Server – Client Architecture in UDP

Content

General Information

File Division

The project is mainly executed by 2 files: **server.py** and **client.py**. Both files are saved in 2 different directories, so that the each directory will contain uploaded/to-upload or downloaded files

Server Path

In server.py file there is a variable called "**serverDirectory**", which contains the directory of the server. This variable has to be modified by users, giving that different users will have different directory paths. The source code contains this path:

"/Users/stivengjinaj/PycharmProjects/Server-client/venv/server"

The OS where the program is executed is **MacOs**. Different operating systems will have different directory indications.

Encoding

For sending signals/messages the program uses "**utf-8**" (Beneficial because it makes possible the usage of practically every character the user may need), while for data transfer the program encodes in **binary** format (read/write)

Buffers

The program uses buffers of size 4096

Libraries

Libraries used are 4: socket, time, os and sys. Socket is used for socket initialization and operations with sockets. Time is used for program pauses. Os is used for operations with files, paths ecc.. Sys is used for the program termination.

Program Execution

Commands

The commands are found on **client.py** file. There are 4 command for user to choose:

- **Download *file_name***
- **Upload *file_name***
- **List**
- **Exit**

Commands are not passed as argument when the **client.py** is run. They appear when we run **client.py** and every time a command is chosen, it will be transmitted to server and both server and client will communication according to the requests. The program won't quit with the termination of a command. The program terminates if user:

- **Terminates the program**
- **Chooses command "Exit"**
- **Enters an unknown command**

Host and Port

Host and port are defined in both files **server.py** and **client.py**. The program uses **localhost** (127.0.0.1) and port **8080**. In order for the program to run both **server** and **client** must have the same values. Otherwise errors will be dealt by **try-except** blocks.

How to run

Firstly run the server.py

python server.py or **python3 server.py**

then run client.py

python server.py or **python3 server.py**

No arguments are needed.

Server

Main Part

In the main part **host** and **port** variables will be declared and the socket will be initialized (using **SOCKET.DGRAM** as attribute). Any connection error during **socket** creation will be dealt by **except**. Server will get the command from client and it will split it. Judging from the first position of the splitted string the **if/else** statements will call one of the functions below in order to accomplish the requests of the client

Functions

ServerList():

This function sends a message called **“VALID”** to the client. The client is waiting this confirmation message in order to start receiving the list of server files. Server then proceeds with appending all file names found in **path** to a list. The list will be converted into a string and the string will be encoded in <u>**utf-8**</u> format. The final string will be sent to client.

ServerExit():

When this function is called the socket will be closed and the program will terminate

ServerDownload(fileToGet):

This function sends a message called **“VALID”** to the client. When the client receives it, it will open a connection to receive data. Server will check which file the client is requesting, so it will check the existence of **fileToGet** in **path**. If file is found, server will notify client and it will calculate the file size and the number of chunks to send to client. In the end all chunks will be sent in binary format to the client and the file will be closed

ServerUpload():

This function sends a message called **“VALID”** to the client. If the command received from client is **“upload”** the process will start. Server will get a message of how much chunks will receive and then a file is created in server. The data coming from client will be sent to the file created in server in binary format.
Connection errors are dealt by **except.**

ServerElse():

This function terminates the program if an unknown command is received by client.

Client

Main Part

In the main part host and port variables will be declared and the socket will be initialized (using **SOCKET.DGRAM** as attribute). Any connection error during socket creation will be dealt by **except**. A **while** loop will begin and the menu of commands will appear. After user chooses the command, the command will be encoded and sent to server. Afterwards the command string will be splitted in order to capture the **download** or **upload** and the name of the file. 6 **if/else** statements will call the functions below based on the request.

Functions

listCommand():
The function waits for a message from server and when it receives it, it will check if the message is: **"VALID"**. If yes it will wait for a new message, which will be the string containing the list of files on server.
Connection errors are dealt by **except.**

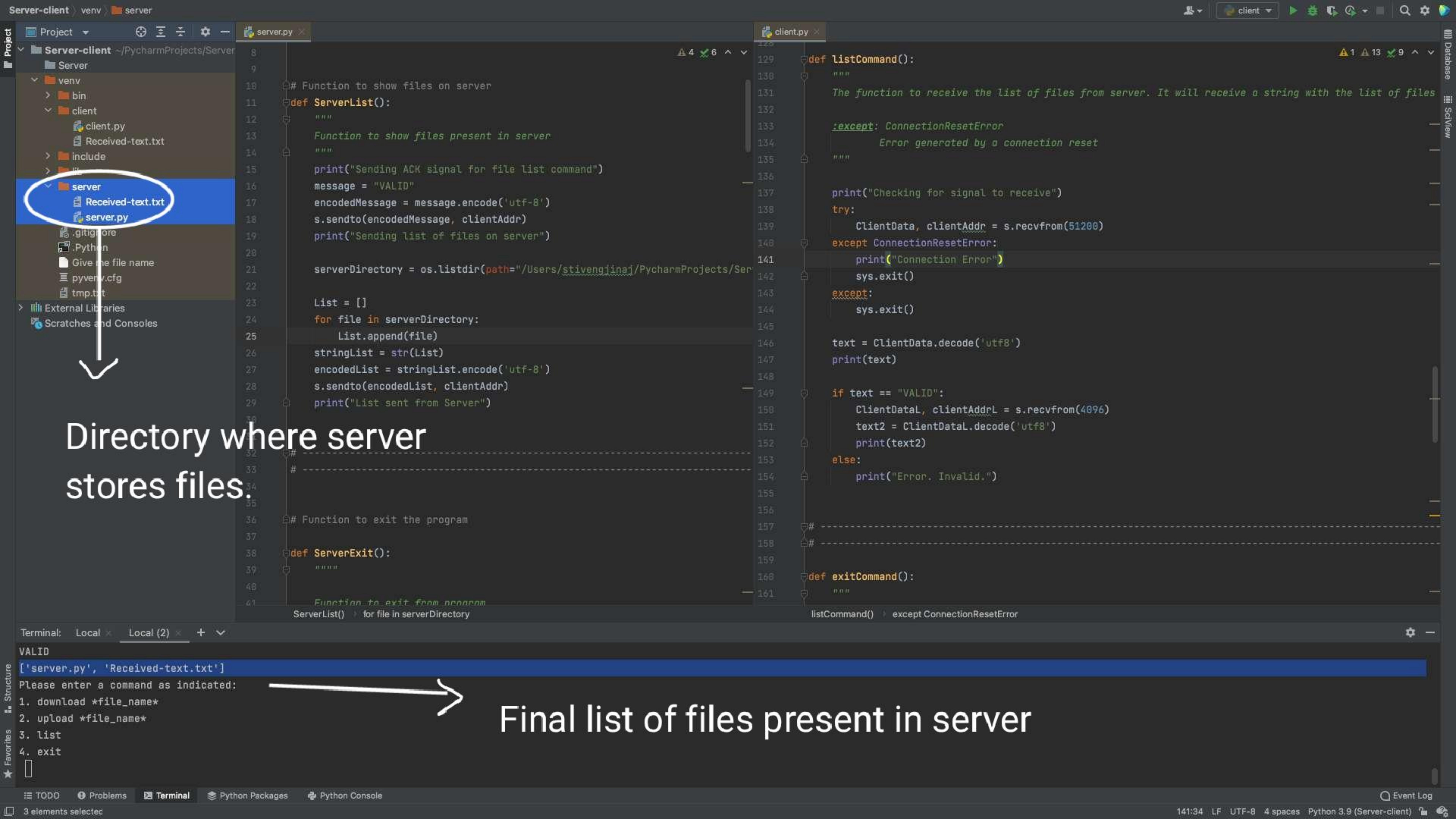
exitCommand():
The function terminates the client program because user chose to.

unknownCommand():
The function terminates the client program because of an unknown commandwas entered.

downloadCommand():
The function will wait for a signal **"VALID"** and if received it will wait for another message **"File exists."**. If the second message is received it will create a new file and the transfer of binary data will begin. In the end the file created will be closed.
Connection errors are dealt by **except.**

uploadCommand():
The function will wait for message **"VALID"** and if received it will calculate the size of the file to send and the chunks needed to be sent. In the end the files will be sent to server.
Connection errors are dealt by **except.**

Wireshark Analysis


Server-client 〉 venv 〉 server
Project
Server-client ~/PycharmProjects/Server
Server
venv
bin
client
client.py
Received-text.txt
include
server
Received-text.txt
server.py
.gitignore
.Python
Give me file name
pyvenv.cfg
tmp.txt
External Libraries
Scratches and Consoles
server.py
client.py
ServerList() 〉 for file in serverDirectory
listCommand() 〉 except ConnectionResetError
Terminal: Local Local (2)
TODO Problems Terminal Python Packages Python Console Event Log
3 elements selected
141:34 LF UTF-8 4 spaces Python 3.9 (Server-client)

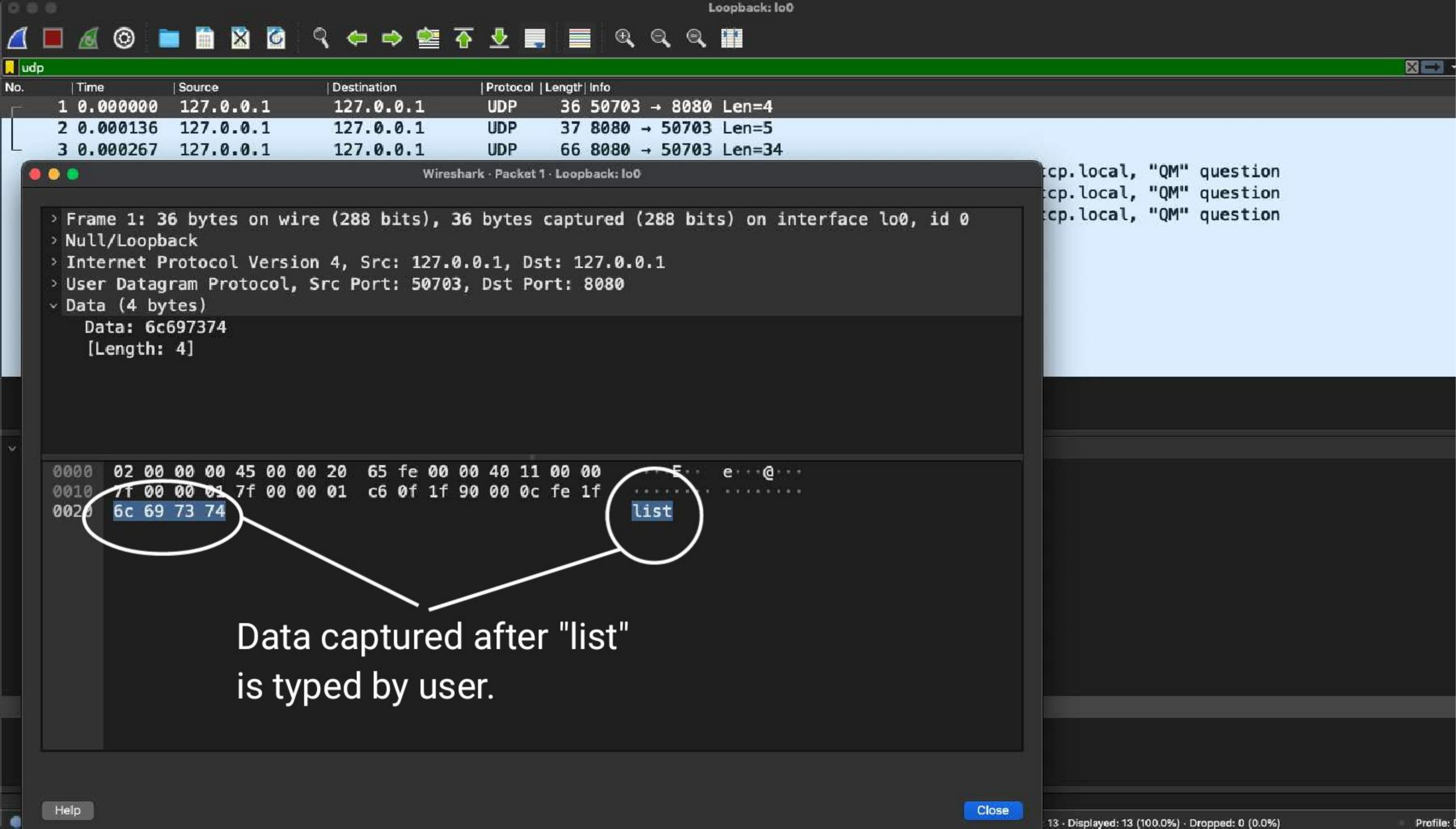
```python
# Function to show files on server
def ServerList():
    """
    Function to show files present in server
    """
    print("Sending ACK signal for file list command")
    message = "VALID"
    encodedMessage = message.encode('utf-8')
    s.sendto(encodedMessage, clientAddr)
    print("Sending list of files on server")

    serverDirectory = os.listdir(path="/Users/stivengjinaj/PycharmProjects/Ser

    List = []
    for file in serverDirectory:
        List.append(file)
    stringList = str(List)
    encodedList = stringList.encode('utf-8')
    s.sendto(encodedList, clientAddr)
    print("List sent from Server")

# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------


# Function to exit the program

def ServerExit():
    """"

    Function to exit from program
```

```python
def listCommand():
    """
    The function to receive the list of files from server. It will receive a string with the list of files

    :except: ConnectionResetError
            Error generated by a connection reset
    """

    print("Checking for signal to receive")
    try:
        ClientData, clientAddr = s.recvfrom(51200)
    except ConnectionResetError:
        print("Connection Error")
        sys.exit()
    except:
        sys.exit()

    text = ClientData.decode('utf8')
    print(text)

    if text == "VALID":
        ClientDataL, clientAddrL = s.recvfrom(4096)
        text2 = ClientDataL.decode('utf8')
        print(text2)
    else:
        print("Error. Invalid.")


# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------


def exitCommand():
    """
```

```
VALID
['server.py', 'Received-text.txt']
Please enter a command as indicated:
1. download *file_name*
2. upload *file_name*
3. list
4. exit
```

Directory where server stores files.

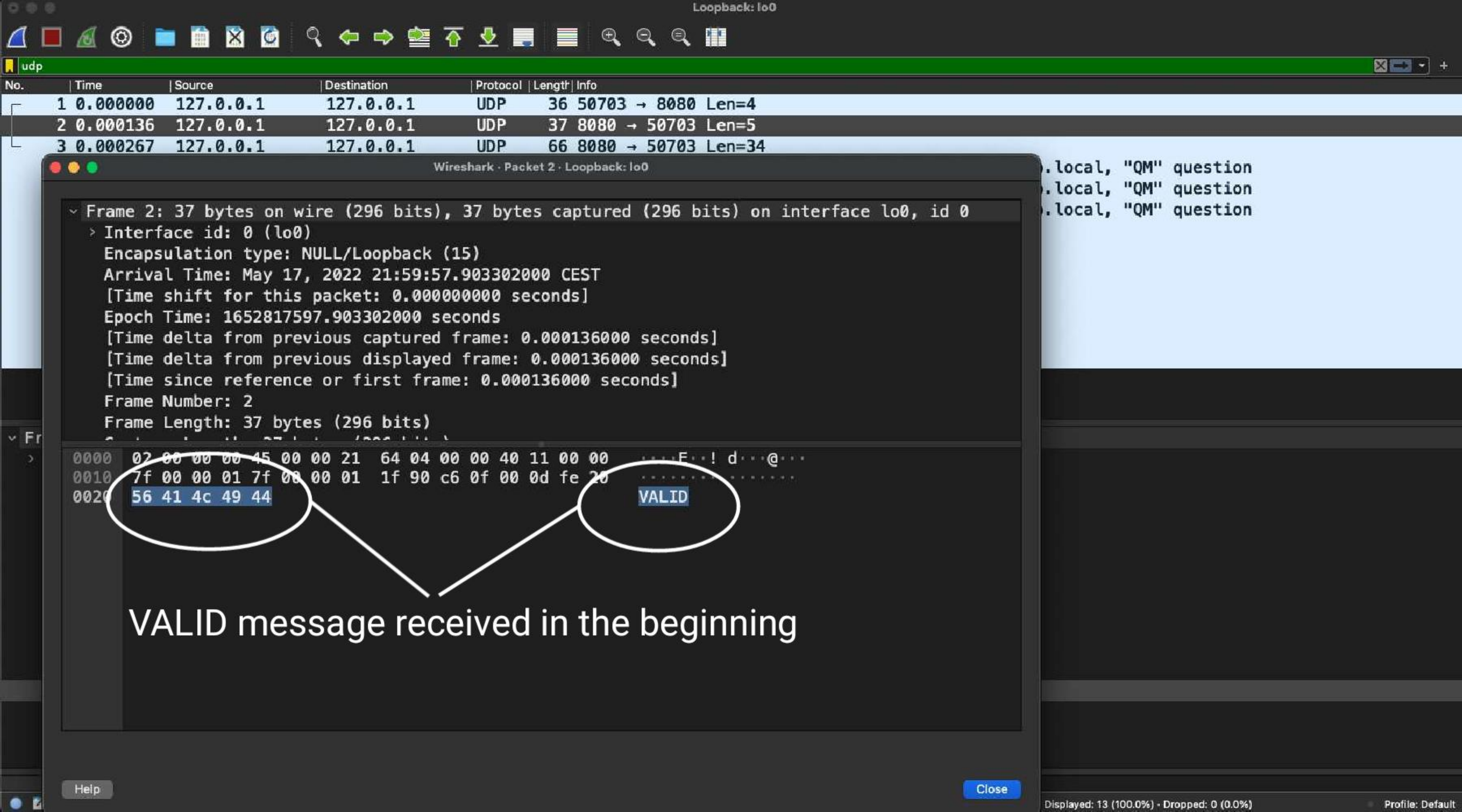
Final list of files present in server



```
Loopback: lo0
udp
No.  Time      Source     Destination  Protocol Length Info
1 0.000000  127.0.0.1  127.0.0.1    UDP      36 50703 → 8080 Len=4
2 0.000136  127.0.0.1  127.0.0.1    UDP      37 8080 → 50703 Len=5
3 0.000267  127.0.0.1  127.0.0.1    UDP      66 8080 → 50703 Len=34
cp.local, "QM" question
cp.local, "QM" question
cp.local, "QM" question
```

```
Wireshark · Packet 1 · Loopback: lo0
> Frame 1: 36 bytes on wire (288 bits), 36 bytes captured (288 bits) on interface lo0, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 50703, Dst Port: 8080
v Data (4 bytes)
    Data: 6c697374
    [Length: 4]

0000  02 00 00 00 45 00 00 20  65 fe 00 00 40 11 00 00   ····E··  e···@···
0010  7f 00 00 01 7f 00 00 01  c6 0f 1f 90 00 0c fe 1f   ········ ········
0020  6c 69 73 74                                        list
```

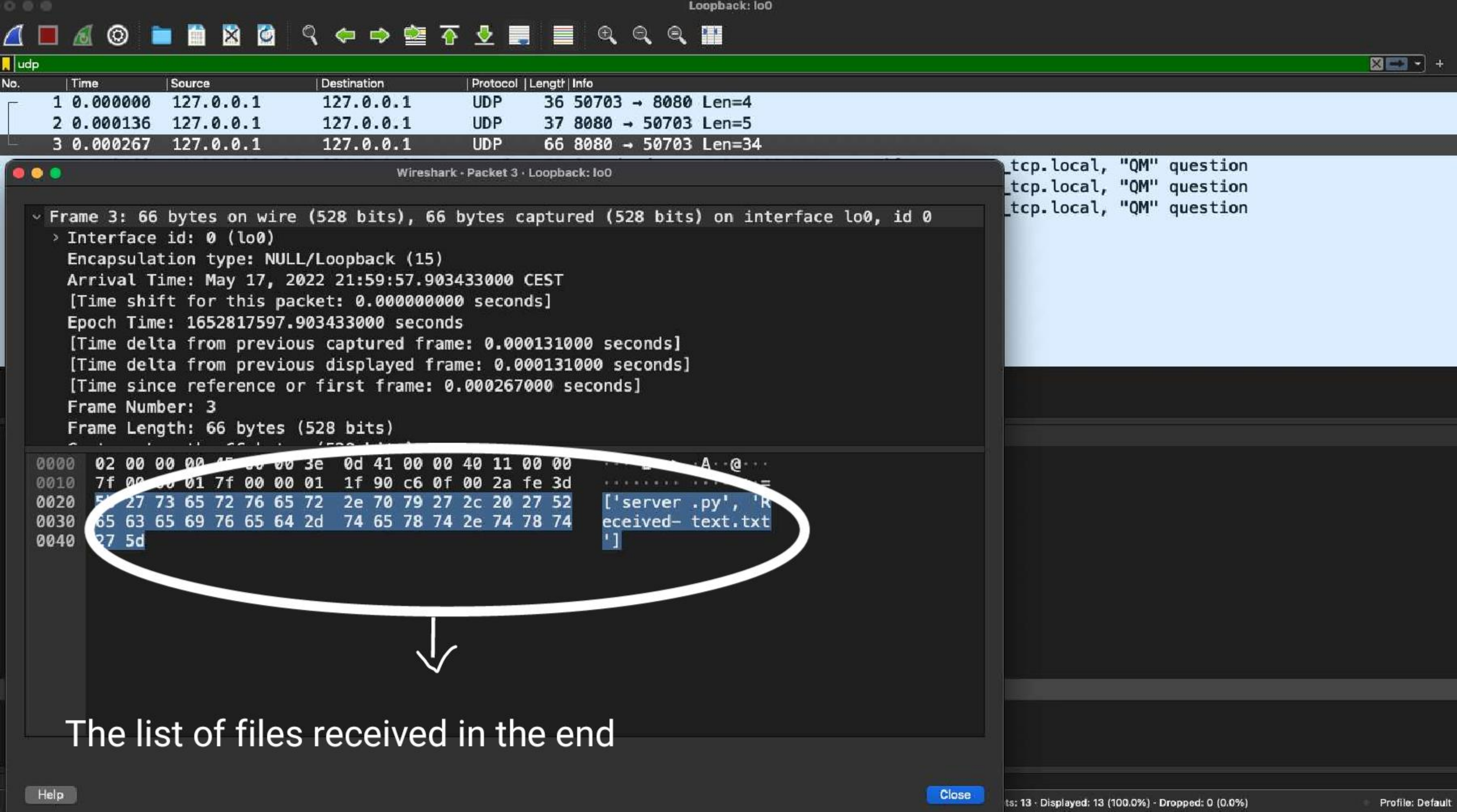
Data captured after "list" is typed by user.



```
Loopback: lo0
udp
No.  Time      Source     Destination  Protocol  Length  Info
  1 0.000000  127.0.0.1  127.0.0.1    UDP       36 50703 → 8080 Len=4
  2 0.000136  127.0.0.1  127.0.0.1    UDP       37 8080 → 50703 Len=5
  3 0.000267  127.0.0.1  127.0.0.1    UDP       66 8080 → 50703 Len=34
                                                   .local, "QM" question
                                                   .local, "QM" question
                                                   .local, "QM" question
```

```
Wireshark · Packet 2 · Loopback: lo0
⌄ Frame 2: 37 bytes on wire (296 bits), 37 bytes captured (296 bits) on interface lo0, id 0
  › Interface id: 0 (lo0)
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: May 17, 2022 21:59:57.903302000 CEST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1652817597.903302000 seconds
    [Time delta from previous captured frame: 0.000136000 seconds]
    [Time delta from previous displayed frame: 0.000136000 seconds]
    [Time since reference or first frame: 0.000136000 seconds]
    Frame Number: 2
    Frame Length: 37 bytes (296 bits)

0000  02 00 00 00 45 00 00 21  64 04 00 00 40 11 00 00   ····E··! d···@···
0010  7f 00 00 01 7f 00 00 01  1f 90 c6 0f 00 0d fe 20   ········ ········
0020  56 41 4c 49 44                                     VALID
```

VALID message received in the beginning



```
udp

No. | Time     | Source    | Destination | Protocol | Length | Info
  1 0.000000  127.0.0.1   127.0.0.1     UDP        36 50703 → 8080 Len=4
  2 0.000136  127.0.0.1   127.0.0.1     UDP        37 8080 → 50703 Len=5
  3 0.000267  127.0.0.1   127.0.0.1     UDP        66 8080 → 50703 Len=34
                                                   _tcp.local, "QM" question
                                                   _tcp.local, "QM" question
                                                   _tcp.local, "QM" question
```

```
⌄ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface lo0, id 0
  › Interface id: 0 (lo0)
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: May 17, 2022 21:59:57.903433000 CEST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1652817597.903433000 seconds
    [Time delta from previous captured frame: 0.000131000 seconds]
    [Time delta from previous displayed frame: 0.000131000 seconds]
    [Time since reference or first frame: 0.000267000 seconds]
    Frame Number: 3
    Frame Length: 66 bytes (528 bits)
```

```
0000  02 00 00 00 45 00 00 3e  0d 41 00 00 40 11 00 00   ····E··> ·A··@···
0010  7f 00 00 01 7f 00 00 01  1f 90 c6 0f 00 2a fe 3d   ········ ·····*·=
0020  5b 27 73 65 72 76 65 72  2e 70 79 27 2c 20 27 52   ['server .py', 'R
0030  65 63 65 69 76 65 64 2d  74 65 78 74 2e 74 78 74   eceived- text.txt
0040  27 5d                                              ']
```

The list of files received in the end

```
ts: 13 · Displayed: 13 (100.0%) · Dropped: 0 (0.0%)          Profile: Default
```