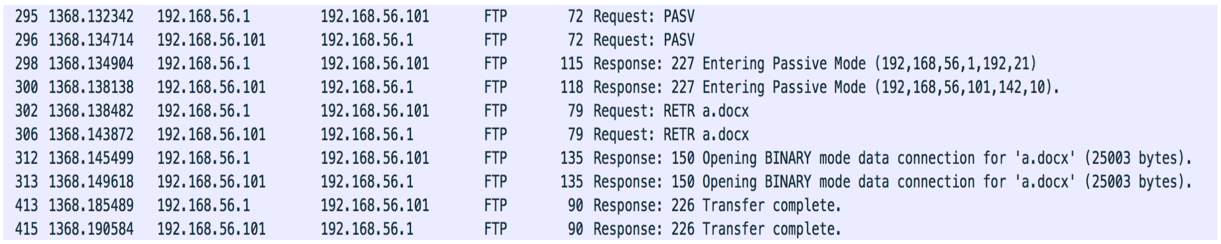
主机(H): 用户名(U): 密码(W): 端口(P): 快速连接(Q)

本地站点: /Users/serenashi/Desktop/NP/test1/

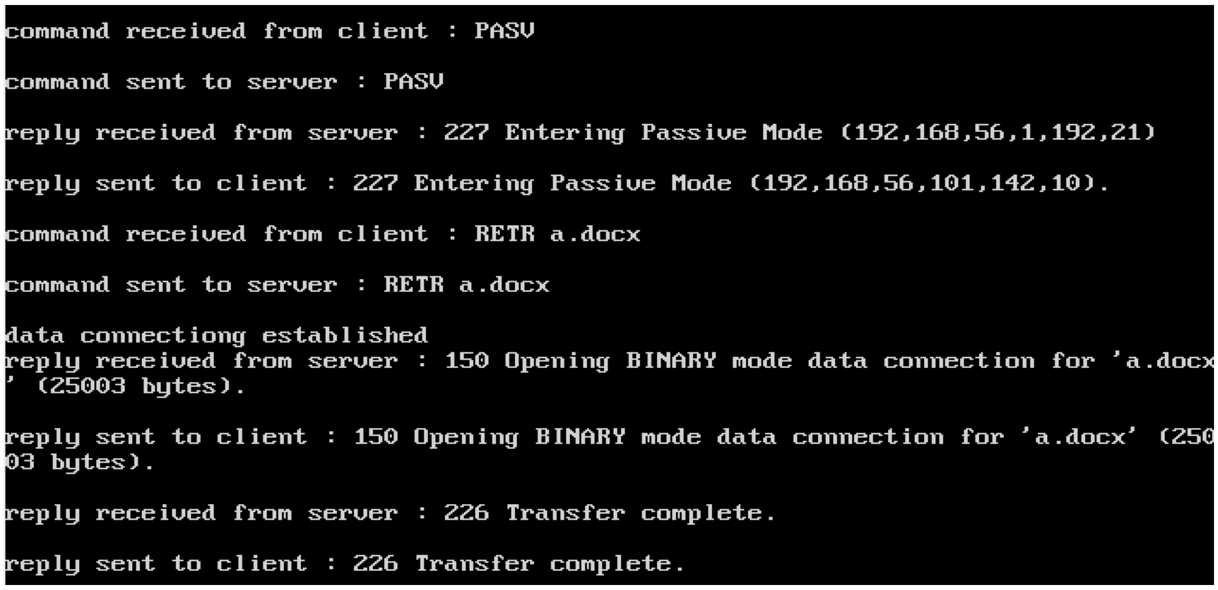
远程站点: /Users/serenashi/Desktop/NP/test2

文件名	文件大小	文件类型	最近修改
..			
.DS_Store	6,148	文件	2016/06/15 01...
1.jpg	259,337	jpg-文件	2016/06/15 18...
Lake.jpg	1,035,613	jpg-文件	2016/06/15 00...
a.docx	25,003	Microsoft Wa..	2016/06/15 23...

文件名	文件大小	文件类型	最近修改	权限	所有者/组
..					
a.docx	25,003	Microsoft...	2016/06/15 ...	adfrw	
2.txt	11	txt-文件	2016/06/12 ...	adfrw	
.DS_St...	6,148	文件	2016/06/15 ...	adfrw	

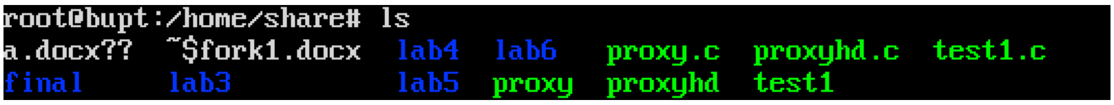


```
295 1368.132342  192.168.56.1     192.168.56.101   FTP    72 Request: PASV
296 1368.134714  192.168.56.101   192.168.56.1     FTP    72 Request: PASV
298 1368.134904  192.168.56.1     192.168.56.101   FTP   115 Response: 227 Entering Passive Mode (192,168,56,1,192,21)
300 1368.138138  192.168.56.101   192.168.56.1     FTP   118 Response: 227 Entering Passive Mode (192,168,56,101,142,10).
302 1368.138482  192.168.56.1     192.168.56.101   FTP    79 Request: RETR a.docx
306 1368.143872  192.168.56.101   192.168.56.1     FTP    79 Request: RETR a.docx
312 1368.145499  192.168.56.1     192.168.56.101   FTP   135 Response: 150 Opening BINARY mode data connection for 'a.docx' (25003 bytes).
313 1368.149618  192.168.56.101   192.168.56.1     FTP   135 Response: 150 Opening BINARY mode data connection for 'a.docx' (25003 bytes).
413 1368.185489  192.168.56.1     192.168.56.101   FTP    90 Response: 226 Transfer complete.
415 1368.190584  192.168.56.101   192.168.56.1     FTP    90 Response: 226 Transfer complete.
```



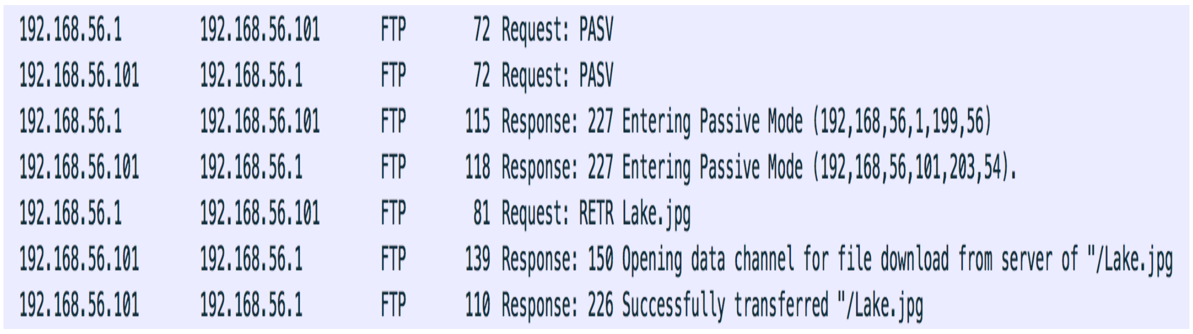
```
command received from client : PASV

command sent to server : PASV

reply received from server : 227 Entering Passive Mode (192,168,56,1,192,21)

reply sent to client : 227 Entering Passive Mode (192,168,56,101,142,10).

command received from client : RETR a.docx

command sent to server : RETR a.docx

data connectiong established
reply received from server : 150 Opening BINARY mode data connection for 'a.docx
' (25003 bytes).

reply sent to client : 150 Opening BINARY mode data connection for 'a.docx' (250
03 bytes).

reply received from server : 226 Transfer complete.

reply sent to client : 226 Transfer complete.
```

7. We store the file a.docx in the cache after one downloading transmission from the server



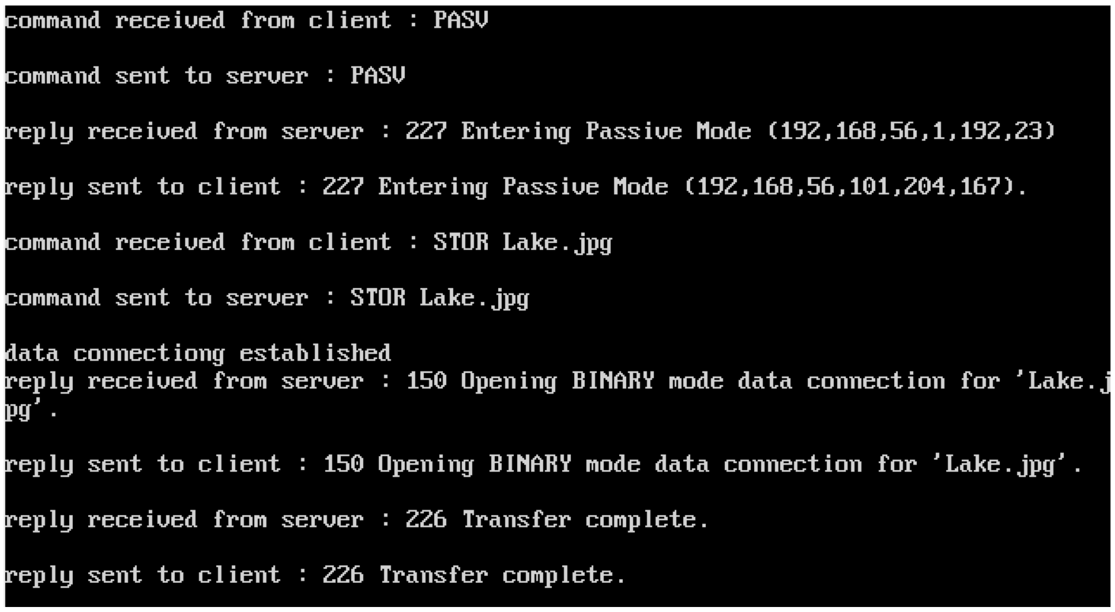
```
root@bupt:/home/share# ls
a.docx??  ~$fork1.docx  lab4  lab6   proxy.c  proxyhd.c  test1.c
final     lab3          lab5  proxy  proxyhd  test1
```

8. If you want to download the same file, you will find that the proxy will show that "The file already exists in the proxy." And won't start the data connection, download the file directly from the cache.

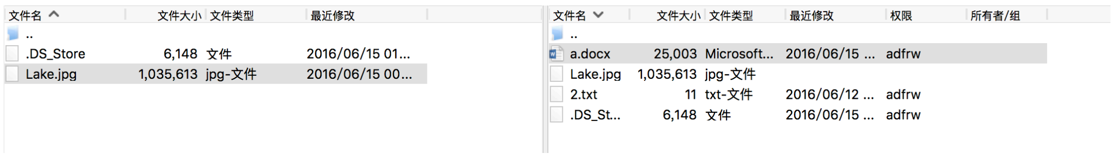


```
192.168.56.1     192.168.56.101   FTP    72 Request: PASV
192.168.56.101   192.168.56.1     FTP    72 Request: PASV
192.168.56.1     192.168.56.101   FTP   115 Response: 227 Entering Passive Mode (192,168,56,1,199,56)
192.168.56.101   192.168.56.1     FTP   118 Response: 227 Entering Passive Mode (192,168,56,101,203,54).
192.168.56.1     192.168.56.101   FTP    81 Request: RETR Lake.jpg
192.168.56.101   192.168.56.1     FTP   139 Response: 150 Opening data channel for file download from server of "/Lake.jpg
192.168.56.101   192.168.56.1     FTP   110 Response: 226 Successfully transferred "/Lake.jpg
```

9. Upload the file

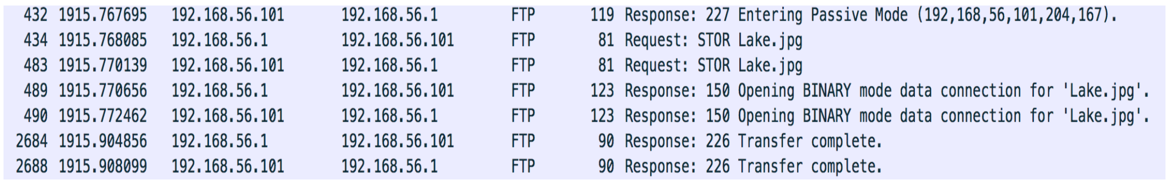


```
command received from client : PASV

command sent to server : PASV

reply received from server : 227 Entering Passive Mode (192,168,56,1,192,23)

reply sent to client : 227 Entering Passive Mode (192,168,56,101,204,167).

command received from client : STOR Lake.jpg

command sent to server : STOR Lake.jpg

data connectiong established
reply received from server : 150 Opening BINARY mode data connection for 'Lake.j
pg'.

reply sent to client : 150 Opening BINARY mode data connection for 'Lake.jpg'.

reply received from server : 226 Transfer complete.

reply sent to client : 226 Transfer complete.
```



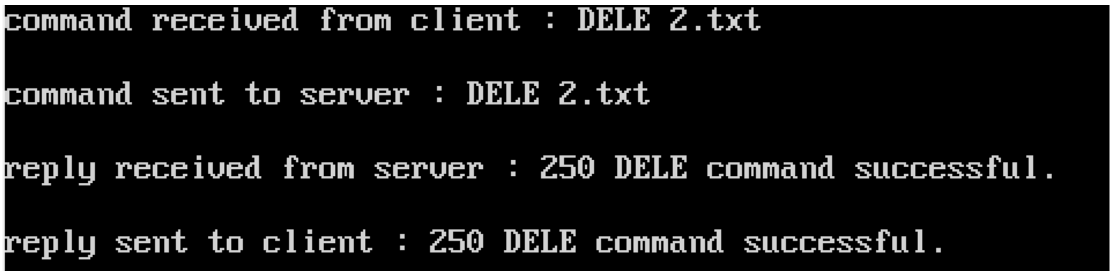
文件名	文件大小	文件类型	最近修改
..			
.DS_Store	6,148	文件	2016/06/15 01...
Lake.jpg	1,035,613	jpg-文件	2016/06/15 00...

文件名	文件大小	文件类型	最近修改	权限	所有者/组
..					
a.docx	25,003	Microsoft...	2016/06/15 ...	adfrw	
Lake.jpg	1,035,613	jpg-文件			
2.txt	11	txt-文件	2016/06/12 ...	adfrw	
.DS_St...	6,148	文件	2016/06/15 ...	adfrw	

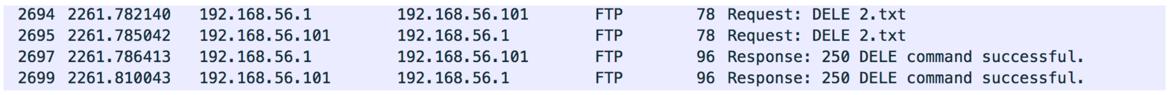


```
 432  1915.767695  192.168.56.101  192.168.56.1    FTP  119 Response: 227 Entering Passive Mode (192,168,56,101,204,167).
 434  1915.768085  192.168.56.1    192.168.56.101  FTP   81 Request: STOR Lake.jpg
 483  1915.770139  192.168.56.101  192.168.56.1    FTP   81 Request: STOR Lake.jpg
 489  1915.770656  192.168.56.1    192.168.56.101  FTP  123 Response: 150 Opening BINARY mode data connection for 'Lake.jpg'.
 490  1915.772462  192.168.56.101  192.168.56.1    FTP  123 Response: 150 Opening BINARY mode data connection for 'Lake.jpg'.
2684  1915.904856  192.168.56.1    192.168.56.101  FTP   90 Response: 226 Transfer complete.
2688  1915.908099  192.168.56.101  192.168.56.1    FTP   90 Response: 226 Transfer complete.
```

10.Delete a file

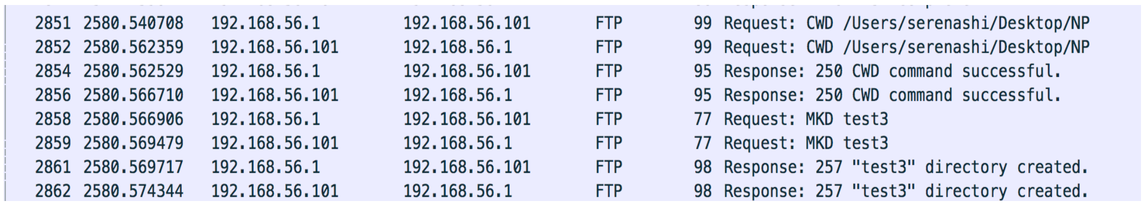


```
command received from client : DELE 2.txt

command sent to server : DELE 2.txt

reply received from server : 250 DELE command successful.

reply sent to client : 250 DELE command successful.
```

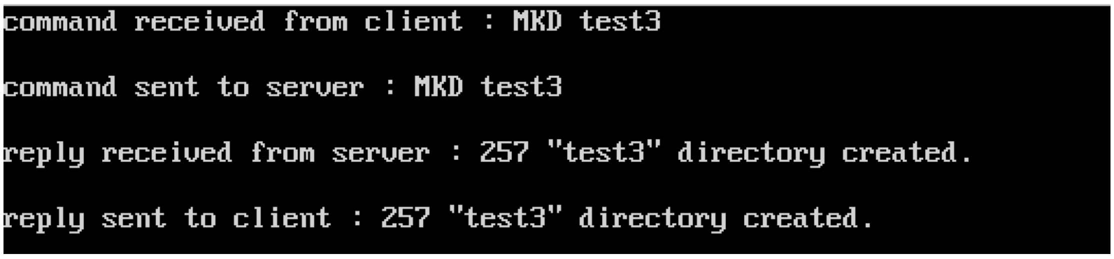


```
2694  2261.782140   192.168.56.1     192.168.56.101   FTP   78  Request: DELE 2.txt
2695  2261.785042   192.168.56.101   192.168.56.1     FTP   78  Request: DELE 2.txt
2697  2261.786413   192.168.56.1     192.168.56.101   FTP   96  Response: 250 DELE command successful.
2699  2261.810043   192.168.56.101   192.168.56.1     FTP   96  Response: 250 DELE command successful.
```

11.Create a new directory in the server



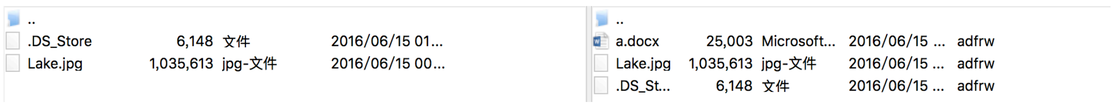
```
2851  2580.540708   192.168.56.1         192.168.56.101       FTP        99  Request: CWD /Users/serenashi/Desktop/NP
2852  2580.562359   192.168.56.101       192.168.56.1         FTP        99  Request: CWD /Users/serenashi/Desktop/NP
2854  2580.562529   192.168.56.1         192.168.56.101       FTP        95  Response: 250 CWD command successful.
2856  2580.566710   192.168.56.101       192.168.56.1         FTP        95  Response: 250 CWD command successful.
2858  2580.566906   192.168.56.1         192.168.56.101       FTP        77  Request: MKD test3
2859  2580.569479   192.168.56.101       192.168.56.1         FTP        77  Request: MKD test3
2861  2580.569717   192.168.56.1         192.168.56.101       FTP        98  Response: 257 "test3" directory created.
2862  2580.574344   192.168.56.101       192.168.56.1         FTP        98  Response: 257 "test3" directory created.
```



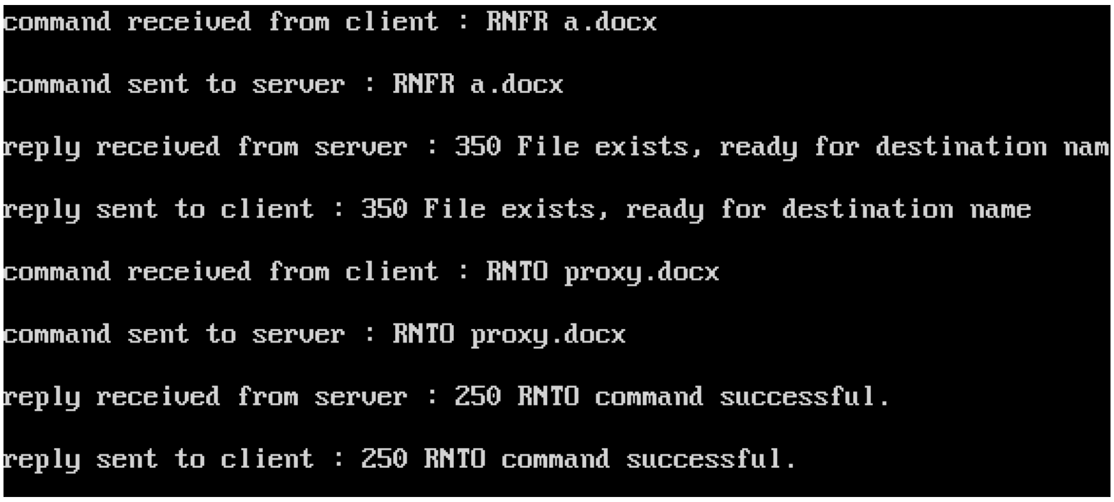
```
command received from client : MKD test3

command sent to server : MKD test3

reply received from server : 257 "test3" directory created.

reply sent to client : 257 "test3" directory created.
```

12.Rename a filename from a.docx to proxy.docx in server

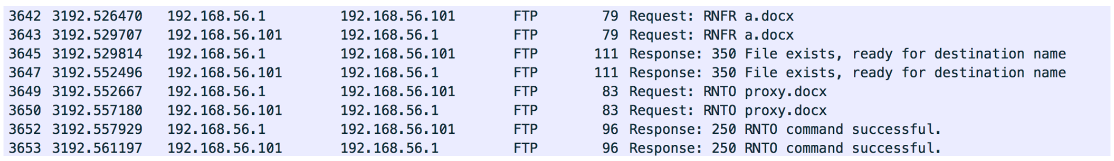
Client initial state:



```
..                                                      ..
.DS_Store        6,148  文件      2016/06/15 01...      a.docx      25,003  Microsoft...  2016/06/15 ...  adfrw
Lake.jpg     1,035,613  jpg-文件  2016/06/15 00...      Lake.jpg 1,035,613  jpg-文件      2016/06/15 ...  adfrw
                                                        .DS_St...    6,148  文件          2016/06/15 ...  adfrw
```

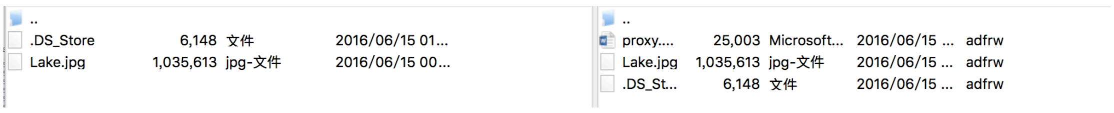


```
command received from client : RNFR a.docx

command sent to server : RNFR a.docx

reply received from server : 350 File exists, ready for destination nam

reply sent to client : 350 File exists, ready for destination name

command received from client : RNTO proxy.docx

command sent to server : RNTO proxy.docx

reply received from server : 250 RNTO command successful.

reply sent to client : 250 RNTO command successful.
```



```
3642  3192.526470   192.168.56.1       192.168.56.101     FTP       79  Request: RNFR a.docx
3643  3192.529707   192.168.56.101     192.168.56.1       FTP       79  Request: RNFR a.docx
3645  3192.529814   192.168.56.1       192.168.56.101     FTP      111  Response: 350 File exists, ready for destination name
3647  3192.552496   192.168.56.101     192.168.56.1       FTP      111  Response: 350 File exists, ready for destination name
3649  3192.552667   192.168.56.1       192.168.56.101     FTP       83  Request: RNTO proxy.docx
3650  3192.557180   192.168.56.101     192.168.56.1       FTP       83  Request: RNTO proxy.docx
3652  3192.557929   192.168.56.1       192.168.56.101     FTP       96  Response: 250 RNTO command successful.
3653  3192.561197   192.168.56.101     192.168.56.1       FTP       96  Response: 250 RNTO command successful.
```

client final state:



```
..                                                      ..
.DS_Store        6,148  文件      2016/06/15 01...      proxy....   25,003  Microsoft...  2016/06/15 ...  adfrw
Lake.jpg     1,035,613  jpg-文件  2016/06/15 00...      Lake.jpg 1,035,613  jpg-文件      2016/06/15 ...  adfrw
                                                        .DS_St...    6,148  文件          2016/06/15 ...  adfrw
```