Project report

Final Project-FTP Proxy

Course Title: *Internet Applications*

Name: *CHANG Yuan(2013213133)*

LIU Weiqiu (2013213152)

Date: *June 15th, 2016*

1. Overview

Through this project, our group completed a FTP proxy program which implements required functionality based on provided framework, thus understanding the related knowledge of FTP deeper.

2. Requirements Analysis

This FTP proxy program based on Linux command line terminal. We used C language to implement coding and gcc compiler to convert our code into executable program, and this proxy implemented requirements as following:

1. This proxy is able to establish control connection with FTP client and FTP server separately.

It will listen client request at port 21 and accept the connection.

It will then connect to FTP server.

2. This proxy is able to establish data connection with FTP client and FTP server separately in two modes (active mode and passive mode).

Under passive mode:

It will send proxy port information to FTP client through control connection, and then listen and accept the data connection.

It will connect to FTP server port which is given through control connection.

It will support receiving and forward commands including PWD, CWD, LIST/MLSD, MDIR, DELE, RNFR/RNTO, RETR, and STOR.

It will support file transmission.

After successful data transmission, this data connection will be break down.

Under active mode:

It will connect to FTP client port which is given through control connection.

It will send proxy port information to FTP server through control connection, and then listen and accept the data connection.

t will support receiving and forward commands including PWD, CWD, LIST/MLSD, MDIR, DELE, RNFR/RNTO, RETR, and STOR.

It will support file transmission.

After successful data transmission, this data connection will be break down.

3. This proxy will be able to provide cache for file transmission.

When receiving RETR command, proxy will check whether the required file is already in the local cache. If not, proxy will download this file from FTP server into local cache and then forward this file to FTP client; If this file already exist, proxy will not download it from FTP server, just sending the local copy to the FTP client.

3. Preliminary Design

3.1 Module Decomposition

The communication between proxy and FTP client or FTP server is governed by TCP/IP protocol suit, and in transport layer, different sockets define the channels for each message to be send to. To sum up, we have 6 sockets to establish and implement the communication, while three of them work in control connection and the others work in data connection. The frame work already provided a logical structure and the select function to detect changes in different sockets, so we can decompose functions of this proxy into 6 parts with respect to 6 sockets.

3.1.1 proxy_cmd_socket.

This socket is responsible for listening FTP client request for initializing control connection, because the control connection will only be initiated by FTP client actively.

However, this socket is not responsible for any other commands` or replies` receiving and forwarding. What`s more, it only works for establish control connection, with no relationship of data connection.

This socket works at port 21.

3.1.2 accept_cmd_socket.

This socket is responsible for communication between proxy and FTP client in the control connection. All the commands send by FTP client and replies send by proxy should go through this socket.

However, this socket is not responsible for any commands` or replies` receiving and forwarding between proxy and FTP server. What`s more, it only works for control connection, no data or file will be send through this socket.

This socket works at port 21.

3.1.3 connect_cmd_socket

This socket is responsible for communication between proxy and FTP server in the control connection. All the commands send by proxy and replies send by FTP server should go through this socket.

However, this socket is not responsible for any commands` or replies` receiving and forwarding between proxy and FTP client. What`s more, it only works for control connection, no data or file will be send through this socket.

This socket works at an ephemeral port which will be assigned by the system.

3.1.4 proxy_data_socket

This socket is responsible for listening request for establishing control connection. Under active mode, this establishing request will be send by FTP

server, under passive mode, this request will be send by FTP client.

However, this socket is not responsible for any commands` or replies` receiving and forwarding ,What`s more, no data or file will be send through this socket.

This socket works at assigned port 20000.(plus 1 when transmission finished)

3.1.5 accept_data_socket

This socket is responsible for data transmission in the data connection. Under active mode, this transmission will between proxy and FTP server, under passive mode, this transmission will between proxy and FTP client.

This socket works at assigned port 20000.(plus 1 when transmission finished)

3.1.6 connect_data_socket

This socket is responsible for data transmission in the data connection. Under active mode, this transmission will between proxy and FTP client, under passive mode, this transmission will between proxy and FTP server.

In active mode, this socket works at 20 port. While in passive mode, this socket works at an ephemeral port which will be assigned by the system.

3.2 Dependencies between modules

3.2.1 Dependencies within the control connection

(1)Proxy_cmd_socket will listen to FTP client, save the client IP and PORT information which will be used by accept_cmd_socket.

(2)Accept_cmd_socket and connect_cmd_socket will receive and forward the control information (commands and replies) which transmitted between each other.

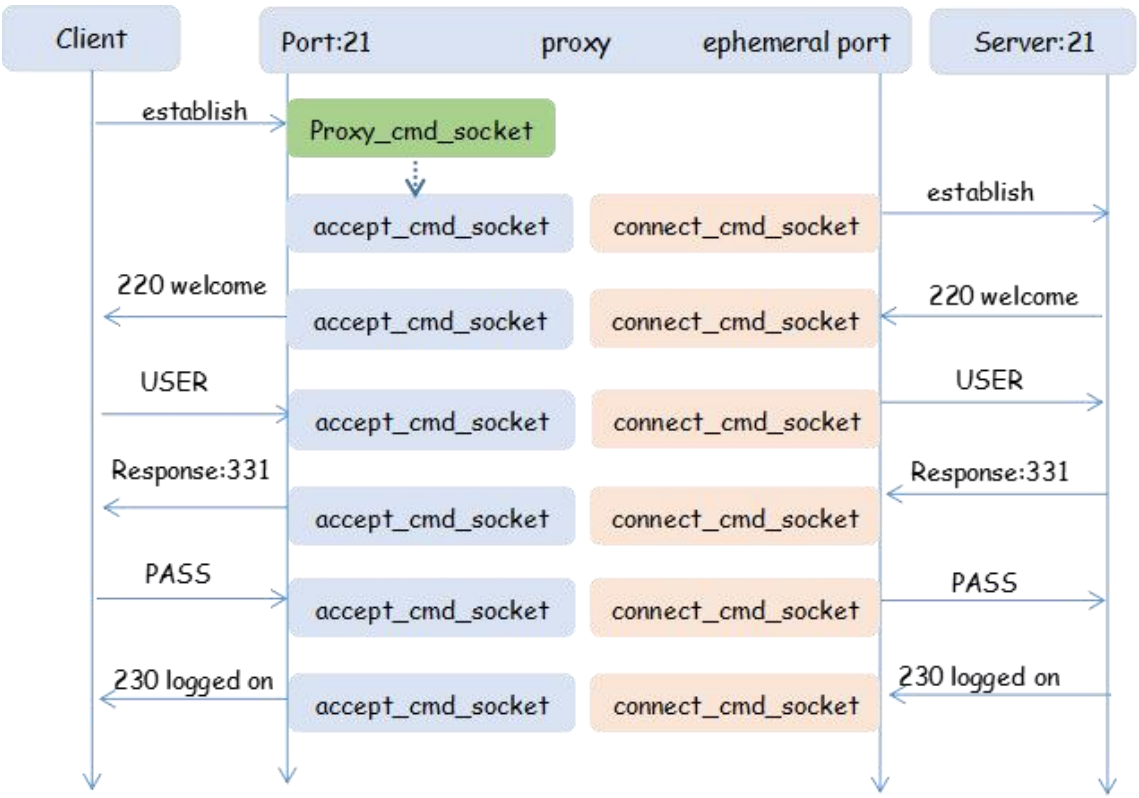


Figure1. control connection establishment

3.2.2 Dependencies within the data connection

(1)Proxy_data_socket will listen to FTP server or FTP client for initializing data connection, save the client IP and PORT information which will be used by accept_data_socket.

(2)Accept_data_socket and connect_data_socket will receive and forward the data (txt , image, pdf files etc.) which transmitted between each other.

3.2.3 Dependencies between the data and control connection

Under ACTIVE mode

accept_cmd_socket will receive and save FTP client PORT information, which will be used by connect_data_socket.

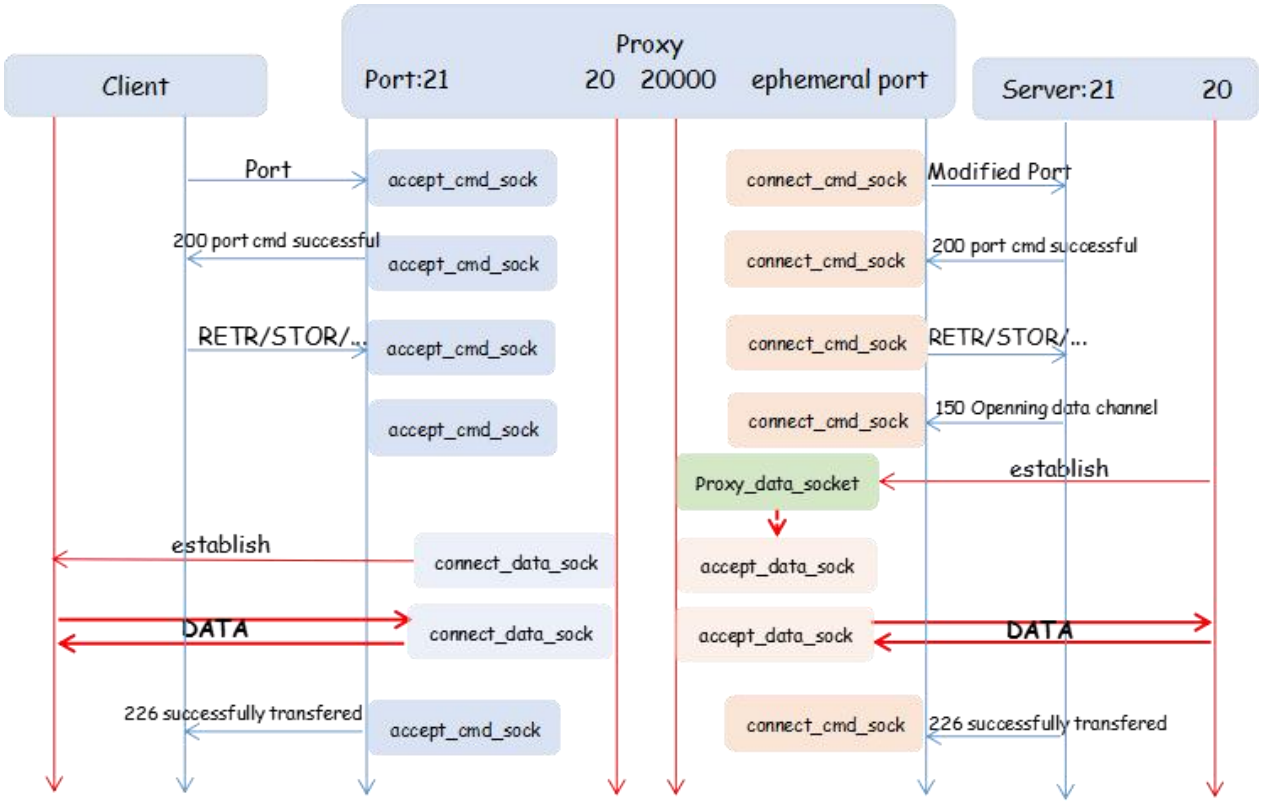


Figure2. Active data connection establishment

Under PASSIVE mode

Connect_cmd_socket will receive FTP server PASV PORT imformation,which will be used by connect_data_socket.

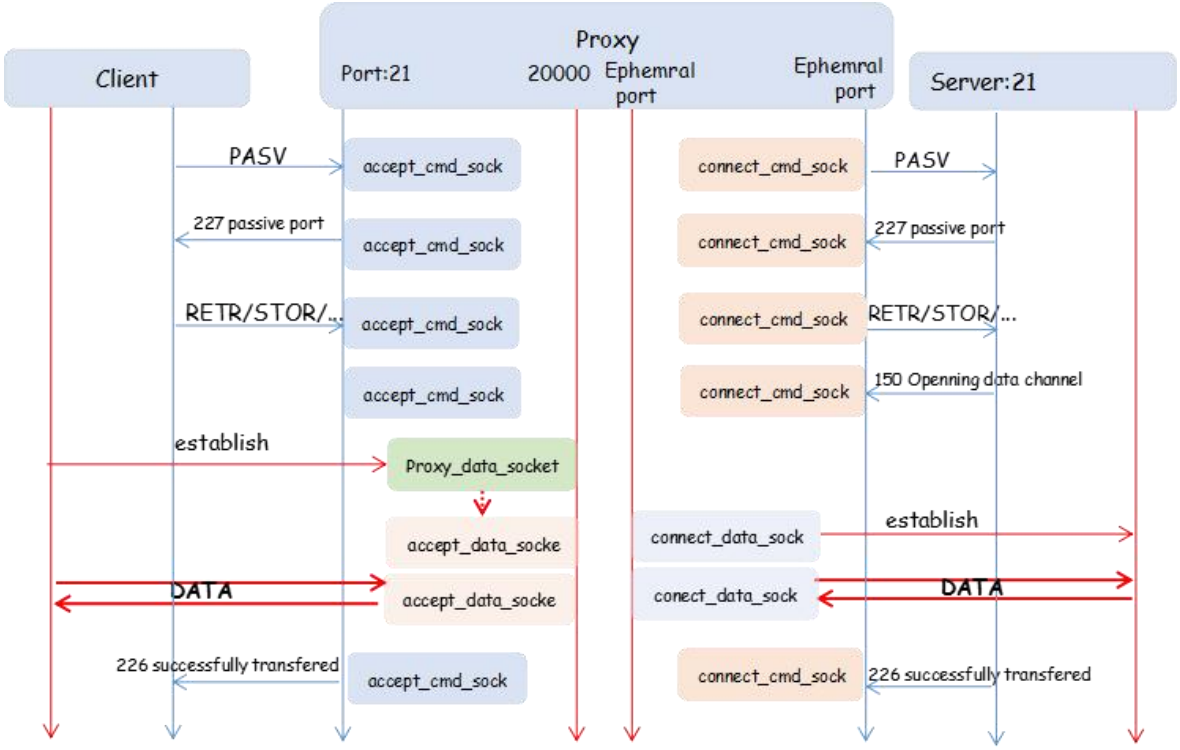


Figure3. passive data connection establishment

4.Detailed Design

4.1Module analysis and design

All modules will work based on the SELECT statement, when select result shows that new events happened in some sockets, proxy will do related process with respect to related process.

4.1.1 select result: proxy_cmd_socket

Based on requirements, this socket will be responsible for following functions:

1.Listen at port 21.

2.When there is request for initialize control connection from client, proxy will immediately establish two new socket: accept_cmd_socket and connect_cmd_socket.

3.It will add new sockets in the master set to ensure that when actions happens in these two socket, the program is able to handle them.

Pseudo code:

```
if(select result includes proxy_cmd_socket){
    accept_cmd_socket=accept(client);
    connect_cmd_socket=connect(server);
}
```

4.1.2 select result: accept_cmd_socket

Based on requirements, this socket will be responsible for following functions:

1.When receiving one packet with no payload(payload length equals to 0), this means FTP client wants to end the control connection. So proxy will close accept_cmd_socket and connect_cmd_socket to break down this control connection, then set the values of the two socket to be 0 as well as remove these two socket in “master_set” to avoid error.

2.Receive client commands and process commands based on their contents.

(1)When receiving “PORT”:

Firstly, proxy will extract the six byte port information and set the data connection mode to be ACTIVE.

Secondly, proxy open one new socket called proxy_data_socket for listening to FTP server for data connection, before the opening, old proxy_data_socket will be closed.

Thirdly, proxy pass the last two byte extracted port information to proxy_data_socket. Finally, proxy refit the 6 byte port information, and save it in one buffer--one character string. This buffer will be used to pass refitted

port information to connect_cmd_socket, in order to establish data connection between proxy and FTP server.

(2)When receiving “RETR”:

Firstly, proxy will extract the file name information.

Secondly, proxy will judge whether this file is already in local cache and the result will be saved in one integer flag called “fileexist”:fileexist will have three value, 1 means file is already in local cache, 2 means required file is not in local cache, and 0 means the command is not RETR , so when receiving other command like STOR or PORT, fileexist will be 0, and in this case, fileexist will only be 1 or 2.

Thirdly, if file not exist, this RETR command will be pass to connect_cmd_socket for forwarding and open new file in local cache to download the required file, if exist, no command will be pass to connect_cmd_socket, proxy will read the file, write required file in accept_data_socket, after sending all the file, proxy will close the file and accept_data_socket

(3)All other commands will be simply received and forward to connect_cmd_socket by using the buffer which is the same one in paragraph(2).

Pseudo code:

```
if(select result includes accept_cmd_socket){
    if(read==0){
        close(sockets);
        Remove sockets from master_set;
    } else{
        if(receive “RETR”){
            if(file exist){
                send data from cache;
            }else{
                Create(file);
            }
        }
        if(receive “PORT”){
            Get port number;
            Change buffer with proxy port;
        }
        Send buffer to connect_cmd_socket;
    }
}
```

4.1.3 select result: connect_cmd_socket

Based on requirements, this socket will be responsible for following

functions:

1.When receiving one packet with no payload(payload length equals to 0), this means FTP server wants to end the control connection. So proxy will close accept_cmd_socket and connect_cmd_socket to break down this control connection, then set the values of the two socket to be 0 as well as remove these two socket in "master_set" to avoid error.

2.Relay client commands and process server replies based on their contents.

(1)When receiving control reply "227", this means FTP server send its port number, so the data connection is established under passive mode. Firstly, proxy will extract the port information and set the data connection mode to be PASSIVE.Secondly, proxy open one new socket called proxy_data_socket for listening to FTP client for data connection, before the opening, old proxy_data_socket will be closed. Thirdly, proxy pass the last two byte extracted port information to proxy_data_socket for listening to FTP client. Finally, proxy refit the 6 byte port information, and save it in one buffer--one character string. This buffer will be used to pass refitted port information to accept_cmd_socket, in order to establish data connection between proxy and FTP client.

(2)All other commands will be simply received and forward to connect_cmd_socket by using the buffer which is the same one as above.

Pseudo code:

```
if(select result includes connect_cmd_socket){
    if(read==0){
        close(socket);
        Remove sockets from master_set;
    } else{
        if(receive "PASV"){
            Get port number;
            Change buffer content with proxy port information;
        }
        Send buffer to accept_cmd_socket;
    }
}
```

4.1.4 select result: proxy_data_socket

Based on requirements, this socket will be responsible for following functions:

1.Listen at an ephemeral port, our group define this port to be 20000.

2.When there is request for establishing data connection, no matter it is from client or server, proxy will immediately establish two new socket: accept_data_socket and connect_data_socket.

3.It will add these two new sockets in the master set to ensure that when

actions happens in these two socket, the program is able to handle them.

Pseudo code:

```
if(select result includes proxy_data_socket){
    accept_cmd_socket=accept(client);//or server
    connect_cmd_socket=connect(server);//or client
}
```

4.1.5 select result: accept_data_socket

Based on requirements, this socket will be responsible for following functions:

1.When receiving one packet with no payload(payload length equals to 0), if the flag "fileexist" shows that file is not in the local cache (equal to 2), this means this data connection is opened for RETR request and file is transmitted successfully and this file is download in cache, so proxy will close the accept_data_connection and connect_data_connection and close the file, then, set value of these two socket to be 0, and remove them from master_set to avoid error.

2.When file is not exist(fileexist==2) and read result is not equal to 0, this means required file is still transmitting, so proxy will send required file information.

3.other data information will be simply send.

Pseudo code:

```
if(select result includes accept_data_socket){
    if(read==0){
        if(RETR file){
            Close(file);
        }
        close(sockets);
        Remove sockets from master_set;
    } else{
        if(RETR file){
            Store file data in cache;
        }
        forward data to connect_cmd_socket;
    }
}
```

4.1.6 select result: connect_data_socket

Based on requirements, this socket will be responsible for following functions:

1. When receiving one packet with no payload(payload length equals to 0),

if the flag “fileexist” shows that file is not in the local cache (equal to 2), this means this data connection is opened for RETR request and file is transmitted successfully and this file is download in cache, so proxy will close the accept_data_connection and connect_data_connection and close the file, then, set value of these two socket to be 0, and remove them from master_set to avoid error.

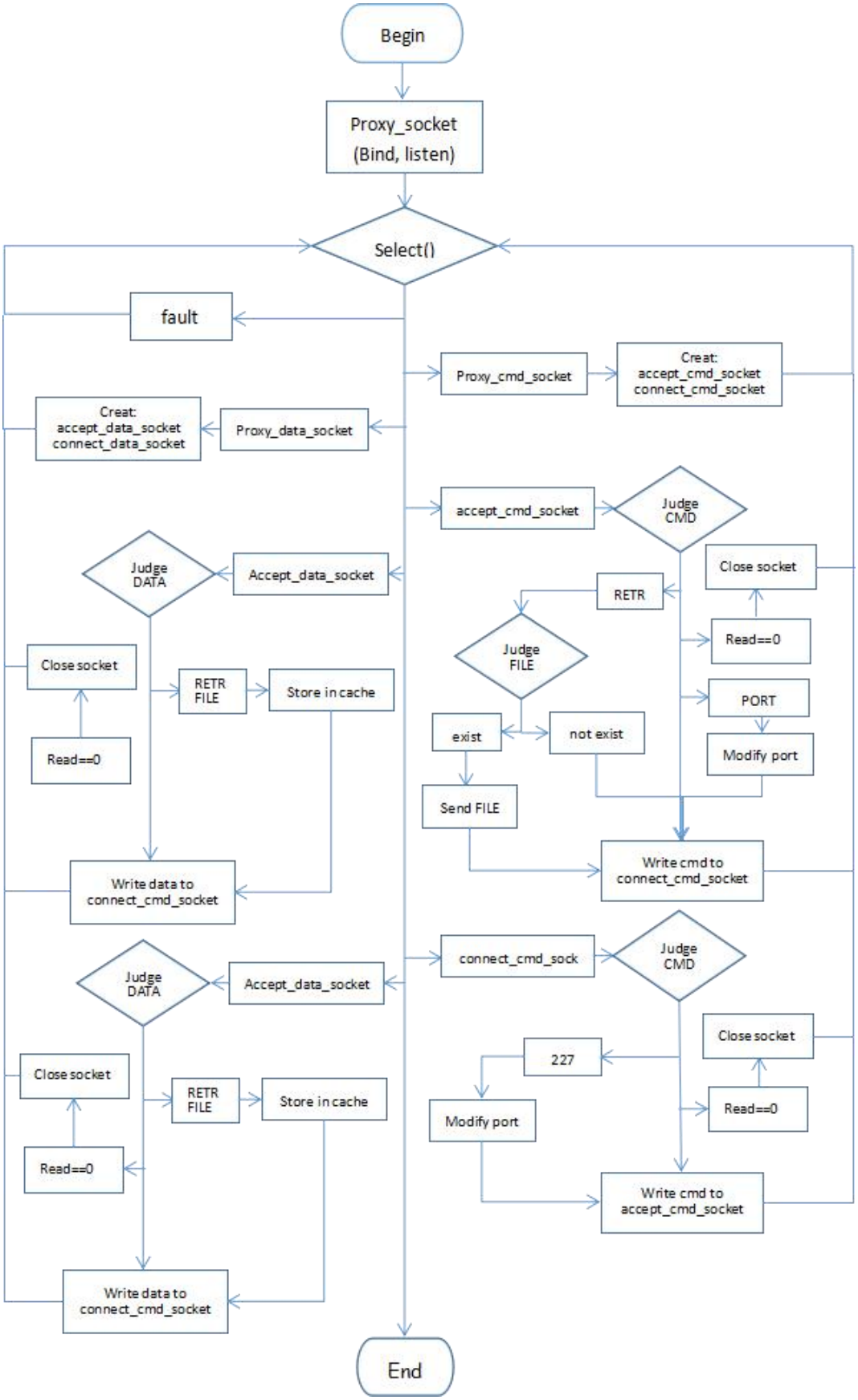
2. proxy will simply forward every information to accept_data_socket.

3. when fileexist==2, this means required file is not exist in cache, so proxy will write received data into one file which is opened at accept_cmd_socket.

Pseudo code:

```
if(select result includes connect_data_socket){
    if(read==0){
        if(RETR file){
            Close(file);
        }
        close(socket);
        Remove sockets from master_set;
    } else{
        if(RETR file){
            Store file data in cache;
        }
        forward data to accept_cmd_socket;
    }
}
```

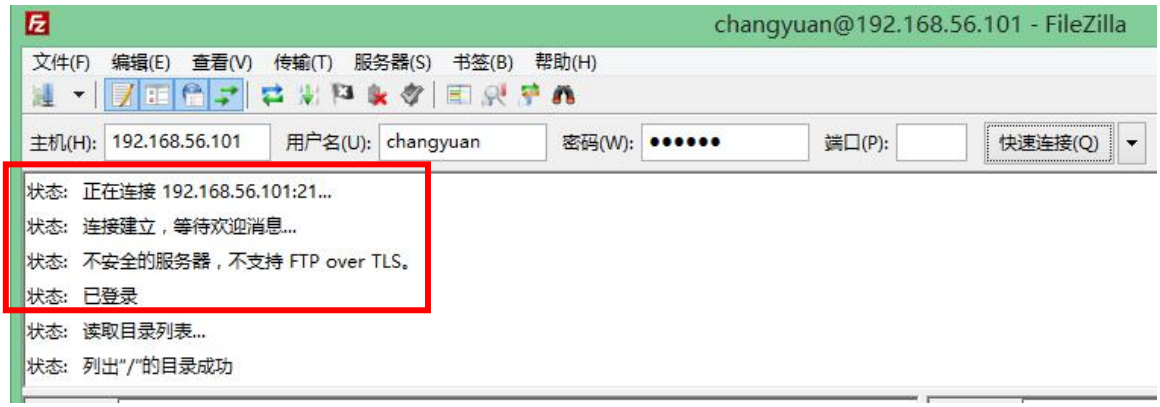
4.2 flow chart



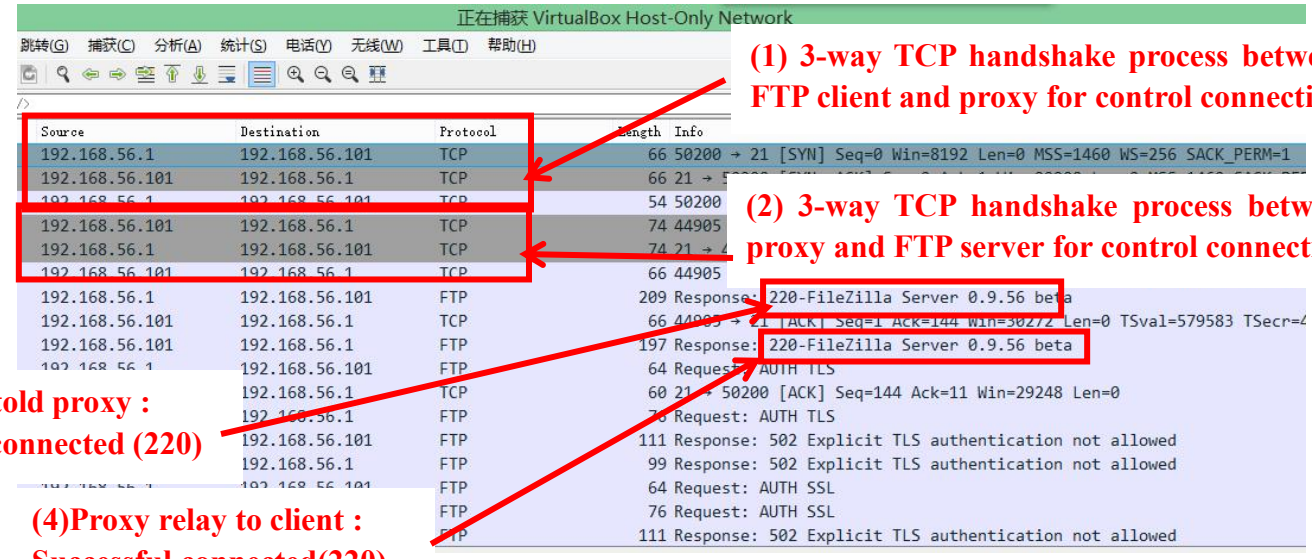
5. Results

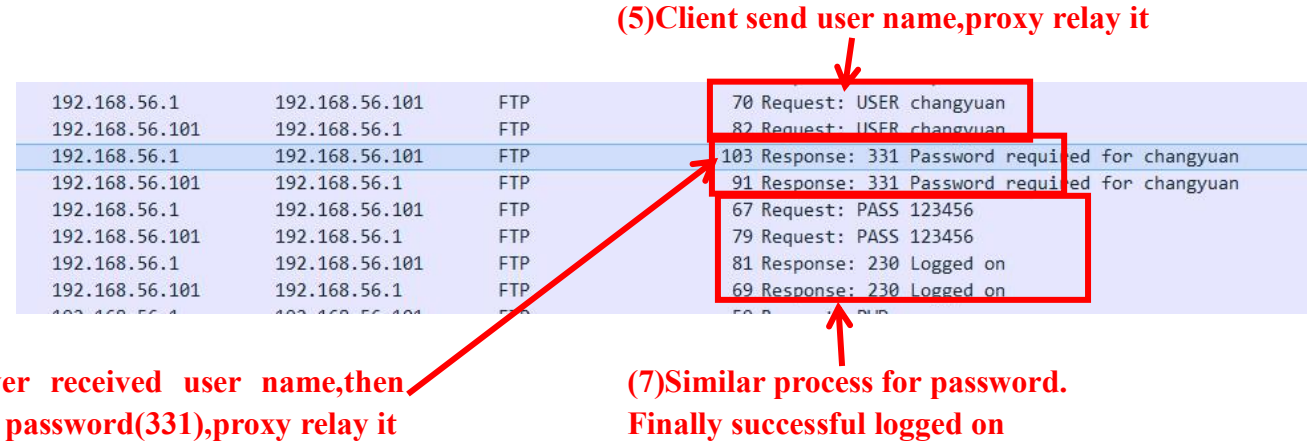
5.1 establish control connection

open FileZilla client and proxy program. Type in the host address 192.168.56.101 and user name and password. Successful connected.



Captured packets analysis:

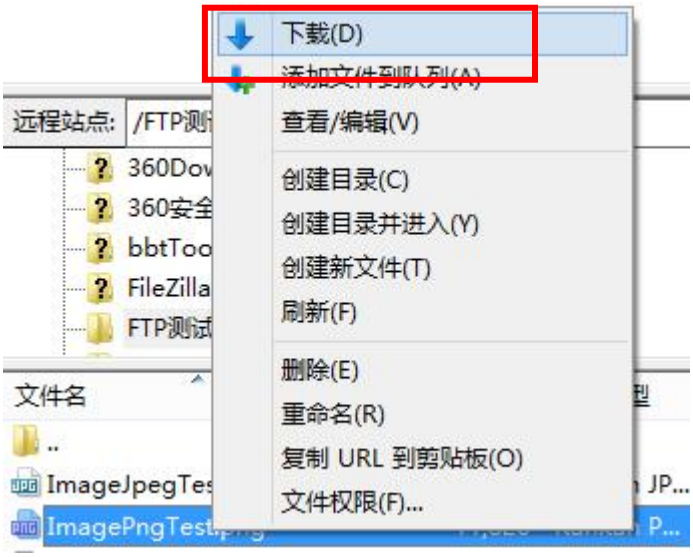




5.2 establish data connection

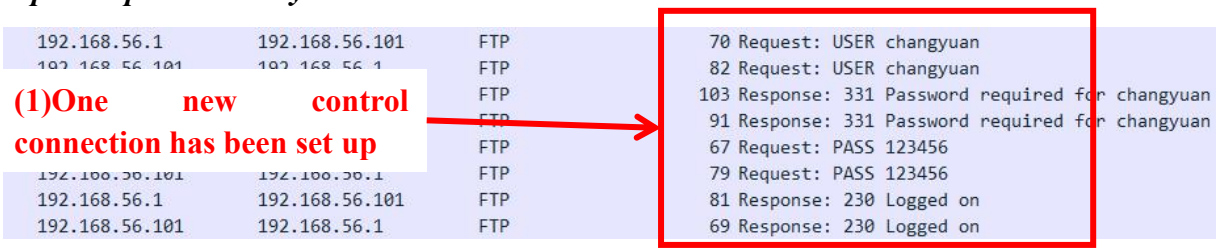
5.2.2 Example:download for the first time at passive mode

Choose one file and down load it from FTP server.

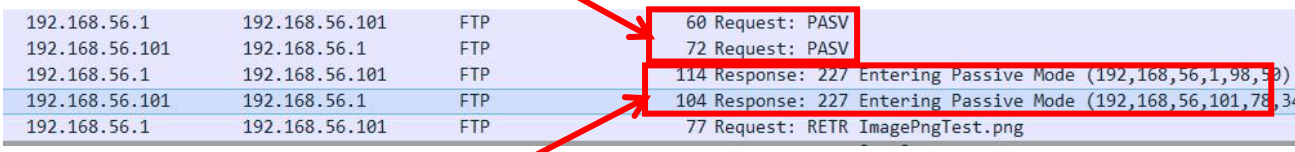


Successful download

captured packets analysis:



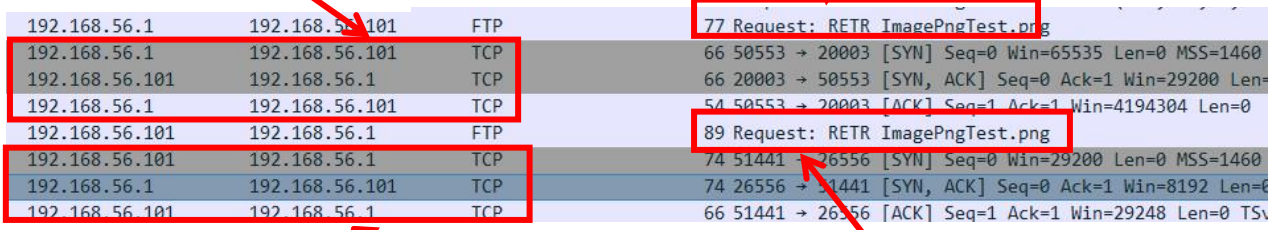
(2)FTP client send PASV command, proxy relay this command



(3)server send its 6 bytes PORT information, proxy modify this information into proxy PORT and send to client

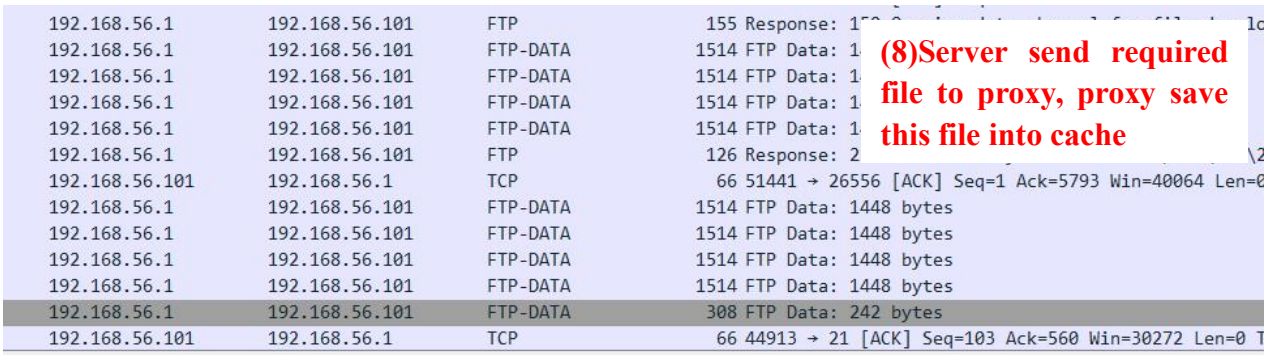
(4)Client send RETR command

(5)3-way TCP handshake between FTP client and proxy for data connection



(7)3-way TCP handshake between proxy and FTP server

(6)Proxy find this file is not exist in local cache, so relay RETR command to server



(8)Server send required file to proxy, proxy save this file into cache

192.168.56.101	192.168.56.1	FTP	203 Response: 150 Opening data channel for file download fro
192.168.56.101	192.168.56.1	TCP	66 51441 → 26556 [ACK] Seq=1 Ack=11828 Win=40832 Len=0 TSva
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data
192.168.56.1	192.168.56.101	TCP	54 50553 →
192.168.56.101	192.168.56.1	TCP	66 51441 →
192.168.56.1	192.168.56.101	TCP	66 26556 → 51441 [ACK] Seq=11828 Ack=2 Win=66560 Len=0 TSva
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes
192.168.56.1	192.168.56.101	TCP	54 50553 → 20003 [ACK] Seq=1 Ack=5841 Win=4194304 Len=0
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes
192.168.56.1	192.168.56.101	TCP	54 50553 → 20003 [ACK] Seq=1 Ack=8761 Win=4194304 Len=0
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes
192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes
192.168.56.1	192.168.56.101	TCP	54 50553 → 20003 [ACK] Seq=1 Ack=11681 Win=4194304 Len=0

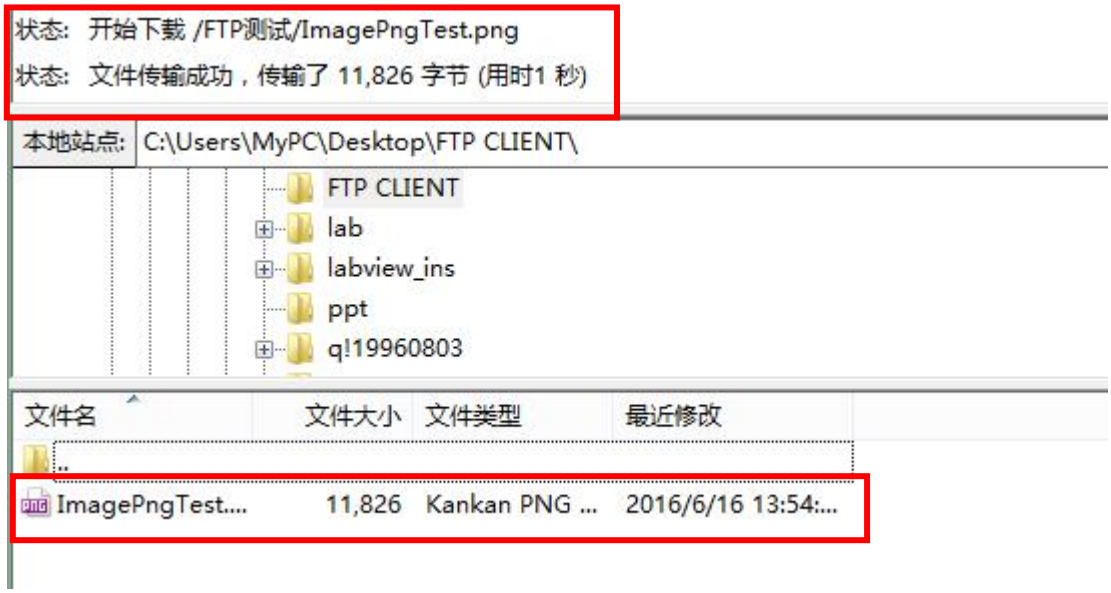
(9)Proxy relay the file to client

Check the cache, we can find transmitted file is already exist.

```
tudent/proxyFile# ls
ImagePngTest.png  main  main.c
tudent/proxyFile#
```

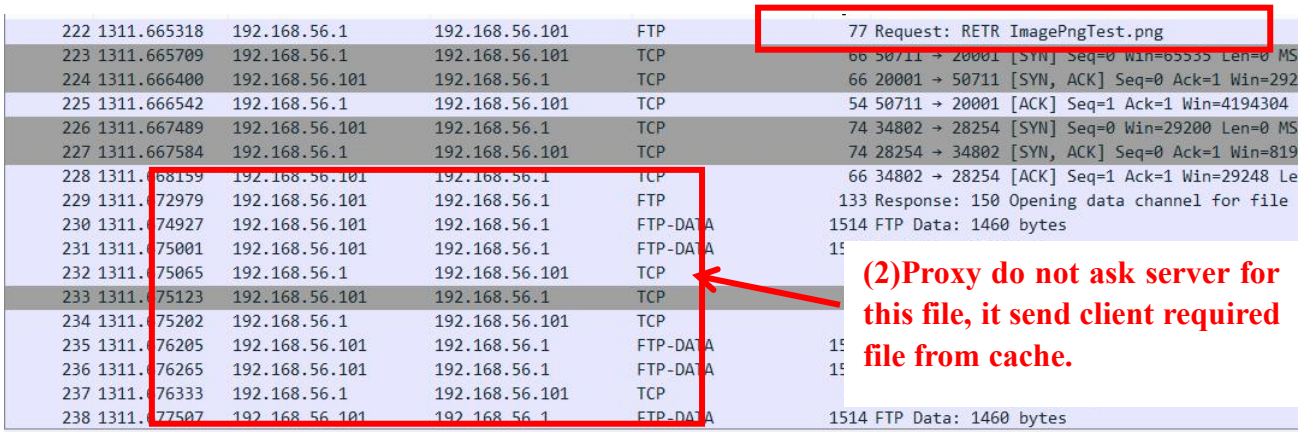
5.2.3 Example:download for the second time at passive mode

Delete the file in client side, and choose to download it again,then successful downloaded.



Captured packets analysis (similar process is omitted)

(1)Client send RETR command, proxy find file is already exist in cache, so proxy will not relay this command



222	1311.665318	192.168.56.1	192.168.56.101	FTP	77 Request: RETR ImagePngTest.png
223	1311.665709	192.168.56.1	192.168.56.101	TCP	66 50711 → 20001 [SYN] Seq=0 Win=65535 Len=0 MS
224	1311.666400	192.168.56.101	192.168.56.1	TCP	66 20001 → 50711 [SYN, ACK] Seq=0 Ack=1 Win=292
225	1311.666542	192.168.56.1	192.168.56.101	TCP	54 50711 → 20001 [ACK] Seq=1 Ack=1 Win=4194304
226	1311.667489	192.168.56.101	192.168.56.1	TCP	74 34802 → 28254 [SYN] Seq=0 Win=29200 Len=0 MS
227	1311.667584	192.168.56.1	192.168.56.101	TCP	74 28254 → 34802 [SYN, ACK] Seq=0 Ack=1 Win=819
228	1311.668159	192.168.56.101	192.168.56.1	TCP	66 34802 → 28254 [ACK] Seq=1 Ack=1 Win=29248 Le
229	1311.672979	192.168.56.101	192.168.56.1	FTP	133 Response: 150 Opening data channel for file
230	1311.674927	192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes
231	1311.675001	192.168.56.101	192.168.56.1	FTP-DATA	15
232	1311.675065	192.168.56.1	192.168.56.101	TCP	
233	1311.675123	192.168.56.101	192.168.56.1	TCP	
234	1311.675202	192.168.56.1	192.168.56.101	TCP	
235	1311.676205	192.168.56.101	192.168.56.1	FTP-DATA	15
236	1311.676265	192.168.56.101	192.168.56.1	FTP-DATA	15
237	1311.676333	192.168.56.1	192.168.56.101	TCP	
238	1311.677507	192.168.56.101	192.168.56.1	FTP-DATA	1514 FTP Data: 1460 bytes

(2)Proxy do not ask server for this file, it send client required file from cache.