Computer Networks Lab

UE19CS256

Week 5

Name: Sreenath Saikumar

Semester: 4 Section: G

SRN: PES2UG19CS406

Date: 27/02/21

Objective:

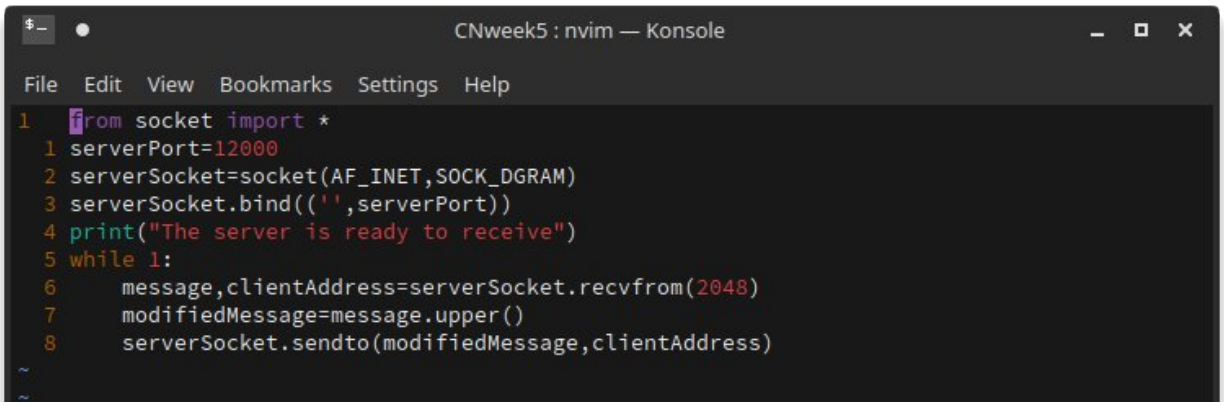
To develop a simple Client-Server application using TCP and UDP.

Task 1:

1) Create an application that will:
   a) Convert lowercase letters to uppercase
      i. e.g. [a..z] to [A..Z]
      ii. Code will not change any special characters, e.g. &*!
   b) If the character is in uppercase, the program must not alter.
2) Create Socket API for both client and server.
3) Must take server address and port from CLI.

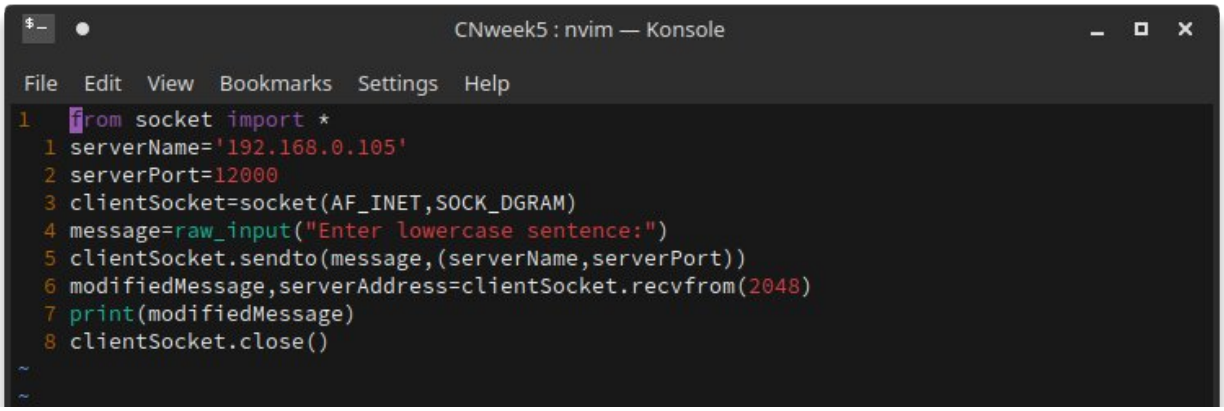
Socket Programming with UDP:

UDPServer.py

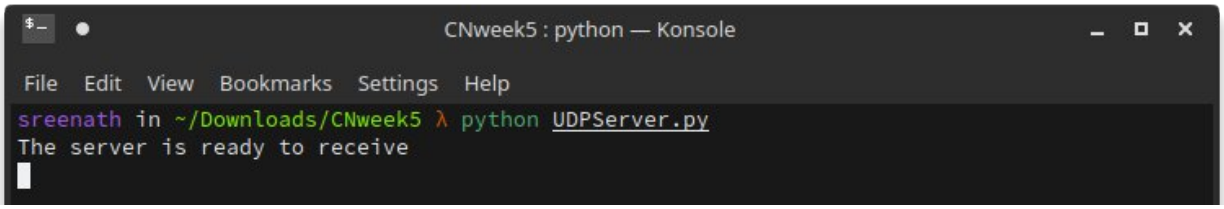



```
from socket import *
serverPort=12000
serverSocket=socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(('',serverPort))
print("The server is ready to receive")
while 1:
    message,clientAddress=serverSocket.recvfrom(2048)
    modifiedMessage=message.upper()
    serverSocket.sendto(modifiedMessage,clientAddress)
```

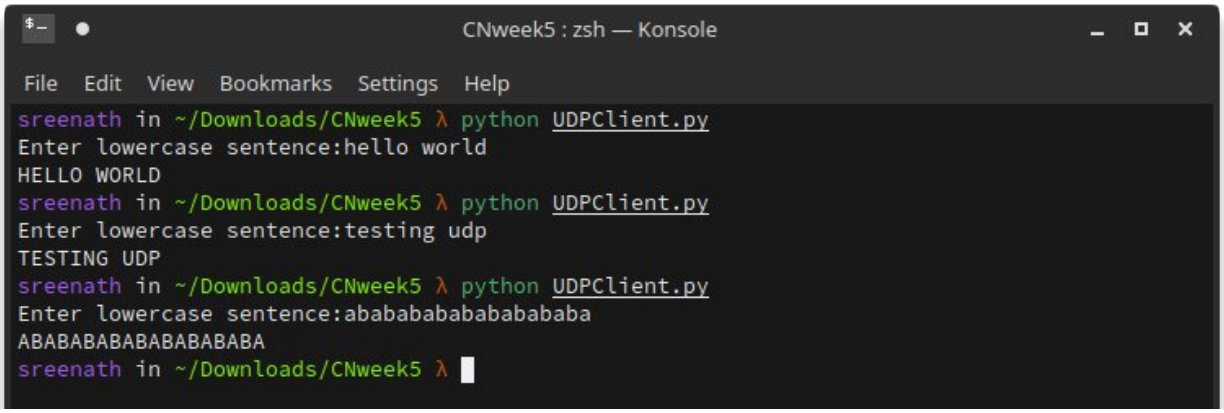
UDPClient.py



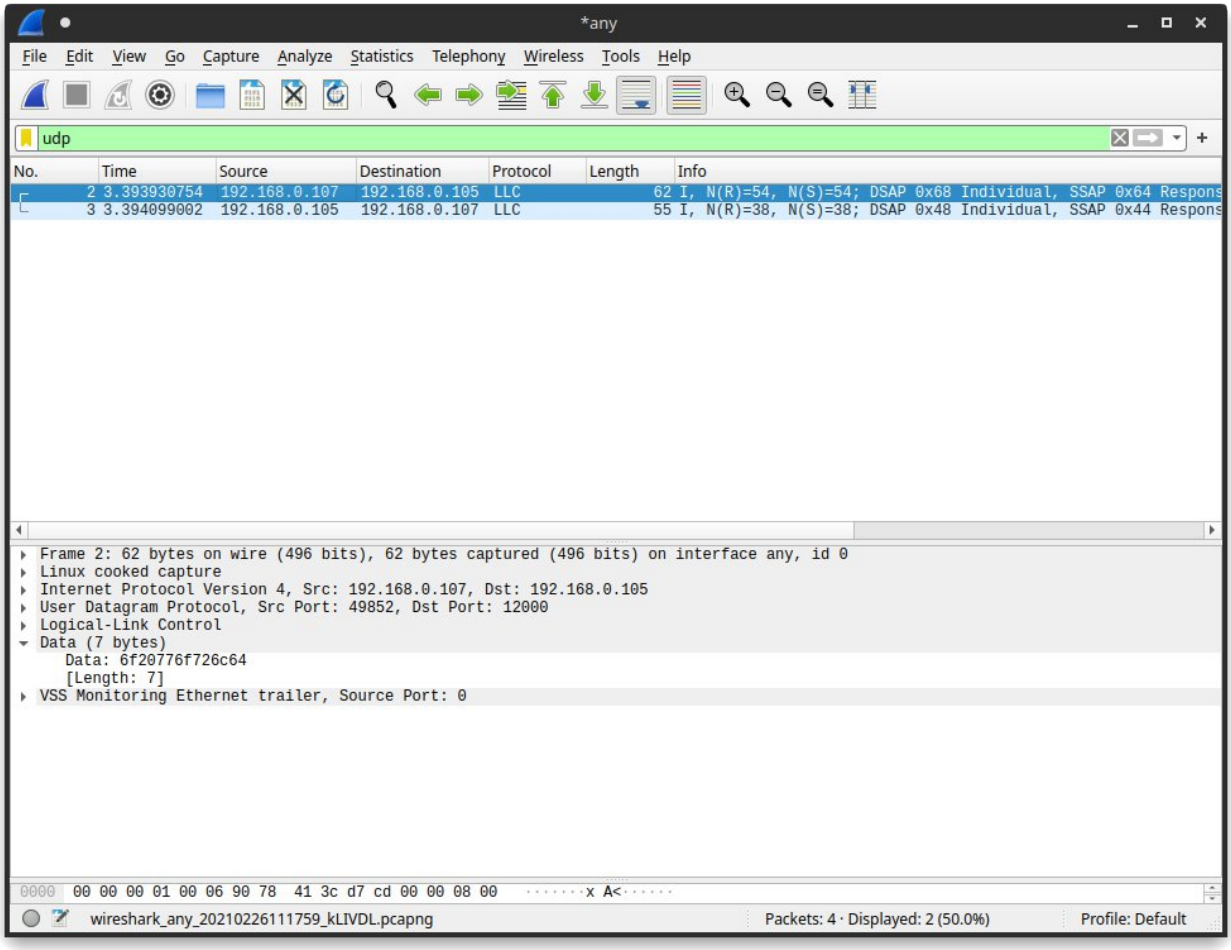
UDP Connection between Server and Client:



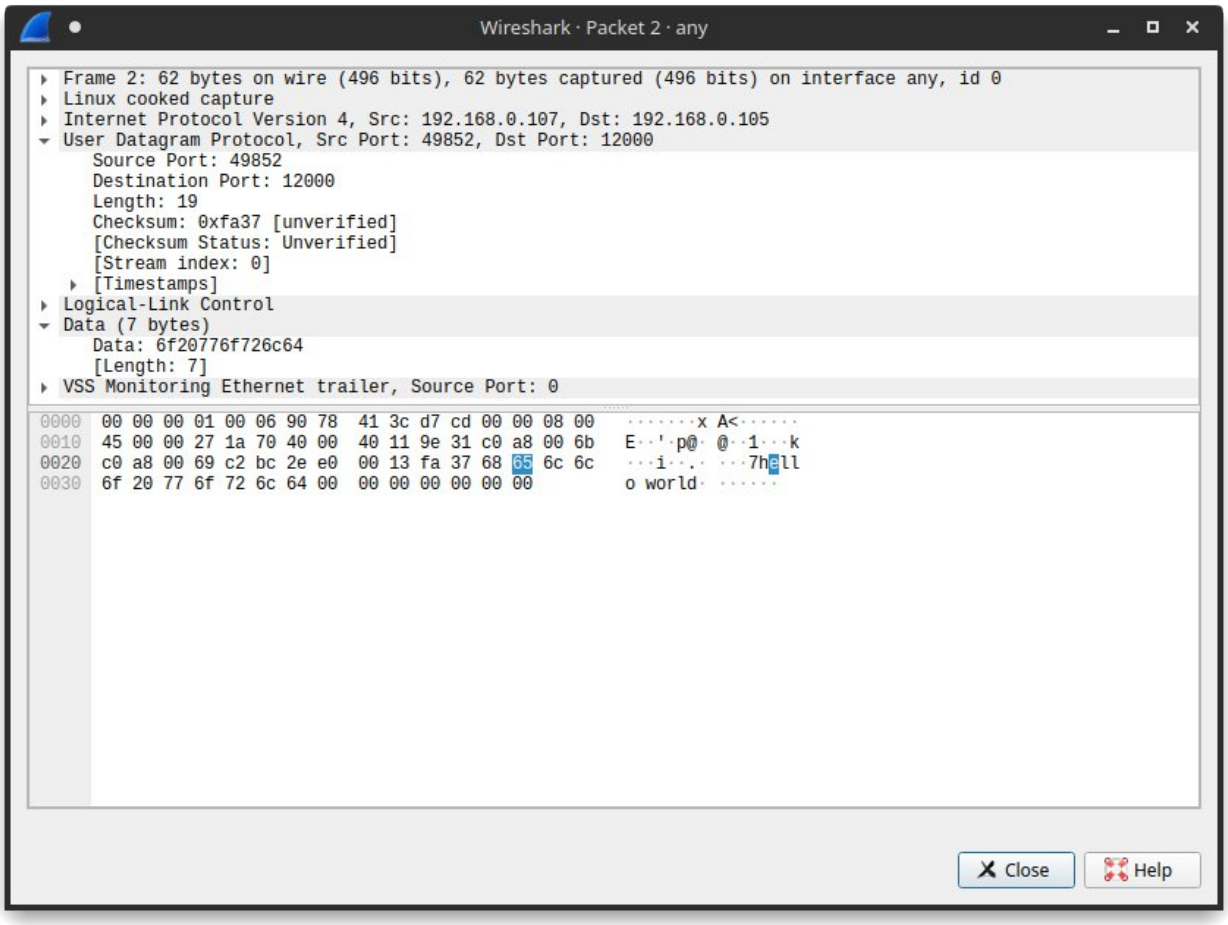
Running UDPServer.py in a terminal instance



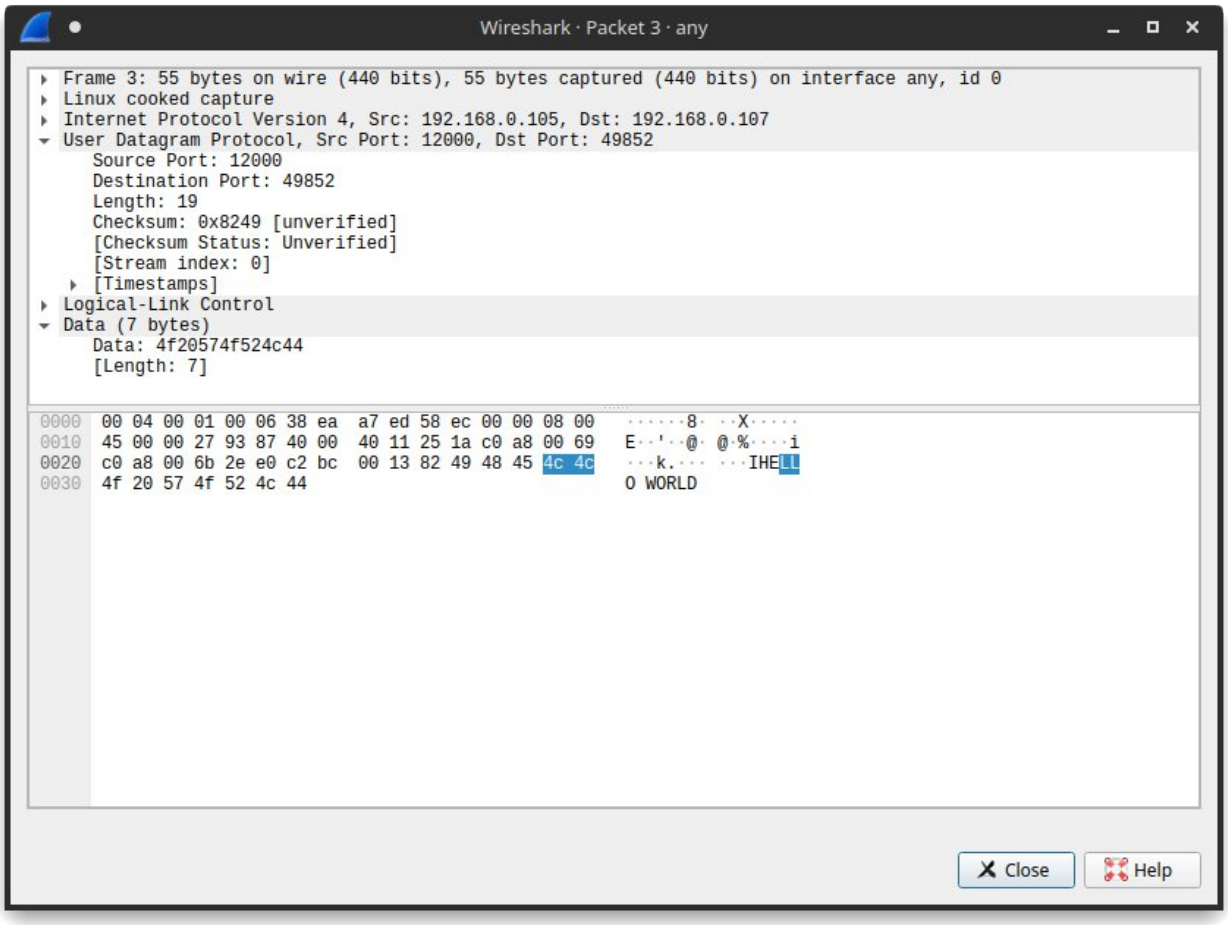
Running UDPClient.py in another terminal instance



Wireshark capture



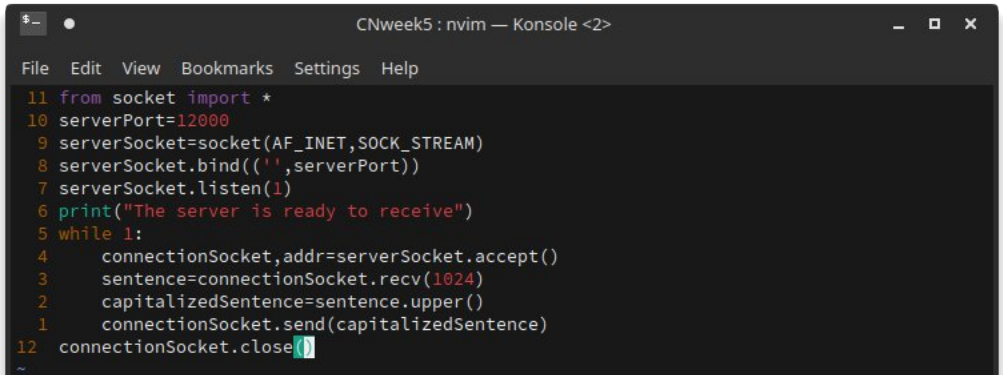
Client to Server



Server to Client

Socket Programming with TCP

TCPServer.py



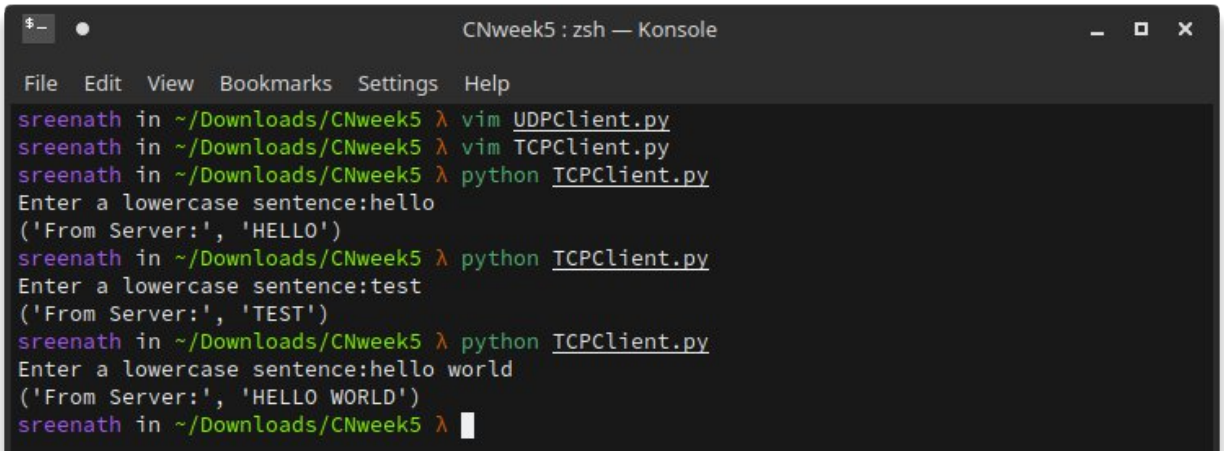
TCPClient.py

```
CNweek5 : nvim — Konsole
File  Edit  View  Bookmarks  Settings  Help
from socket import *
serverName='192.168.0.105'
serverPort=12000
clientSocket=socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence=raw_input("Enter a lowercase sentence:")
clientSocket.send(sentence)
modifiedSentence=clientSocket.recv(1024)
print("From Server:",modifiedSentence)
clientSocket.close()
~
```

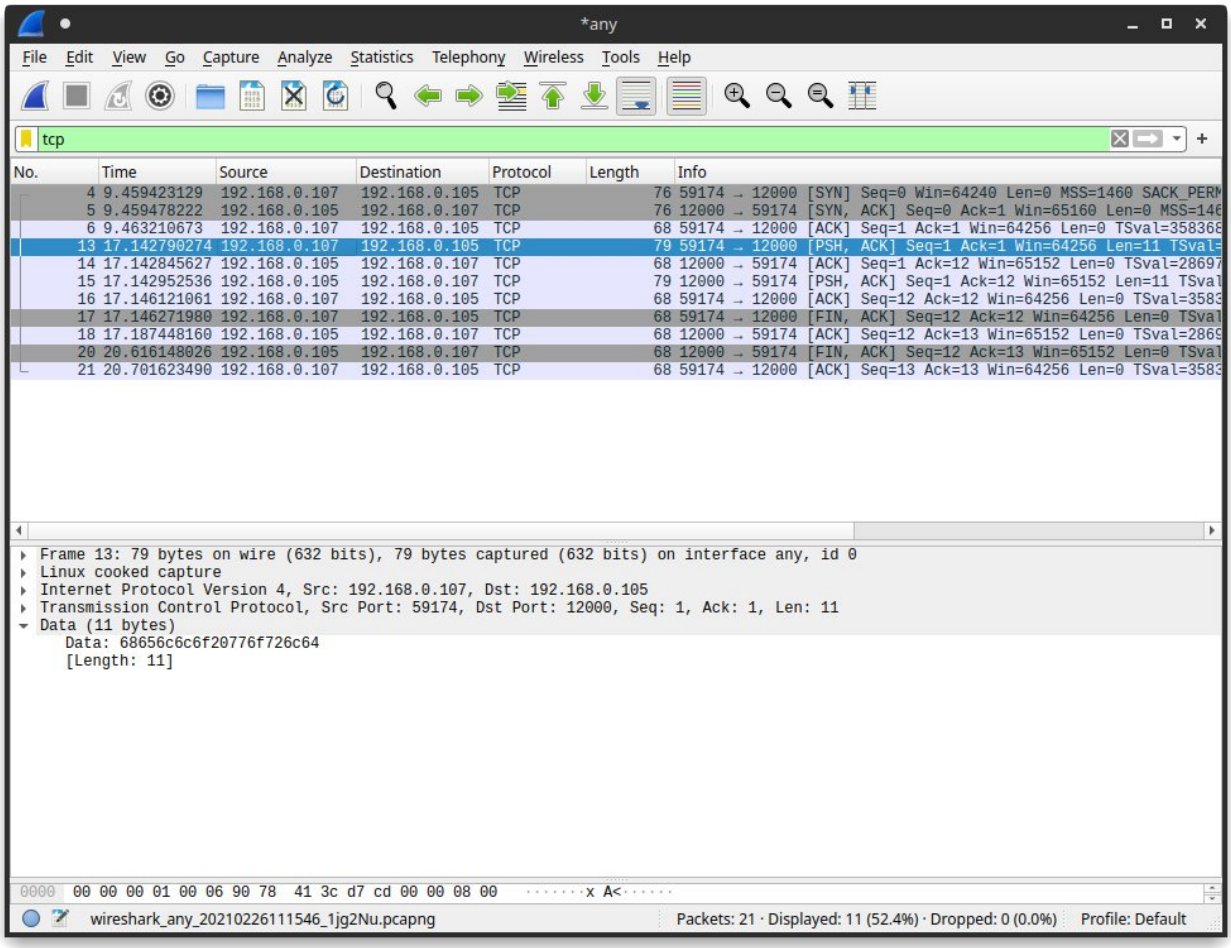
TCP Connection between Server and Client:

```
CNweek5 : python — Konsole
File  Edit  View  Bookmarks  Settings  Help
sreenath in ~/Downloads/CNweek5 λ python UDPServer.py
The server is ready to receive
^CTraceback (most recent call last):
  File "UDPServer.py", line 7, in <module>
    message,clientAddress=serverSocket.recvfrom(2048)
KeyboardInterrupt
sreenath in ~/Downloads/CNweek5 λ vim UDPServer.py
sreenath in ~/Downloads/CNweek5 λ vim TCPServer.py
sreenath in ~/Downloads/CNweek5 λ python TCPServer.py
The server is ready to receive
```

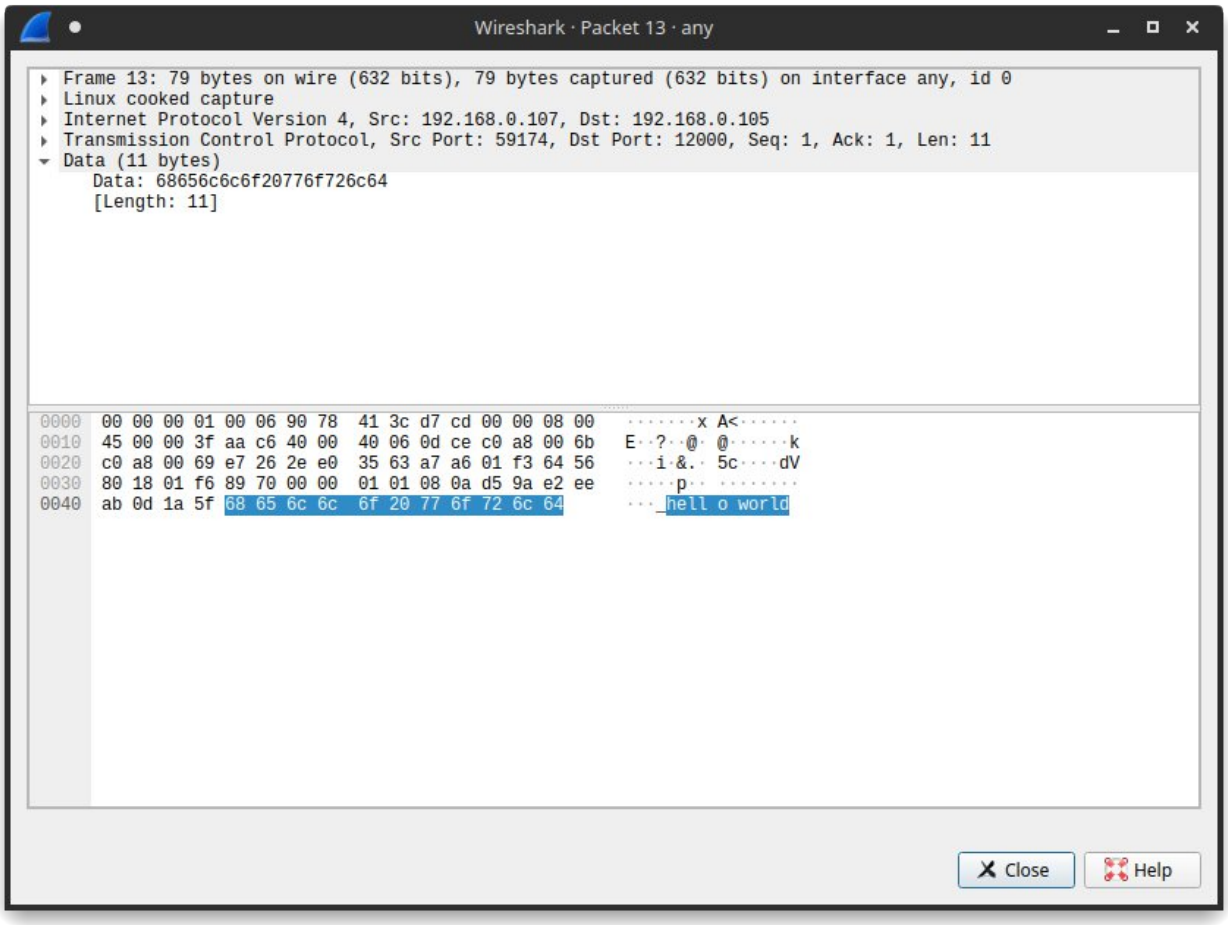
TCP Server running in one terminal instance



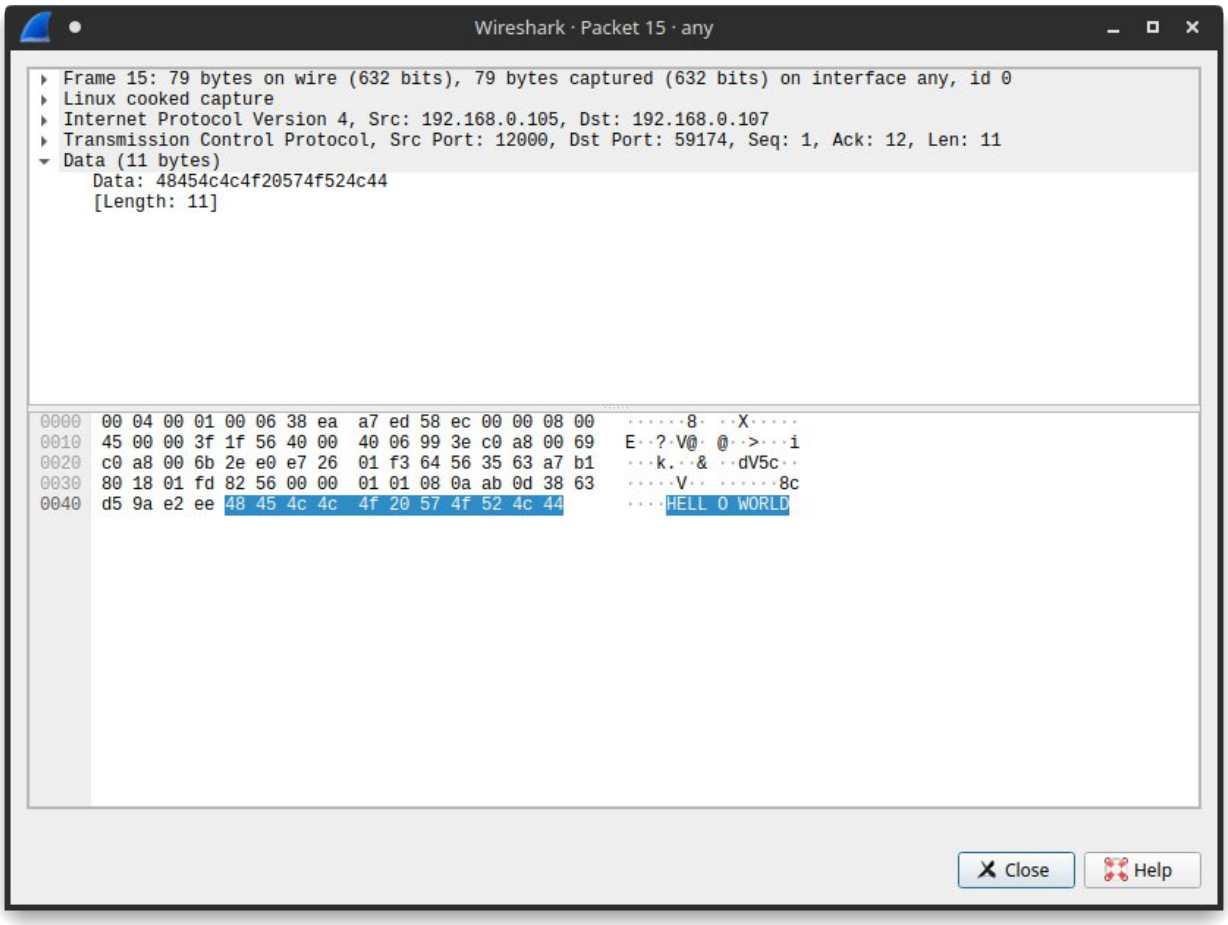
TCP Client in another terminal instance



Wireshark Capture



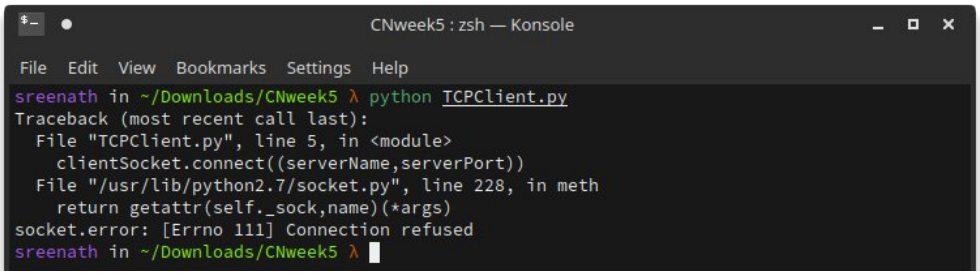
Client to Server



Server to Client

Problems:

1. Suppose you run TCPClient before you run TCPServer. What happens? Why?
    a. We get a `ConnectionRefusedError` since the server socket application hasn't been started and isn't listening on the given port number. Hence, any connection request sent from a client machine to the specified IP and port is immediately refused.



2. Suppose you run UDPClient before you run UDPServer. What happens? Why?
    a. Since UDP doesn't require any prior connection to be established, we will not get any errors as UDP is connectionless and transfers data to a destination IP and port without verifying whether the connection exists. If any packets of data are sent while the server isn't running, the packets are simply lost.
3. What happens if you use different port numbers for the client and server sides?
    a. We obtain a `ConnectionRefusedError` for a TCP connection since the server socket application isn't listening for requests on the same port number that the client is sending data to.

    However, on a UDP Connection, by virtue of it's connectionless behaviour, all data sent by the client is just lost since the destination doesn't exist.

Task 2: Web Server

In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will

a) create a connection socket when contacted by a client (browser);

b) receive the HTTP request from this connection;

c) parse the request to determine the specific file being requested;

d) get the requested file from the server's file system;

e) create an HTTP response message consisting of the requested file preceded by header lines; and

f) send the response over the TCP connection to the requesting browser. If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.

For this assignment, the companion Web site provides the skeleton code for your server. Your job is to complete the code, run your server, and then test your server by sending requests from browsers running on different hosts. If you run your server on a host that already has a Web server running on it, then you should use a different port than port 80 for your Web Server.

WebServer.py

CNweek5 : nvim — Konsole

File Edit View Bookmarks Settings Help

```
# Import socket module
from socket import *

# Create a TCP server socket
#(AF_INET is used for IPv4 protocols)
#(SOCK_STREAM is used for TCP)

serverSocket = socket(AF_INET, SOCK_STREAM)

# Assign a port number
serverPort = 6789

# Bind the socket to server address and server port
serverSocket.bind(("", serverPort))

# Listen to at most 1 connection at a time
serverSocket.listen(1)

# Server should be up and running and listening to the incoming connections
while True:
        print 'Ready to serve...'

        # Set up a new connection from the client
        connectionSocket, addr = serverSocket.accept()

        # If an exception occurs during the execution of try clause
        # the rest of the clause is skipped
        # If the exception type matches the word after except
```

NORMAL WebServer.py pyt… utf-8[dos] 46% ≡ 28/60 ln :8 ☰ [25]tra…

CNweek5 : nvim — Konsole

File Edit View Bookmarks Settings Help

```python
        try:
                # Receives the request message from the client
                message =  connectionSocket.recv(1024)
                # Extract the path of the requested object from the message
                # The path is the second part of HTTP header, identified by [1]
                filename = message.split()[1]
                # Because the extracted path of the HTTP request includes
                # a character '\', we read the path from the second character
                f = open(filename[1:])
                # Store the entire contenet of the requested file in a temporary buffer
                outputdata = f.read()
                # Send the HTTP response header line to the connection socket
                connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n")

                # Send the content of the requested file to the connection socket
                for i in range(0, len(outputdata)):
                        connectionSocket.send(outputdata[i])
                connectionSocket.send("\r\n")

                # Close the client connection socket
                connectionSocket.close()

        except IOError:
                # Send HTTP response message for file not found
                connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n")
                connectionSocket.send("<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n")
```

NORMAL WebServer.py pyt… utf-8[dos] 91% ≡ 55/60 ln :8 ☰ [25]tra…

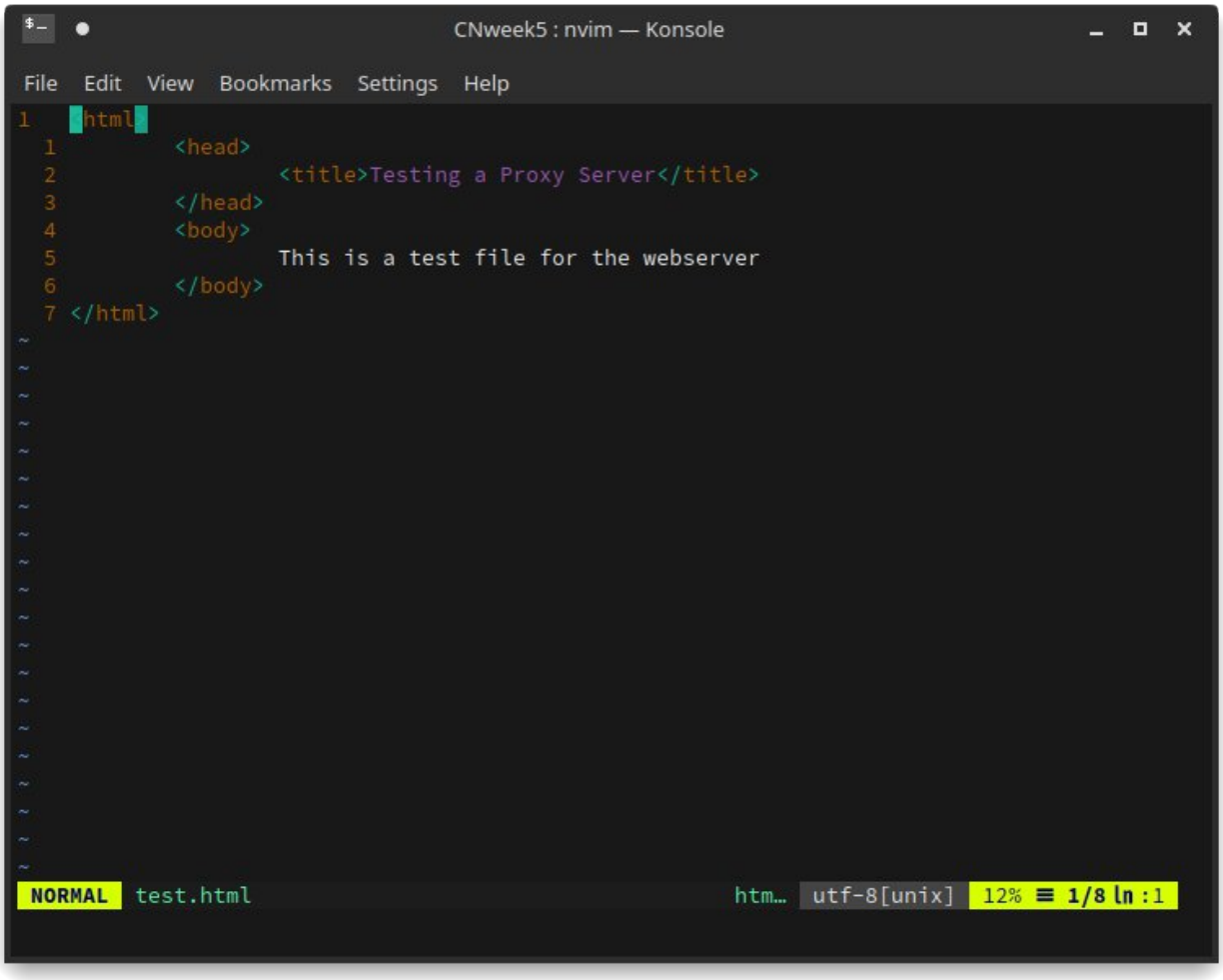
CNweek5 : nvim — Konsole

File Edit View Bookmarks Settings Help

```
                filename = message.split()[1]
                # Because the extracted path of the HTTP request includes
                # a character '\', we read the path from the second character
                f = open(filename[1:])
                # Store the entire contenet of the requested file in a temporary buffer
                outputdata = f.read()
                # Send the HTTP response header line to the connection socket
                connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n")

                # Send the content of the requested file to the connection socket
                for i in range(0, len(outputdata)):
                        connectionSocket.send(outputdata[i])
                connectionSocket.send("\r\n")

                # Close the client connection socket
                connectionSocket.close()

        except IOError:
                # Send HTTP response message for file not found
                connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n")
                connectionSocket.send("<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n")
                # Close the client connection socket
                connectionSocket.close()

serverSocket.close()
```

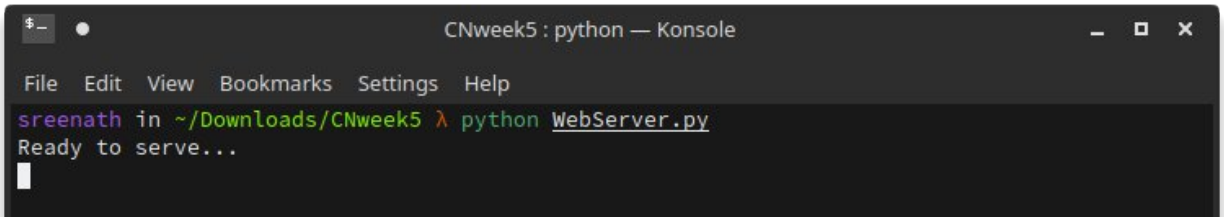
NORMAL WebServer.py pyt… utf-8[dos] 100% ≡ 60/60 ln :1 ≡ [25]tra…

Source Code for Web Server.

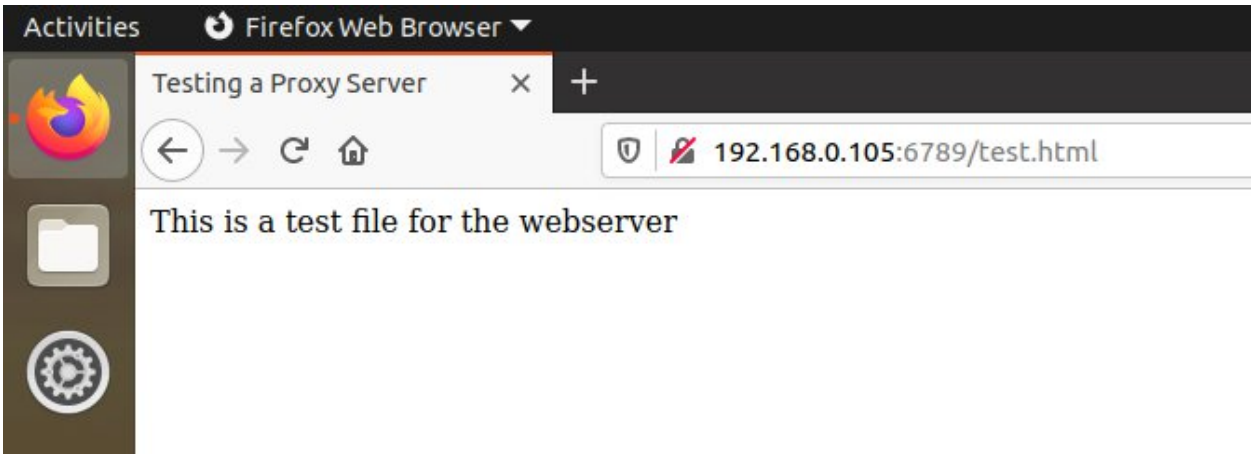


Test HTML file

Running the Web Server:

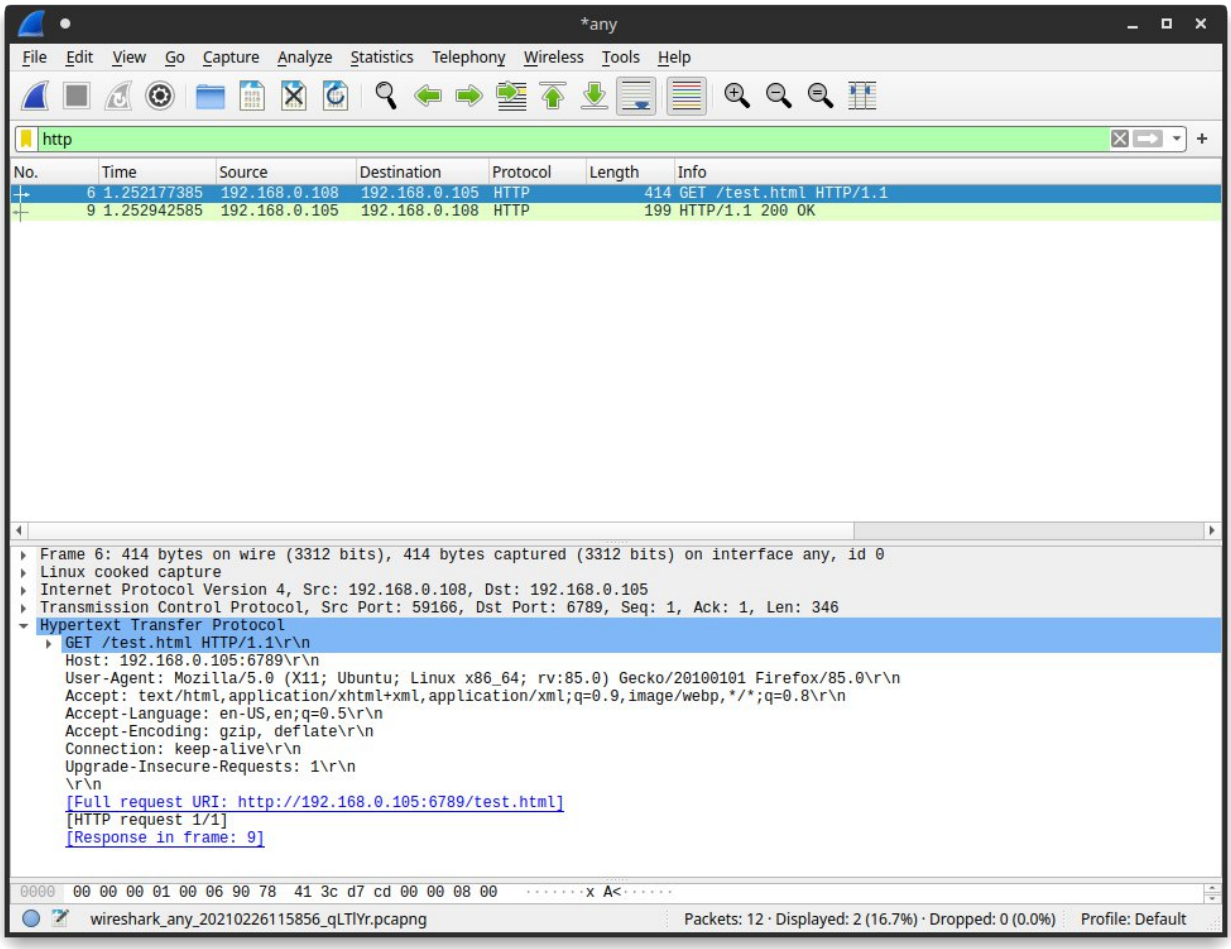


Server Side

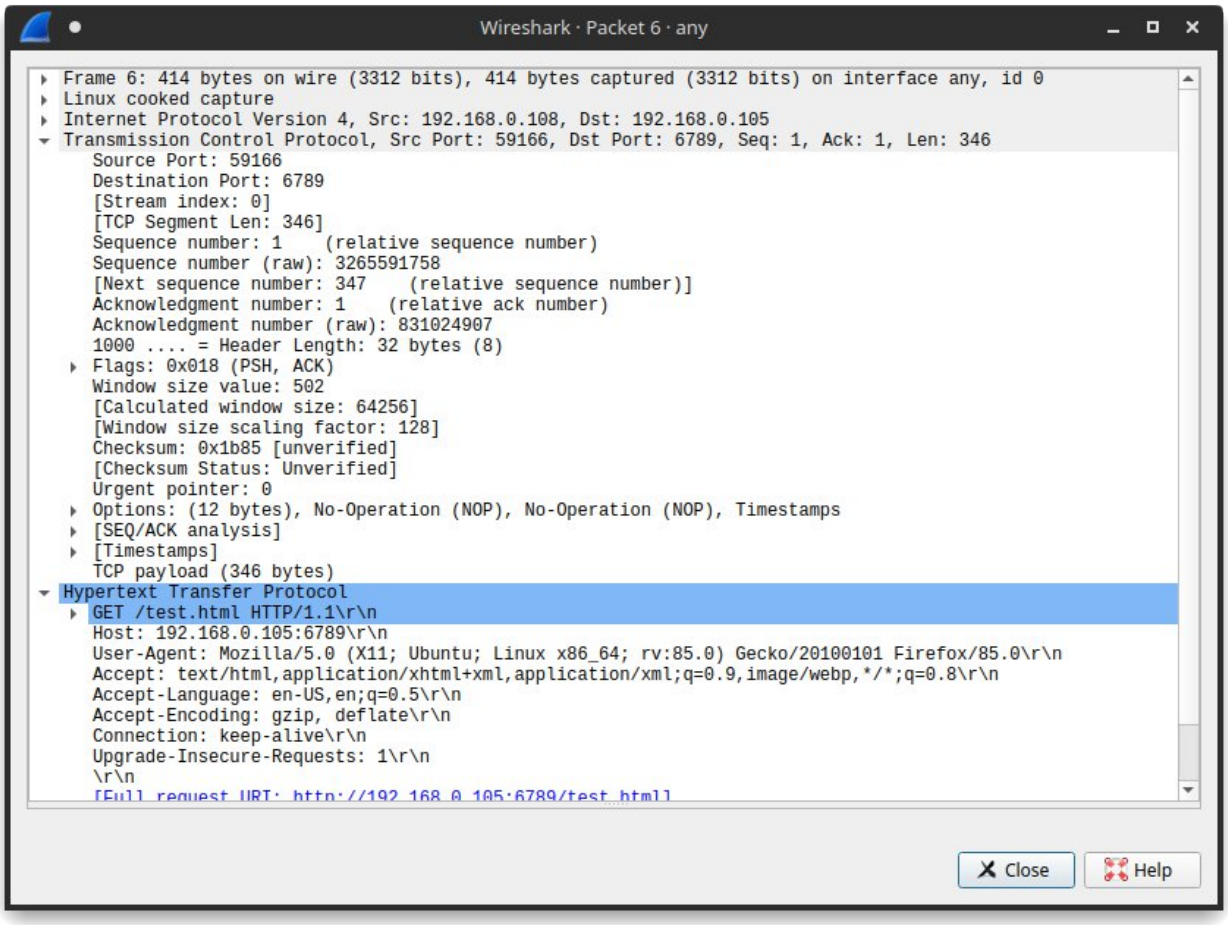


Accessing `test.html` from the client side

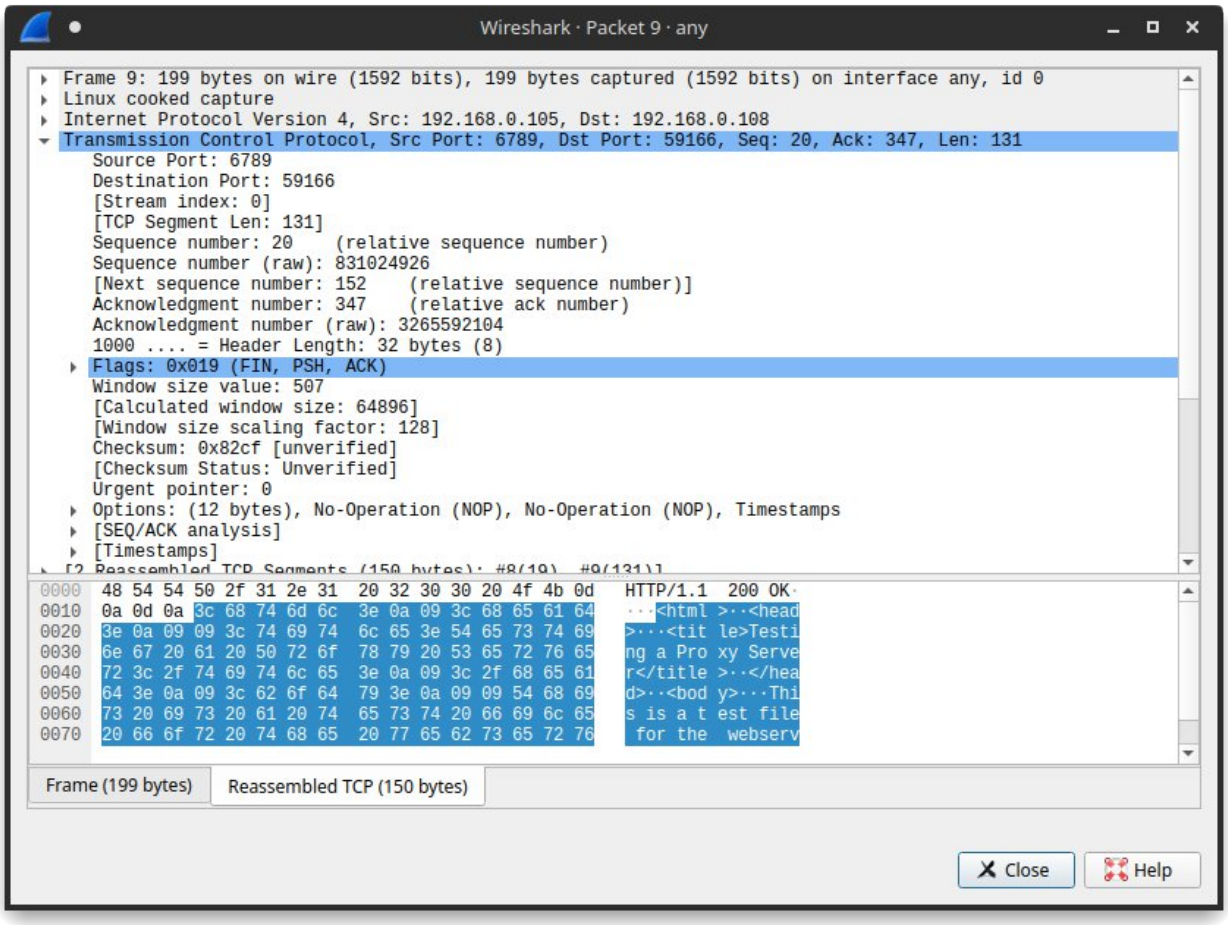
If the file doesn't exist on the server, we get a `404 Not found` error.



Wireshark capture



HTTP request packet



HTTP Response Packet