COMPUTER NETWORKS LABORATORY

By:
Nitish S
PES2201800368
5 ‘A’

WEEK – 5- Simple Client-Server Application using Network Socket Programming

Date: 21/10/2020

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

Pre requisites:
- Basic understanding of networking concepts and socket programming
- Knowledge of python

Sockets
Sockets are just the **endpoints of a two-way communication link** in a network. Socket helps in the communication of two processes/programs on a network (eg. Internet). The programs can communicate by reading/writing via their sockets. A socket comprises of: ***IP Address & Port number.***

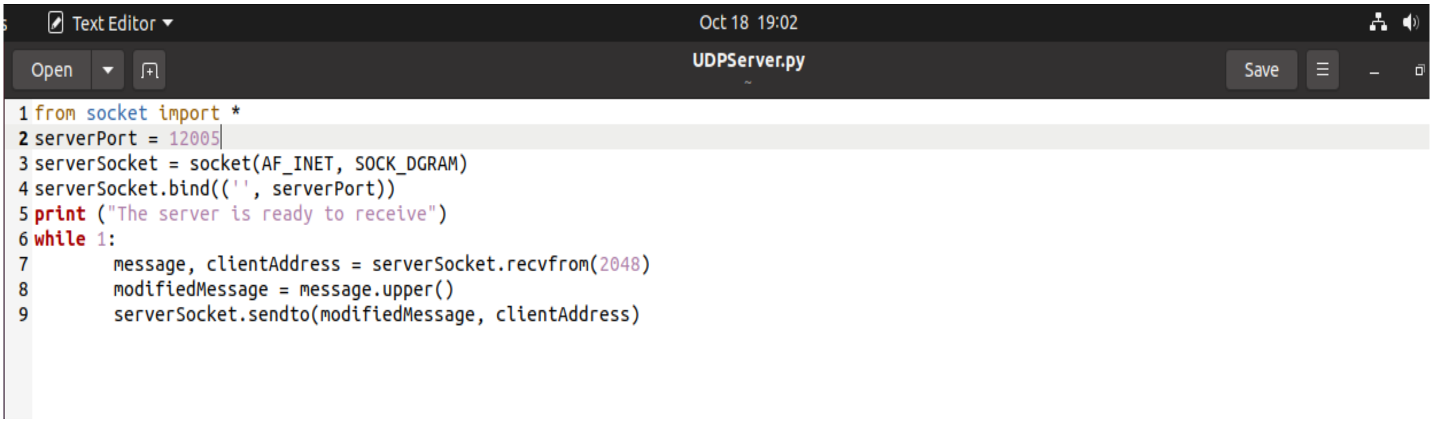
Task 1:

1. Create an application that will
a. Convert lowercase letters to uppercase
- e.g. [a…z] to [A…Z]
- 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 both for
3. Must take the server address and port from the CLI

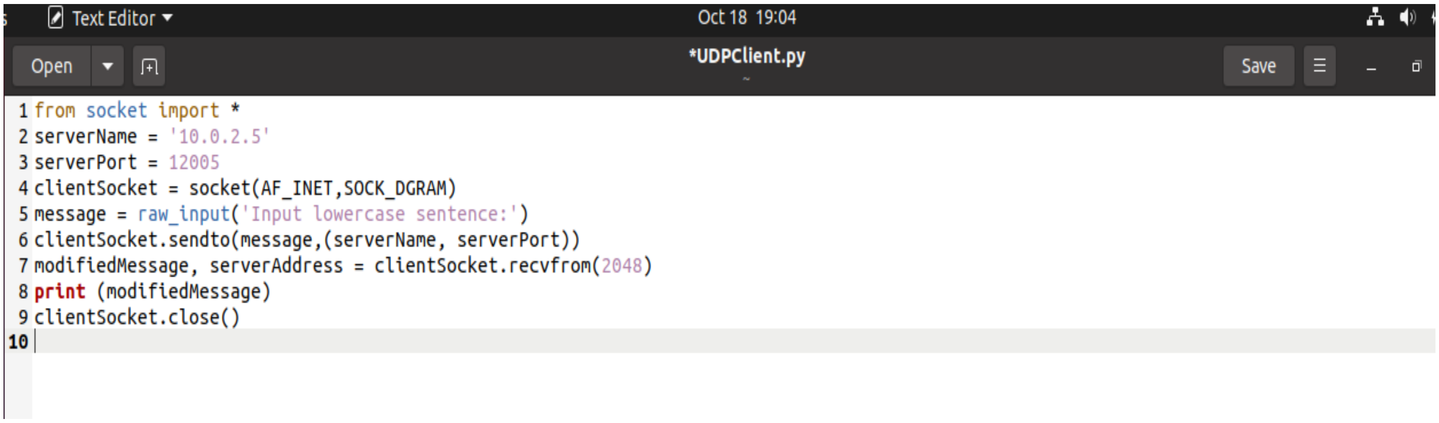
Socket Programming with UDP:-

UDPServer.py:



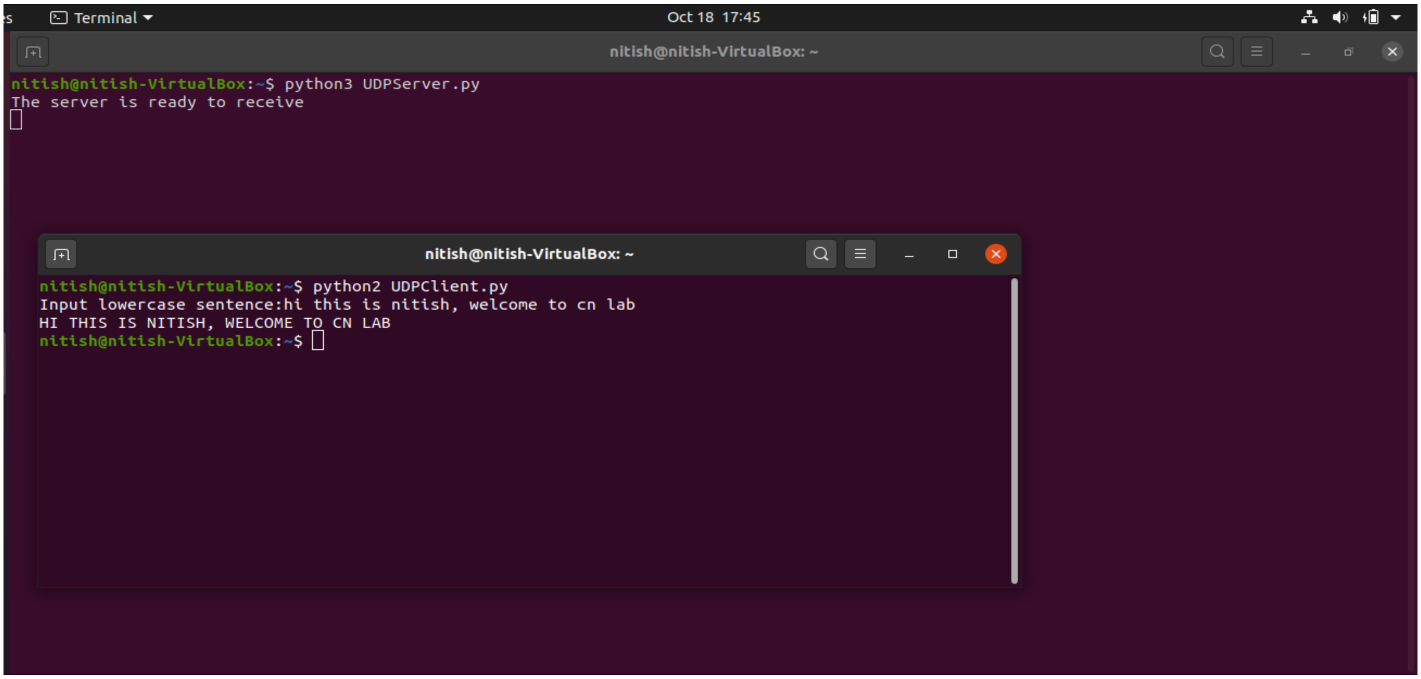
```
from socket import *
serverPort = 12005
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print ("The server is ready to receive")
while 1:
        message, clientAddress = serverSocket.recvfrom(2048)
        modifiedMessage = message.upper()
        serverSocket.sendto(modifiedMessage, clientAddress)
```

UDPClient.py:



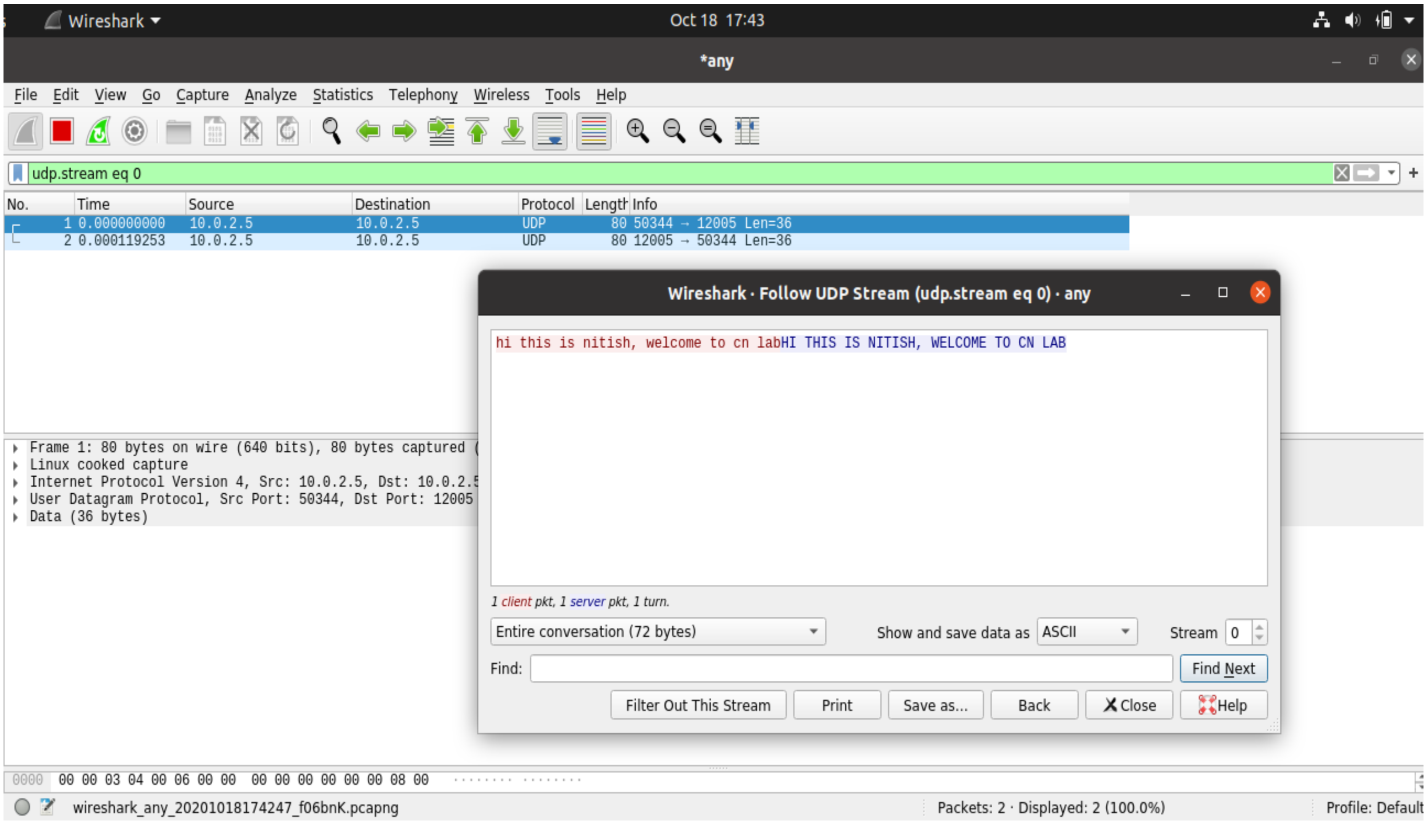
```
from socket import *
serverName = '10.0.2.5'
serverPort = 12005
clientSocket = socket(AF_INET,SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print (modifiedMessage)
clientSocket.close()

```

EXECUTION:



```
nitish@nitish-VirtualBox:~$ python3 UDPServer.py
The server is ready to receive
```

```
nitish@nitish-VirtualBox:~$ python2 UDPClient.py
Input lowercase sentence:hi this is nitish, welcome to cn lab
HI THIS IS NITISH, WELCOME TO CN LAB
nitish@nitish-VirtualBox:~$
```



Wireshark
Oct 18 17:43
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

udp.stream eq 0

```
No.   Time         Source     Destination  Protocol Length Info
    1 0.000000000  10.0.2.5   10.0.2.5     UDP          80 50344 → 12005 Len=36
    2 0.000119253  10.0.2.5   10.0.2.5     UDP          80 12005 → 50344 Len=36
```

```
▸ Frame 1: 80 bytes on wire (640 bits), 80 bytes captured (
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.5
▸ User Datagram Protocol, Src Port: 50344, Dst Port: 12005
▸ Data (36 bytes)
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
```

```
hi this is nitish, welcome to cn labHI THIS IS NITISH, WELCOME TO CN LAB
```

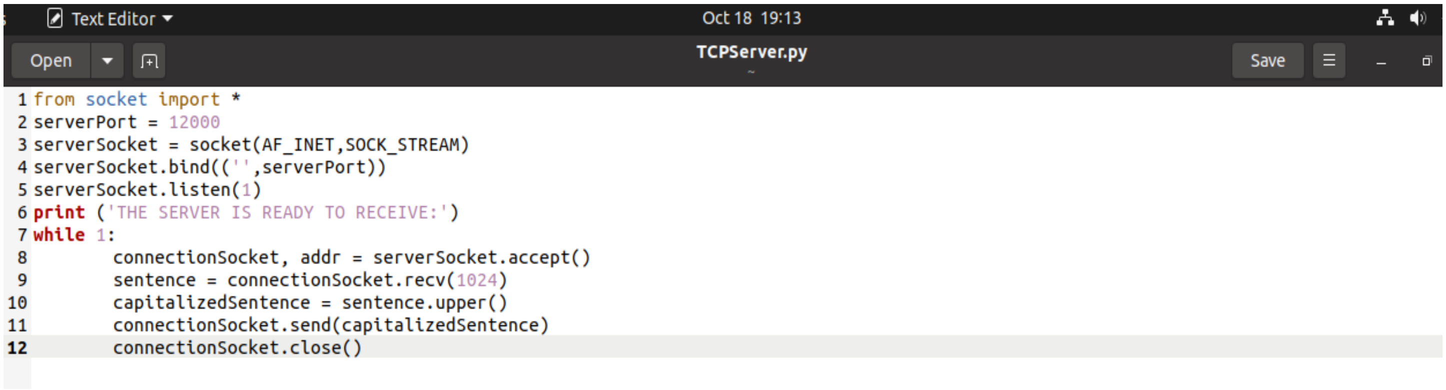
1 client pkt, 1 server pkt, 1 turn.

Entire conversation (72 bytes) Show and save data as ASCII Stream 0

wireshark_any_20201018174247_f06bnK.pcapng Packets: 2 · Displayed: 2 (100.0%) Profile: Default

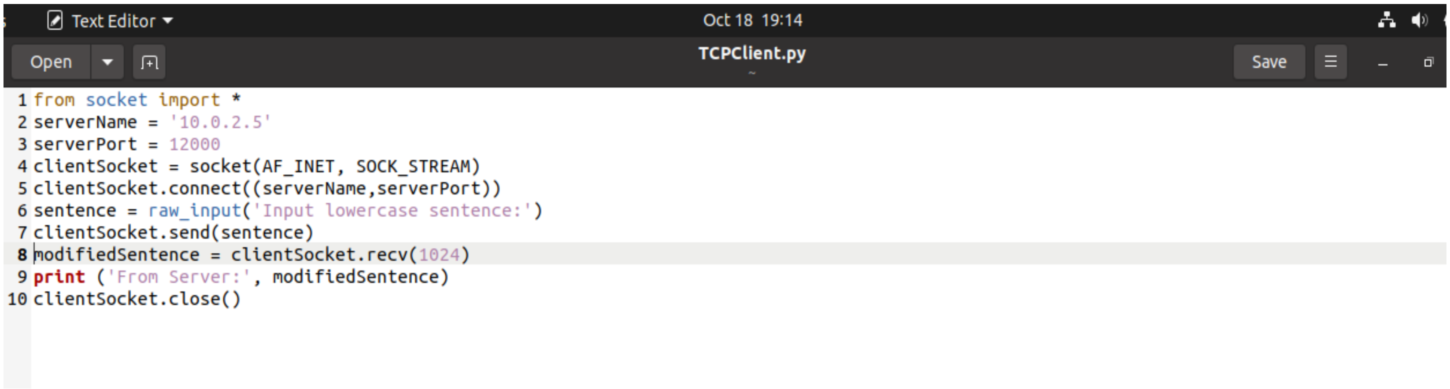
Socket Programming with TCP:-

TCPServer.py:



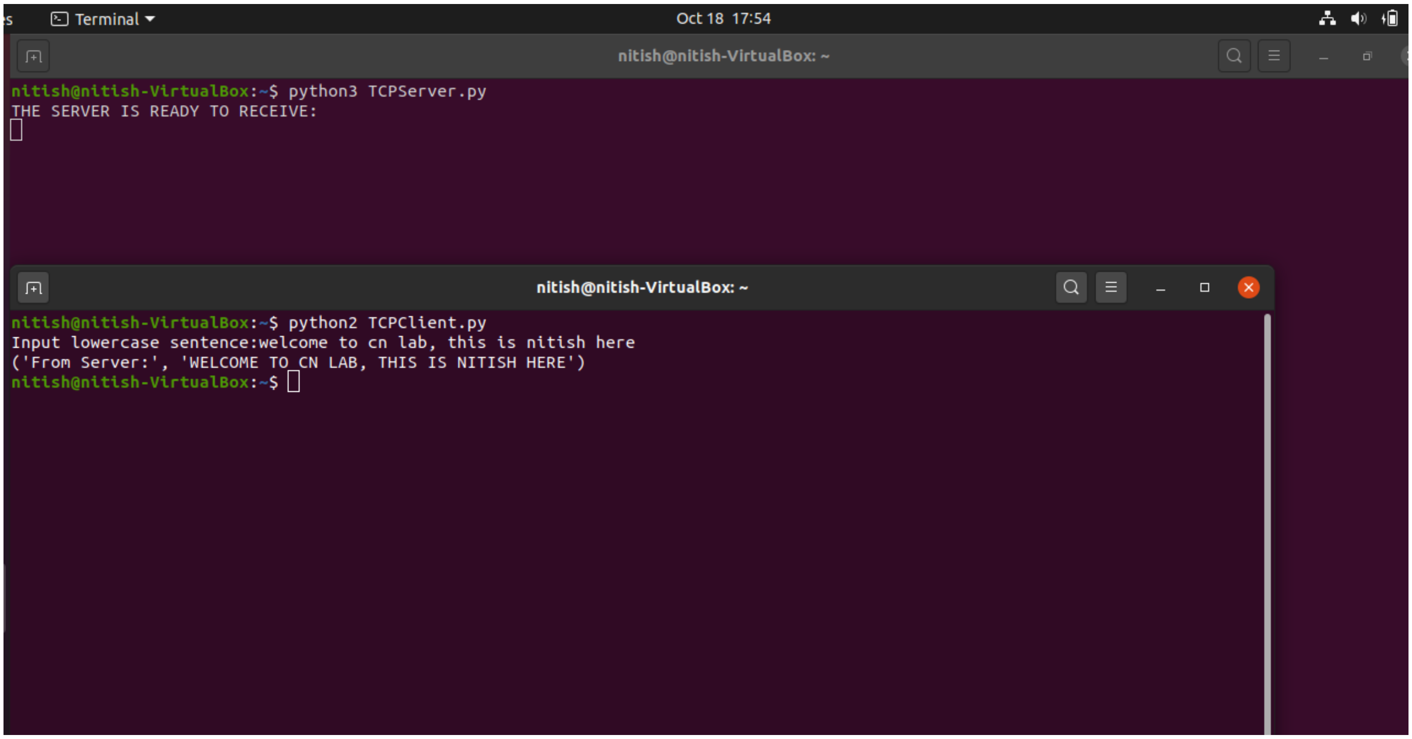
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print ('THE SERVER IS READY TO RECEIVE:')
while 1:
        connectionSocket, addr = serverSocket.accept()
        sentence = connectionSocket.recv(1024)
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
        connectionSocket.close()
```

TCPClient.py:



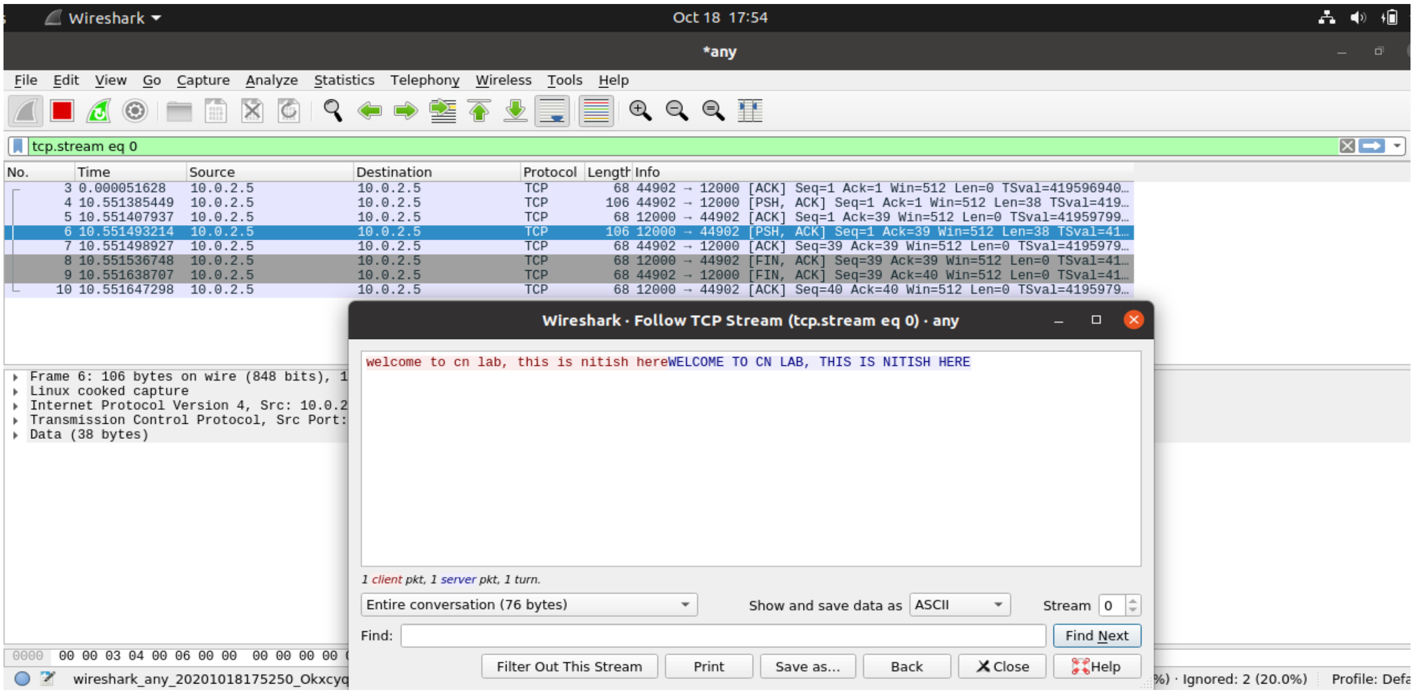
```
from socket import *
serverName = '10.0.2.5'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence)
clientSocket.close()
```

EXECUTION:



```
nitish@nitish-VirtualBox:~$ python3 TCPServer.py
THE SERVER IS READY TO RECEIVE:
```

```
nitish@nitish-VirtualBox:~$ python2 TCPClient.py
Input lowercase sentence:welcome to cn lab, this is nitish here
('From Server:', 'WELCOME TO CN LAB, THIS IS NITISH HERE')
nitish@nitish-VirtualBox:~$
```



Wireshark
Oct 18 17:54
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Follow TCP Stream (tcp.stream eq 0) · any
Filter Out This Stream
Print
Save as…
Back
Close
Help
Find Next

tcp.stream eq 0

```
No.  Time          Source     Destination  Protocol Length Info
   3 0.000051628   10.0.2.5   10.0.2.5     TCP       68 44902 → 12000 [ACK] Seq=1 Ack=1 Win=512 Len=0 TSval=419596940…
   4 10.551385449  10.0.2.5   10.0.2.5     TCP      106 44902 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=38 TSval=419…
   5 10.551407937  10.0.2.5   10.0.2.5     TCP       68 12000 → 44902 [ACK] Seq=1 Ack=39 Win=512 Len=0 TSval=41959799…
   6 10.551493214  10.0.2.5   10.0.2.5     TCP      106 12000 → 44902 [PSH, ACK] Seq=1 Ack=39 Win=512 Len=38 TSval=41…
   7 10.551498927  10.0.2.5   10.0.2.5     TCP       68 44902 → 12000 [ACK] Seq=39 Ack=39 Win=512 Len=0 TSval=4195979…
   8 10.551536748  10.0.2.5   10.0.2.5     TCP       68 12000 → 44902 [FIN, ACK] Seq=39 Ack=39 Win=512 Len=0 TSval=41…
   9 10.551638707  10.0.2.5   10.0.2.5     TCP       68 44902 → 12000 [FIN, ACK] Seq=39 Ack=40 Win=512 Len=0 TSval=41…
  10 10.551647298  10.0.2.5   10.0.2.5     TCP       68 12000 → 44902 [ACK] Seq=40 Ack=40 Win=512 Len=0 TSval=4195979…
```

```
Frame 6: 106 bytes on wire (848 bits), 1
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2
Transmission Control Protocol, Src Port:
Data (38 bytes)
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 (
```

wireshark_any_20201018175250_Okxcyq

%) · Ignored: 2 (20.0%)   Profile: Defa

```
welcome to cn lab, this is nitish hereWELCOME TO CN LAB, THIS IS NITISH HERE
```

1 client pkt, 1 server pkt, 1 turn.

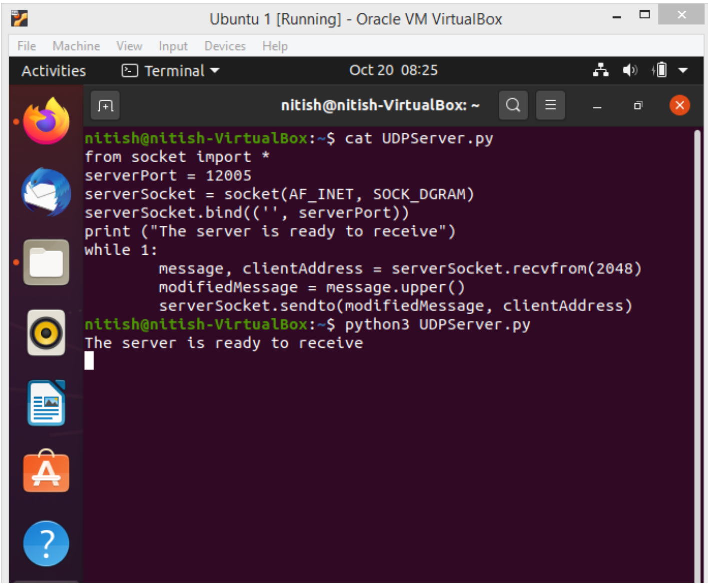
Entire conversation (76 bytes)   Show and save data as ASCII   Stream 0

Find:

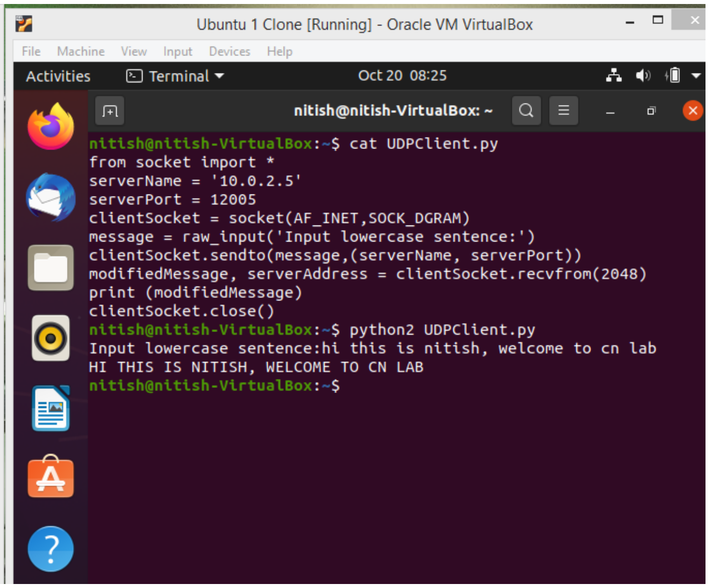
EXECUTION OF TASK 1 Using Two VMs:-

Client Machine: 10.0.2.4 (Ubuntu 1 Clone). Server Machine: 10.0.2.5 (Ubuntu 1)

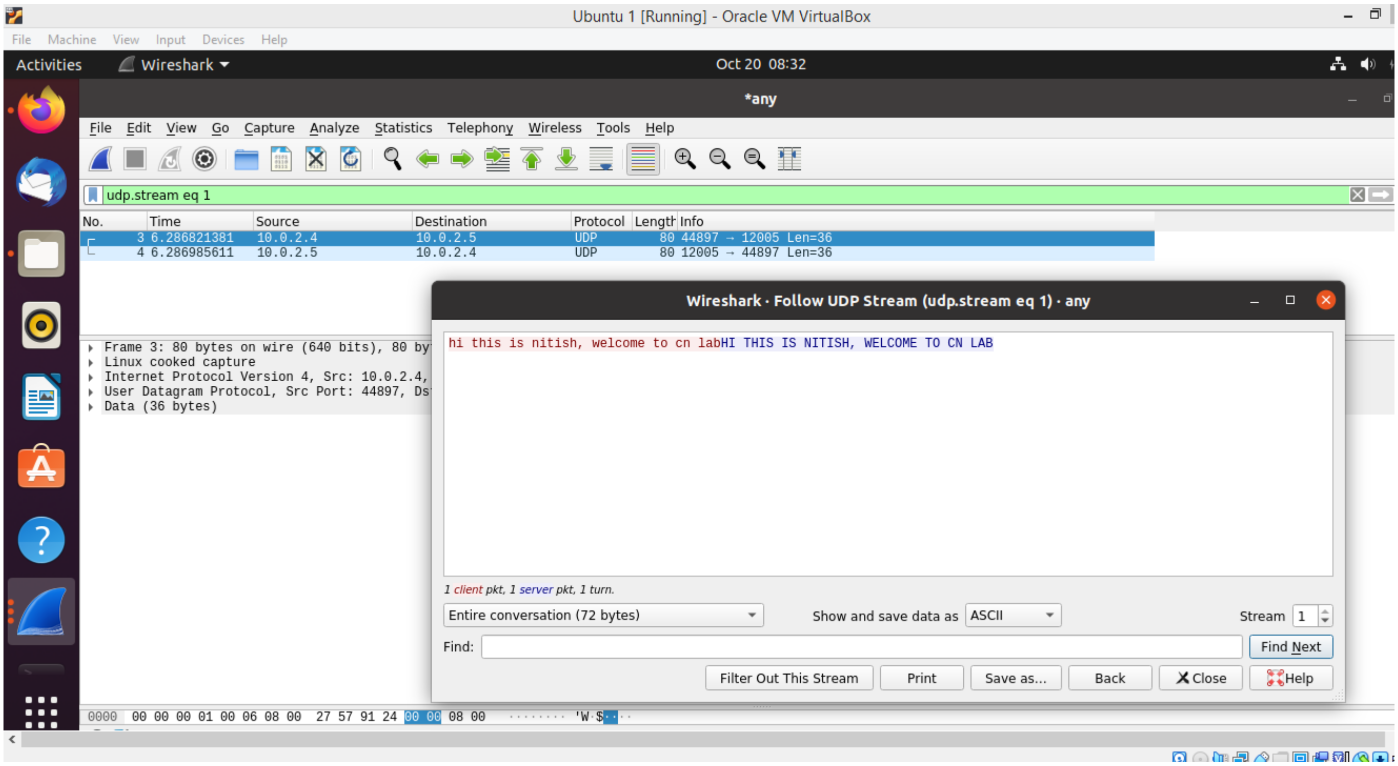
SOCKET PROGRAMMING USING UDP:-


Ubuntu 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 20 08:25
nitish@nitish-VirtualBox: ~

```
nitish@nitish-VirtualBox:~$ cat UDPServer.py
from socket import *
serverPort = 12005
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)
nitish@nitish-VirtualBox:~$ python3 UDPServer.py
The server is ready to receive
```


Ubuntu 1 Clone [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 20 08:25
nitish@nitish-VirtualBox: ~

```
nitish@nitish-VirtualBox:~$ cat UDPClient.py
from socket import *
serverName = '10.0.2.5'
serverPort = 12005
clientSocket = socket(AF_INET,SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print (modifiedMessage)
clientSocket.close()
nitish@nitish-VirtualBox:~$ python2 UDPClient.py
Input lowercase sentence:hi this is nitish, welcome to cn lab
HI THIS IS NITISH, WELCOME TO CN LAB
nitish@nitish-VirtualBox:~$
```


Ubuntu 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Oct 20 08:32
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Follow UDP Stream (udp.stream eq 1) · any
Filter Out This Stream
Print
Save as…
Back
Close
Help
Find Next

udp.stream eq 1

```
No.   Time         Source     Destination  Protocol Length Info
  3 6.286821381   10.0.2.4   10.0.2.5     UDP       80 44897 → 12005 Len=36
  4 6.286985611   10.0.2.5   10.0.2.4     UDP       80 12005 → 44897 Len=36
```

```
▸ Frame 3: 80 bytes on wire (640 bits), 80 by
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.4,
▸ User Datagram Protocol, Src Port: 44897, Ds
▸ Data (36 bytes)
```

```
hi this is nitish, welcome to cn labHI THIS IS NITISH, WELCOME TO CN LAB
```

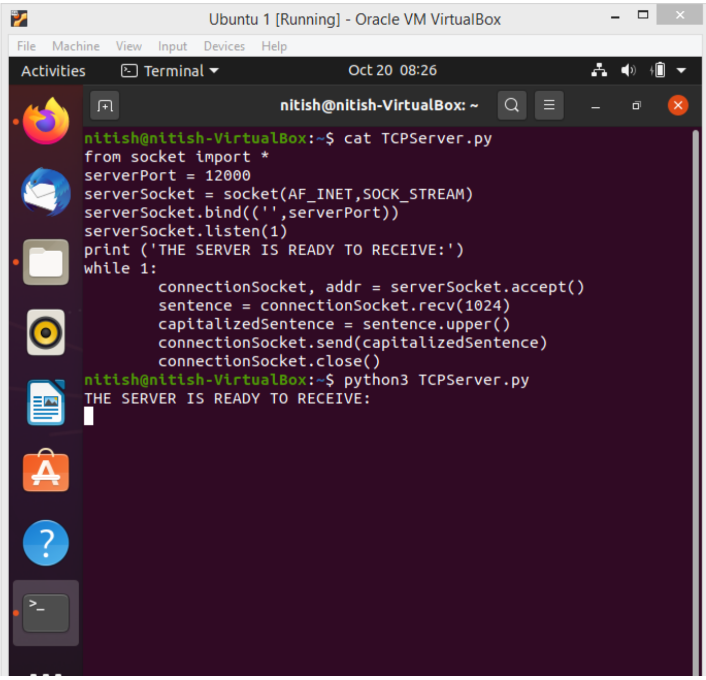
1 client pkt, 1 server pkt, 1 turn.

Entire conversation (72 bytes) Show and save data as ASCII Stream 1

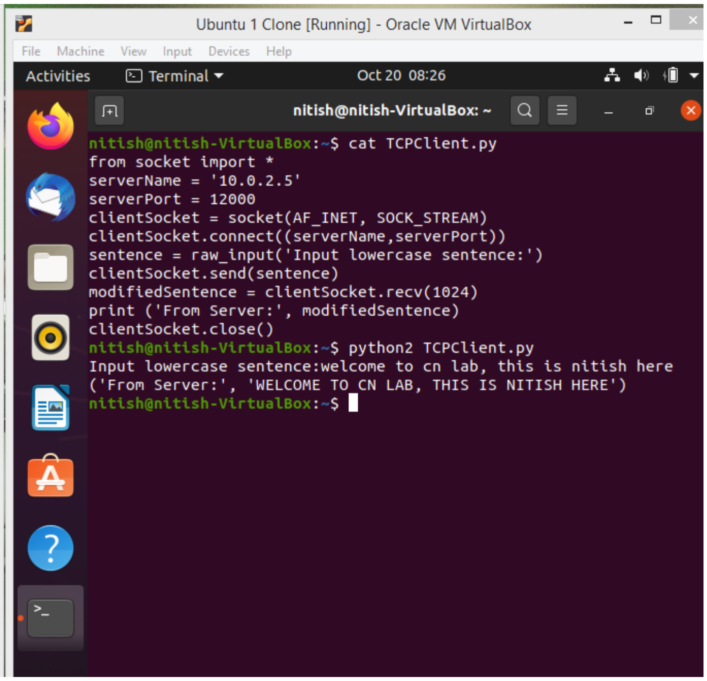
Find:

```
0000  00 00 00 01 00 06 08 00  27 57 91 24 00 00 08 00   ········ 'W·$····
```

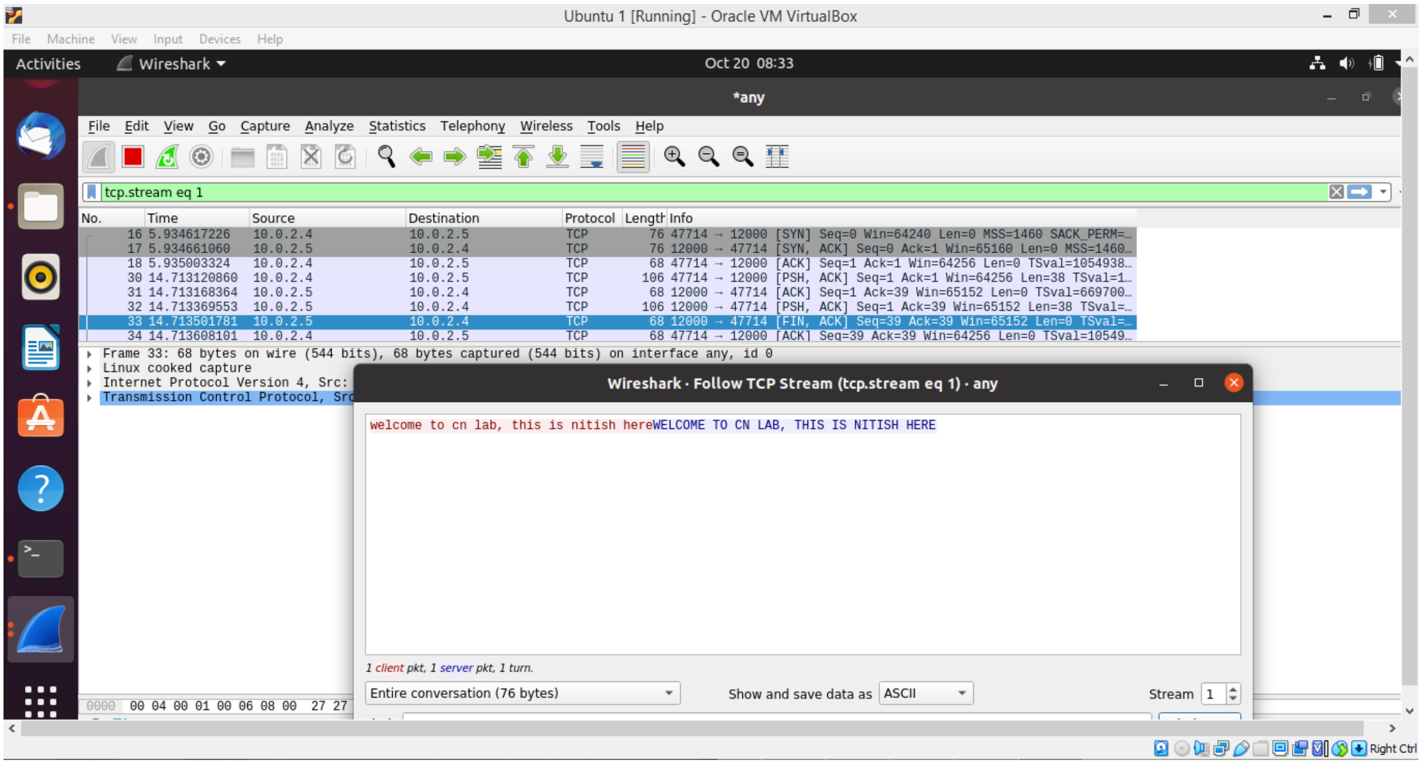
SOCKET PROGRAMMING USING TCP:-


Ubuntu 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 20 08:26
nitish@nitish-VirtualBox: ~

```
nitish@nitish-VirtualBox:~$ cat TCPServer.py
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print ('THE SERVER IS READY TO RECEIVE:')
while 1:
        connectionSocket, addr = serverSocket.accept()
        sentence = connectionSocket.recv(1024)
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
        connectionSocket.close()
nitish@nitish-VirtualBox:~$ python3 TCPServer.py
THE SERVER IS READY TO RECEIVE:
```


Ubuntu 1 Clone [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Oct 20 08:26
nitish@nitish-VirtualBox: ~

```
nitish@nitish-VirtualBox:~$ cat TCPClient.py
from socket import *
serverName = '10.0.2.5'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence)
clientSocket.close()
nitish@nitish-VirtualBox:~$ python2 TCPClient.py
Input lowercase sentence:welcome to cn lab, this is nitish here
('From Server:', 'WELCOME TO CN LAB, THIS IS NITISH HERE')
nitish@nitish-VirtualBox:~$
```


Ubuntu 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Oct 20 08:33
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Follow TCP Stream (tcp.stream eq 1) · any
Right Ctrl

tcp.stream eq 1

```
No.    Time          Source      Destination  Protocol Length Info
    16 5.934617226   10.0.2.4    10.0.2.5     TCP        76 47714 → 12000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=…
    17 5.934661060   10.0.2.5    10.0.2.4     TCP        76 12000 → 47714 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460…
    18 5.935003324   10.0.2.4    10.0.2.5     TCP        68 47714 → 12000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1054938…
    30 14.713120860  10.0.2.4    10.0.2.5     TCP       106 47714 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=38 TSval=1…
    31 14.713168364  10.0.2.5    10.0.2.4     TCP        68 12000 → 47714 [ACK] Seq=1 Ack=39 Win=65152 Len=0 TSval=669700…
    32 14.713369553  10.0.2.5    10.0.2.4     TCP       106 12000 → 47714 [PSH, ACK] Seq=1 Ack=39 Win=65152 Len=38 TSval=…
    33 14.713501781  10.0.2.5    10.0.2.4     TCP        68 12000 → 47714 [FIN, ACK] Seq=39 Ack=39 Win=65152 Len=0 TSval=…
    34 14.713608101  10.0.2.4    10.0.2.5     TCP        68 47714 → 12000 [ACK] Seq=39 Ack=39 Win=64256 Len=0 TSval=10549…

▸ Frame 33: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src:
▸ Transmission Control Protocol, Src

0000  00 04 00 01 00 06 08 00  27 27
```

```
welcome to cn lab, this is nitish hereWELCOME TO CN LAB, THIS IS NITISH HERE
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (76 bytes)   Show and save data as ASCII   Stream 1

Problems:

Install and compile the Python programs TCPClient and UDPClient on one host and TCPServer and UDPServer on another host.

1. Suppose you run TCPClient before you run TCPServer. What happens? Why?

Ans:-

If you run TCPClient first, then the client will attempt to make a TCP connection with a non-existent server process i.e. running TCPClient without TCPServer is same as running the Client without a server. A TCP connection will not be made.

2. Suppose you run UDPClient before you run UDPServer. What happens? Why?

Ans:-

For UDP, connection is established only when there is a transaction request from Client side. So if we run Client UDP program first it should work fine. But when a transaction request is made and server isn’t still running, the Client waits till server responds or till it gets response.

3. What happens if you use different port numbers for the client and server sides?

Ans:-

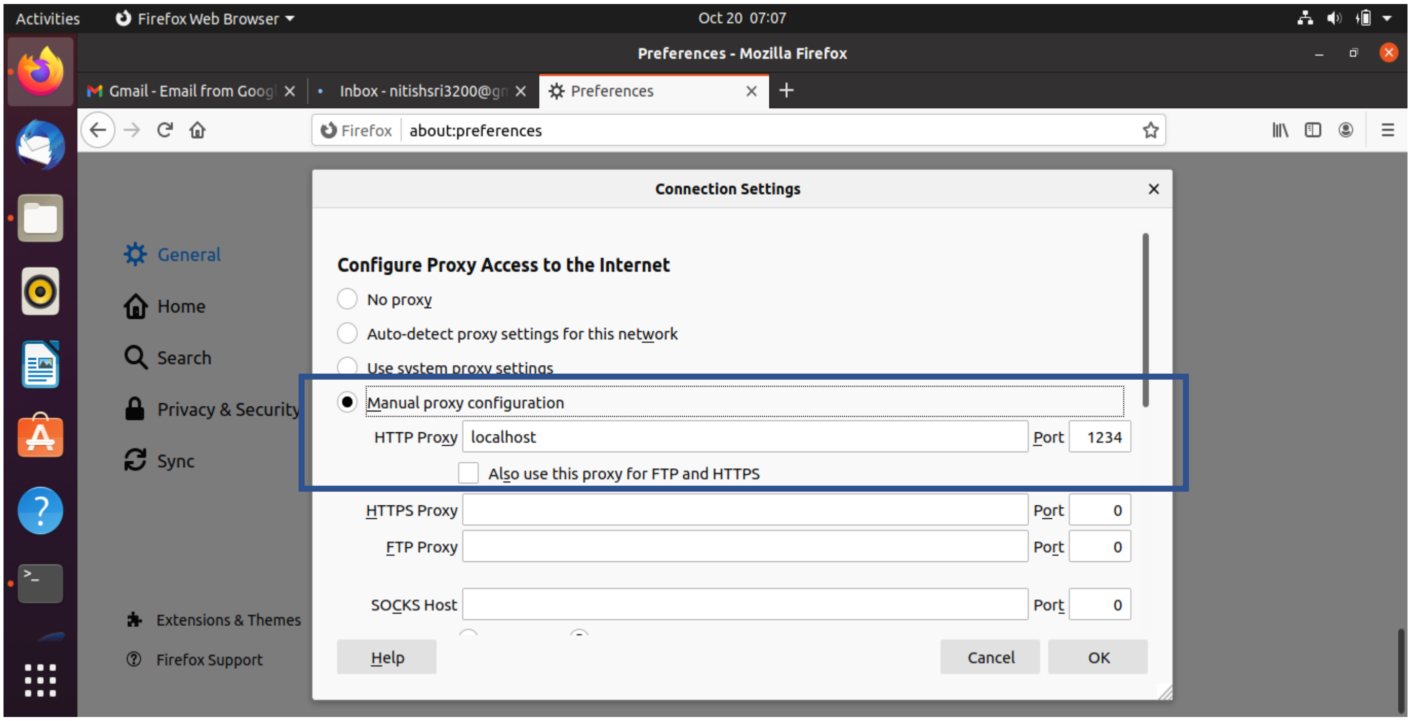
Using different ports is same like using no servers or clients. For TCP, connection won’t be established and for UDP it will work fine but will wait infinitely until response is received.

TASK – 3: Multi-Threaded Web Proxy

In this assignment, you will develop a Web proxy. When your proxy receives an HTTP request for an object from a browser, it generates a new HTTP request for the same object and sends it to the origin server. When the proxy receives the corresponding HTTP response with the object from the origin server, it creates a new HTTP response, including the object, and sends it to the client. This proxy will be multi-threaded, so that it will be able to handle multiple requests at the same time.
For this assignment, the companion Web site provides the skeleton code for the proxy server. Your job is to complete the code, and then test it by having different browsers request Web objects via your proxy.

Attached along with the submission ProxyServer.py file.

CONFIGURATIONS ON THE BROWSER:-

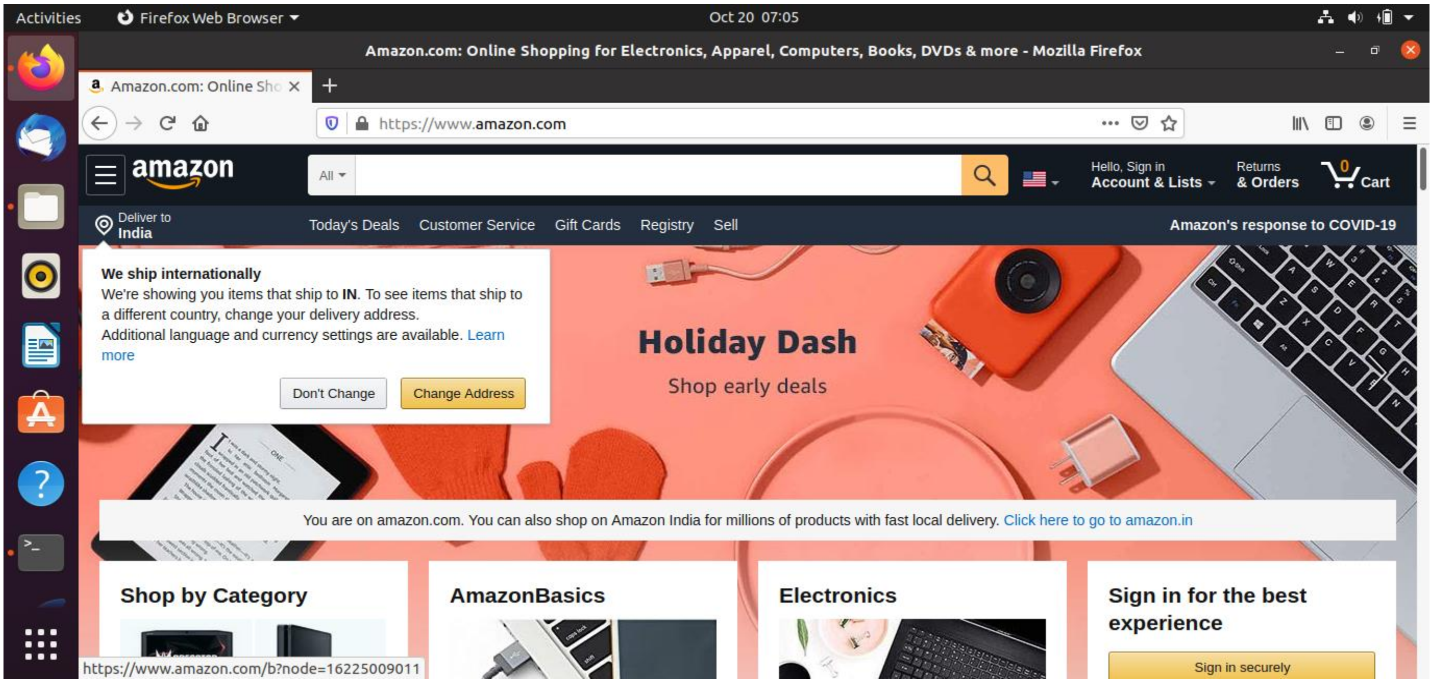


Configure Proxy Access to the Internet

○ No proxy
○ Auto-detect proxy settings for this network
○ Use system proxy settings
● Manual proxy configuration

HTTP Proxy: localhost Port: 1234
□ Also use this proxy for FTP and HTTPS
HTTPS Proxy: Port: 0
FTP Proxy: Port: 0
SOCKS Host: Port: 0

www.amazon.com on the browser

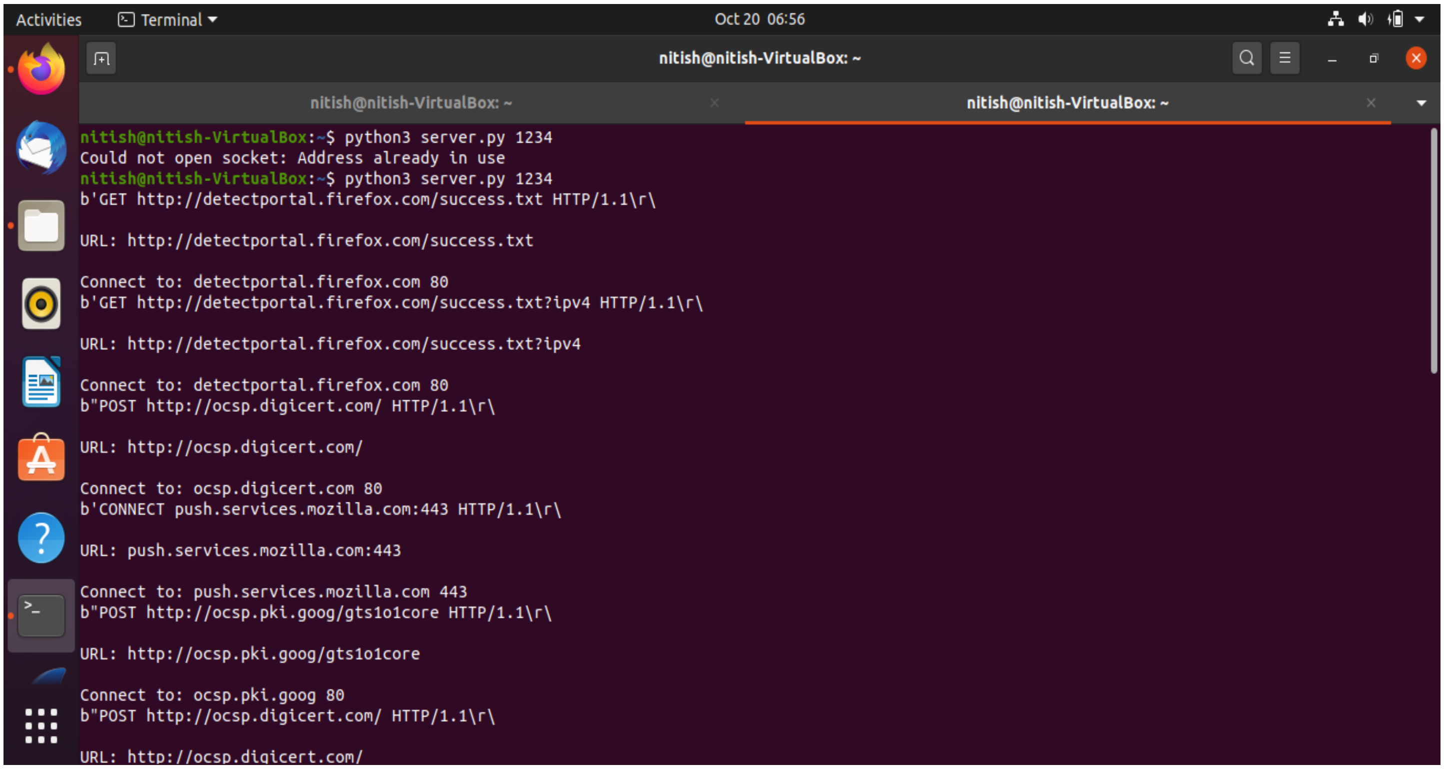


We ship internationally

We're showing you items that ship to **IN**. To see items that ship to a different country, change your delivery address.

Additional language and currency settings are available. Learn more

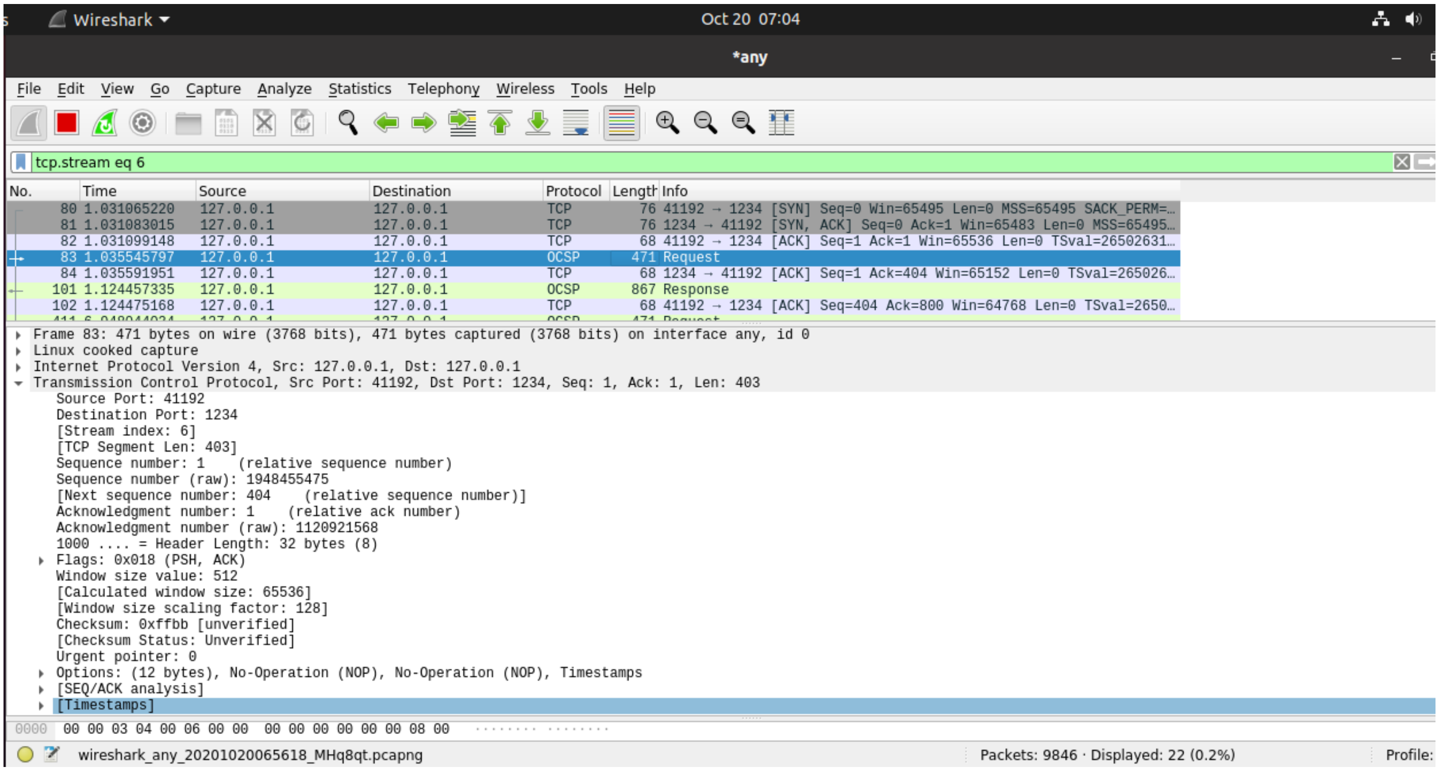
You are on amazon.com. You can also shop on Amazon India for millions of products with fast local delivery. Click here to go to amazon.in



Activities
Terminal
Oct 20 06:56
nitish@nitish-VirtualBox: ~
nitish@nitish-VirtualBox: ~
nitish@nitish-VirtualBox: ~

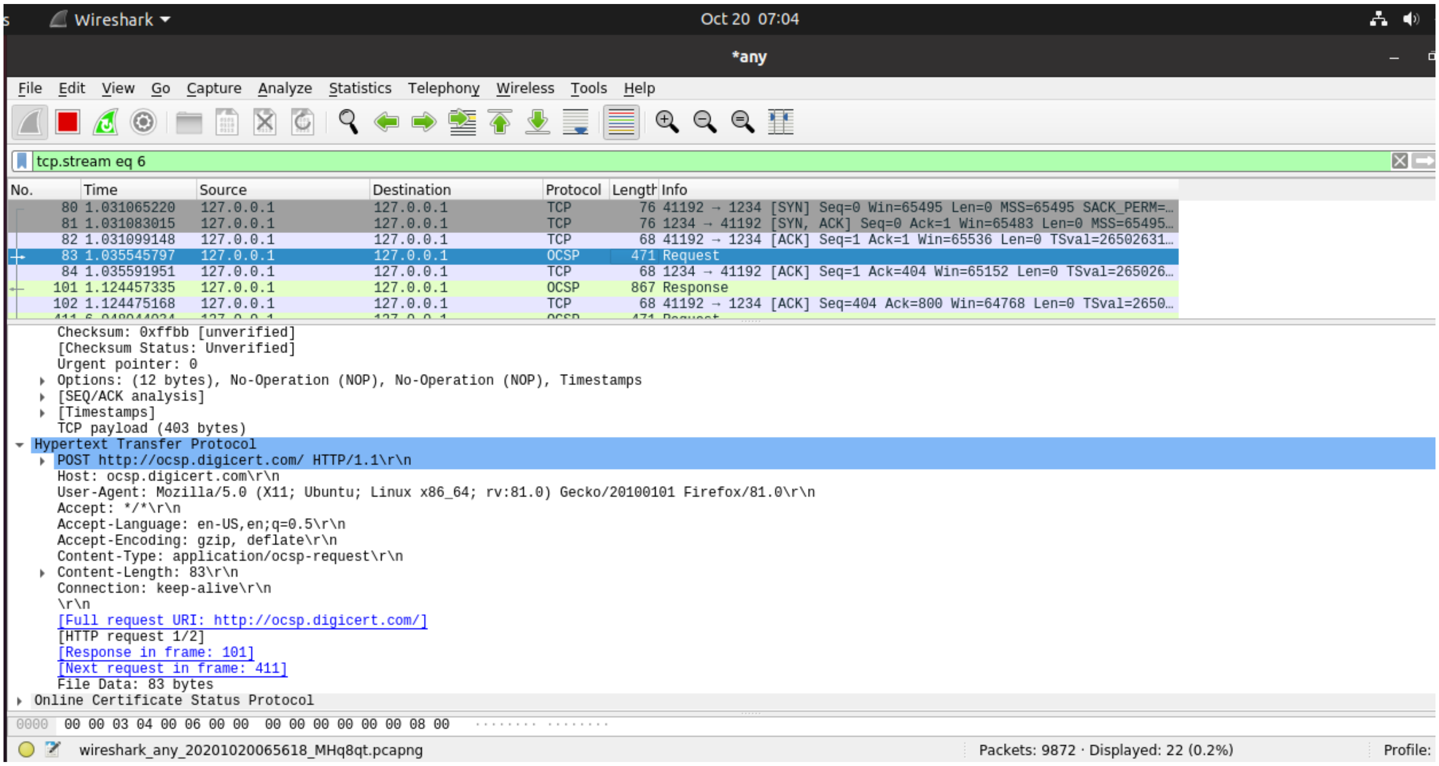
```
nitish@nitish-VirtualBox:~$ python3 server.py 1234
Could not open socket: Address already in use
nitish@nitish-VirtualBox:~$ python3 server.py 1234
b'GET http://detectportal.firefox.com/success.txt HTTP/1.1\r\

URL: http://detectportal.firefox.com/success.txt

Connect to: detectportal.firefox.com 80
b'GET http://detectportal.firefox.com/success.txt?ipv4 HTTP/1.1\r\

URL: http://detectportal.firefox.com/success.txt?ipv4

Connect to: detectportal.firefox.com 80
b"POST http://ocsp.digicert.com/ HTTP/1.1\r\

URL: http://ocsp.digicert.com/

Connect to: ocsp.digicert.com 80
b'CONNECT push.services.mozilla.com:443 HTTP/1.1\r\

URL: push.services.mozilla.com:443

Connect to: push.services.mozilla.com 443
b"POST http://ocsp.pki.goog/gts1o1core HTTP/1.1\r\

URL: http://ocsp.pki.goog/gts1o1core

Connect to: ocsp.pki.goog 80
b"POST http://ocsp.digicert.com/ HTTP/1.1\r\

URL: http://ocsp.digicert.com/
```

Wireshark capture: OCSP Request Message



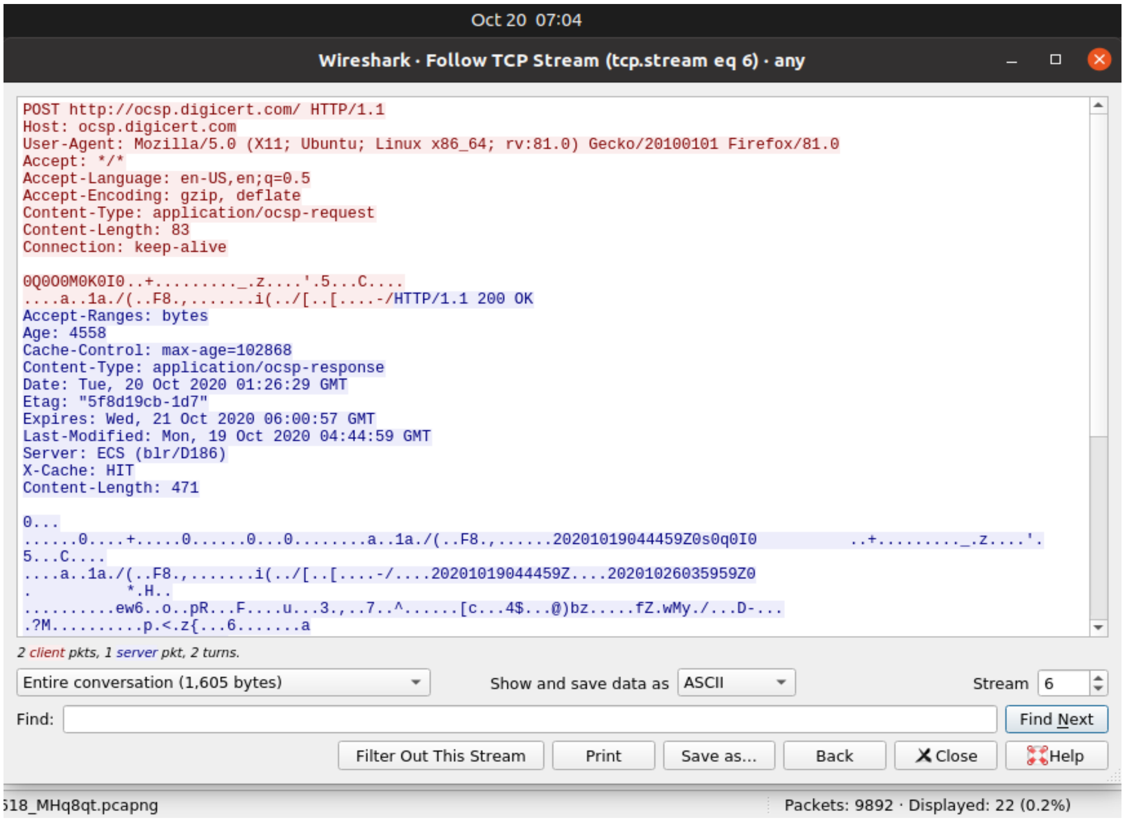
Wireshark
Oct 20 07:04
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.stream eq 6

No.   Time         Source      Destination  Protocol Length Info
   80 1.031065220  127.0.0.1   127.0.0.1    TCP        76 41192 → 1234 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
   81 1.031083015  127.0.0.1   127.0.0.1    TCP        76 1234 → 41192 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495…
   82 1.031099148  127.0.0.1   127.0.0.1    TCP        68 41192 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=26502631…
   83 1.035545797  127.0.0.1   127.0.0.1    OCSP      471 Request
   84 1.035591951  127.0.0.1   127.0.0.1    TCP        68 1234 → 41192 [ACK] Seq=1 Ack=404 Win=65152 Len=0 TSval=265026…
  101 1.124457335  127.0.0.1   127.0.0.1    OCSP      867 Response
  102 1.124475168  127.0.0.1   127.0.0.1    TCP        68 41192 → 1234 [ACK] Seq=404 Ack=800 Win=64768 Len=0 TSval=2650…

▸ Frame 83: 471 bytes on wire (3768 bits), 471 bytes captured (3768 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▾ Transmission Control Protocol, Src Port: 41192, Dst Port: 1234, Seq: 1, Ack: 1, Len: 403
    Source Port: 41192
    Destination Port: 1234
    [Stream index: 6]
    [TCP Segment Len: 403]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 1948455475
    [Next sequence number: 404    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 1120921568
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 512
    [Calculated window size: 65536]
    [Window size scaling factor: 128]
    Checksum: 0xffbb [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]

0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········

wireshark_any_20201020065618_MHq8qt.pcapng          Packets: 9846 · Displayed: 22 (0.2%)          Profile:
```



Wireshark
Oct 20 07:04
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.stream eq 6

No.   Time         Source      Destination  Protocol Length Info
   80 1.031065220  127.0.0.1   127.0.0.1    TCP        76 41192 → 1234 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
   81 1.031083015  127.0.0.1   127.0.0.1    TCP        76 1234 → 41192 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495…
   82 1.031099148  127.0.0.1   127.0.0.1    TCP        68 41192 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=26502631…
   83 1.035545797  127.0.0.1   127.0.0.1    OCSP      471 Request
   84 1.035591951  127.0.0.1   127.0.0.1    TCP        68 1234 → 41192 [ACK] Seq=1 Ack=404 Win=65152 Len=0 TSval=265026…
  101 1.124457335  127.0.0.1   127.0.0.1    OCSP      867 Response
  102 1.124475168  127.0.0.1   127.0.0.1    TCP        68 41192 → 1234 [ACK] Seq=404 Ack=800 Win=64768 Len=0 TSval=2650…

    Checksum: 0xffbb [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (403 bytes)
▾ Hypertext Transfer Protocol
  ▸ POST http://ocsp.digicert.com/ HTTP/1.1\r\n
    Host: ocsp.digicert.com\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:81.0) Gecko/20100101 Firefox/81.0\r\n
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/ocsp-request\r\n
  ▸ Content-Length: 83\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://ocsp.digicert.com/]
    [HTTP request 1/2]
    [Response in frame: 101]
    [Next request in frame: 411]
    File Data: 83 bytes
▸ Online Certificate Status Protocol

0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········

wireshark_any_20201020065618_MHq8qt.pcapng          Packets: 9872 · Displayed: 22 (0.2%)          Profile:
```



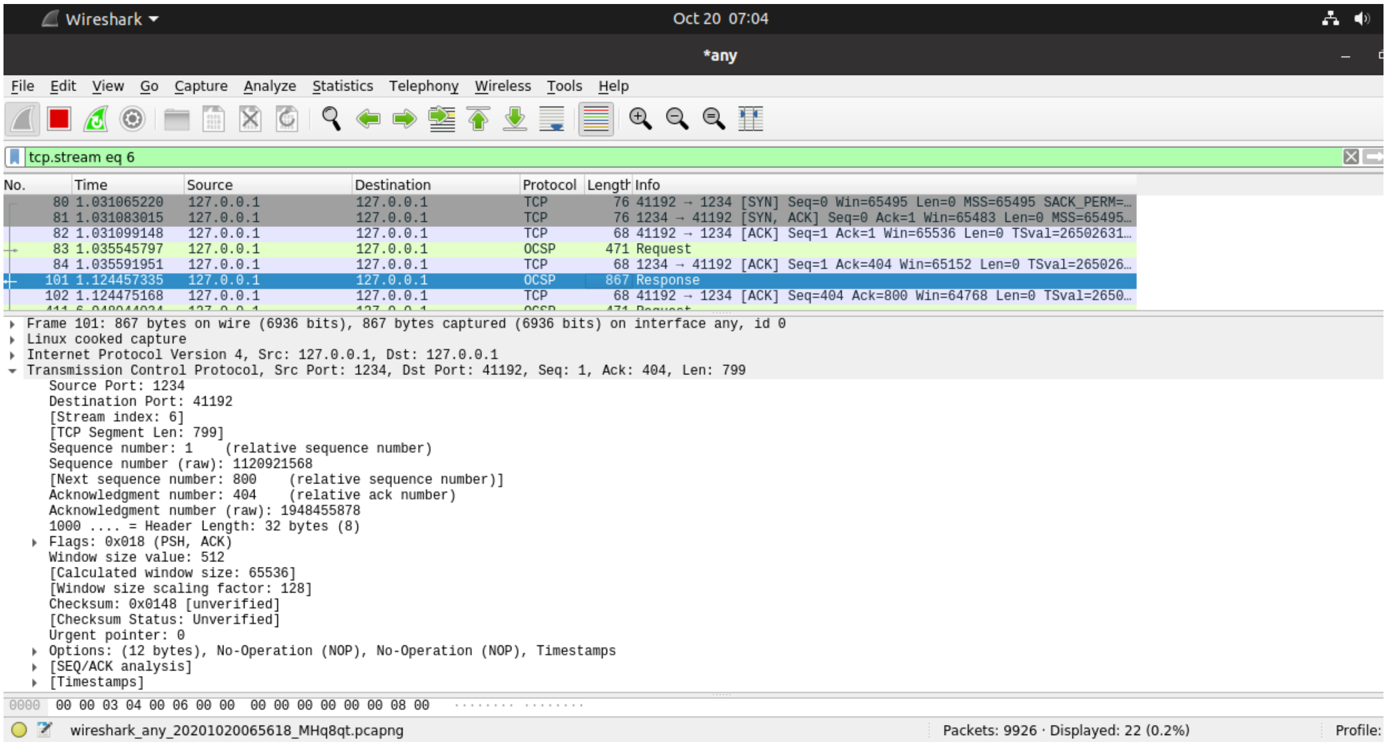
```
POST http://ocsp.digicert.com/ HTTP/1.1
Host: ocsp.digicert.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/ocsp-request
Content-Length: 83
Connection: keep-alive

0Q0O0M0K0I0..+........._.z....'.5...C....
....a..1a./(..F8.,.......i(../[..[.....-/HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 4558
Cache-Control: max-age=102868
Content-Type: application/ocsp-response
Date: Tue, 20 Oct 2020 01:26:29 GMT
Etag: "5f8d19cb-1d7"
Expires: Wed, 21 Oct 2020 06:00:57 GMT
Last-Modified: Mon, 19 Oct 2020 04:44:59 GMT
Server: ECS (blr/D186)
X-Cache: HIT
Content-Length: 471

0...
......0....+.....0......0...0........a..1a./(..F8.,......20201019044459Z0s0q0I0        ..+........._.z....'.
5...C....
....a..1a./(..F8.,.......i(../[..[.....-/....20201019044459Z....20201026035959Z0
.           *.H..
..........ew6..o..pR...F....u...3.,..7..^......[c...4$...@)bz.....fZ.wMy./...D-...
.?M..........p.<.z{...6.......a
```

2 client pkts, 1 server pkt, 2 turns.

Entire conversation (1,605 bytes) · Show and save data as: ASCII · Stream: 6

518_MHq8qt.pcapng · Packets: 9892 · Displayed: 22 (0.2%)

Wireshark capture: OCSP Response Message



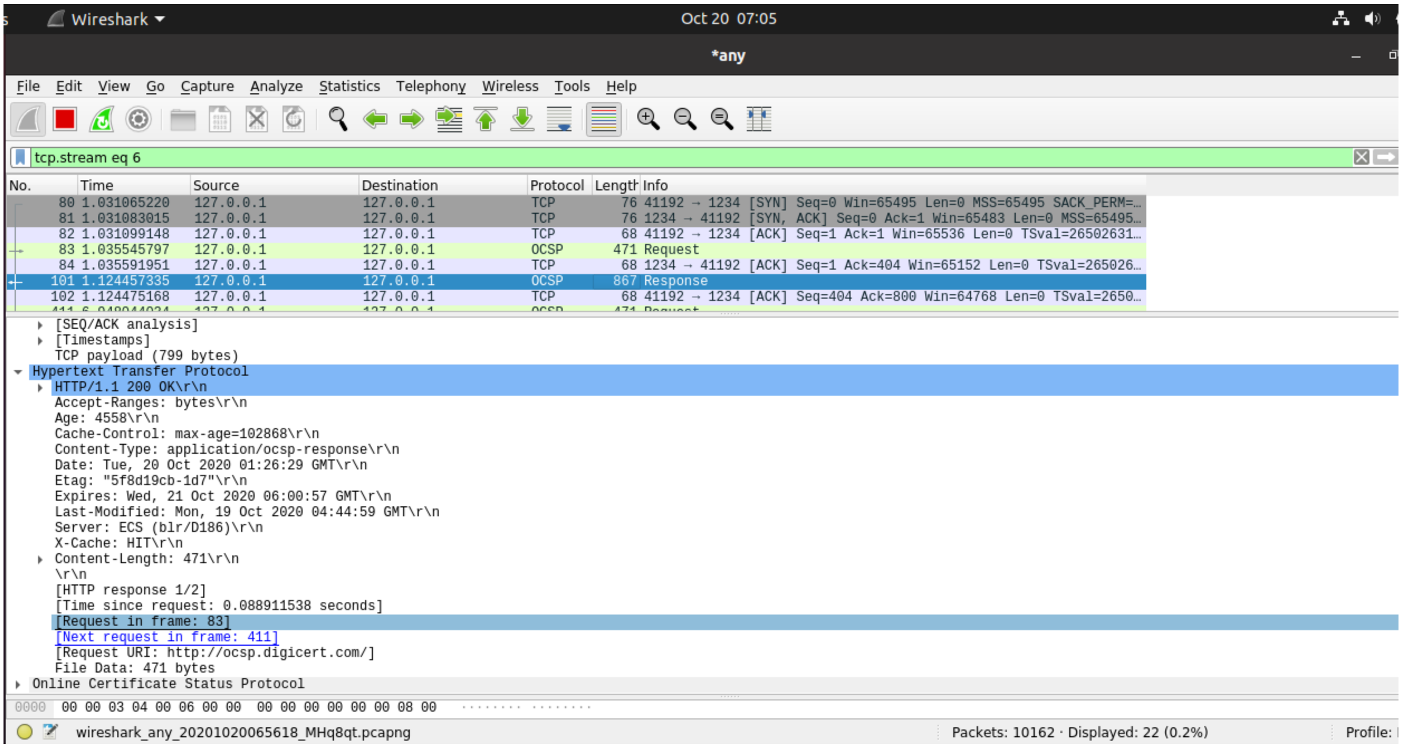
Wireshark
Oct 20 07:04
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Filter: tcp.stream eq 6

```
No.  Time         Source     Destination Protocol Length Info
  80 1.031065220  127.0.0.1  127.0.0.1   TCP        76 41192 → 1234 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
  81 1.031083015  127.0.0.1  127.0.0.1   TCP        76 1234 → 41192 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495…
  82 1.031099148  127.0.0.1  127.0.0.1   TCP        68 41192 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=26502631…
  83 1.035545797  127.0.0.1  127.0.0.1   OCSP      471 Request
  84 1.035591951  127.0.0.1  127.0.0.1   TCP        68 1234 → 41192 [ACK] Seq=1 Ack=404 Win=65152 Len=0 TSval=265026…
 101 1.124457335  127.0.0.1  127.0.0.1   OCSP      867 Response
 102 1.124475168  127.0.0.1  127.0.0.1   TCP        68 41192 → 1234 [ACK] Seq=404 Ack=800 Win=64768 Len=0 TSval=2650…
```

```
▸ Frame 101: 867 bytes on wire (6936 bits), 867 bytes captured (6936 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▾ Transmission Control Protocol, Src Port: 1234, Dst Port: 41192, Seq: 1, Ack: 404, Len: 799
    Source Port: 1234
    Destination Port: 41192
    [Stream index: 6]
    [TCP Segment Len: 799]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 1120921568
    [Next sequence number: 800    (relative sequence number)]
    Acknowledgment number: 404    (relative ack number)
    Acknowledgment number (raw): 1948455878
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 512
    [Calculated window size: 65536]
    [Window size scaling factor: 128]
    Checksum: 0x0148 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
```

wireshark_any_20201020065618_MHq8qt.pcapng · Packets: 9926 · Displayed: 22 (0.2%) · Profile:



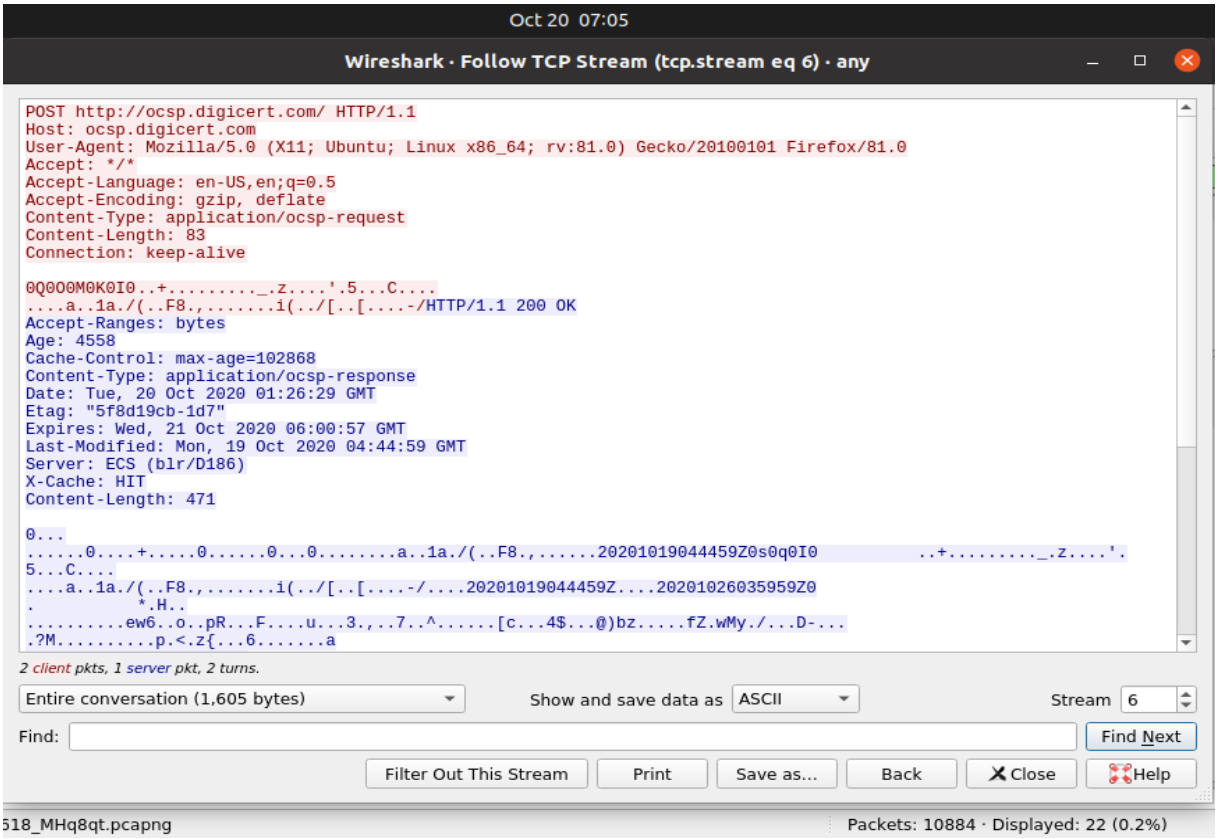
Wireshark
Oct 20 07:05
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.stream eq 6
```

```
No.   Time         Source      Destination  Protocol Length Info
 80 1.031065220  127.0.0.1   127.0.0.1    TCP       76 41192 → 1234 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
 81 1.031083015  127.0.0.1   127.0.0.1    TCP       76 1234 → 41192 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495…
 82 1.031099148  127.0.0.1   127.0.0.1    TCP       68 41192 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=26502631…
 83 1.035545797  127.0.0.1   127.0.0.1    OCSP     471 Request
 84 1.035591951  127.0.0.1   127.0.0.1    TCP       68 1234 → 41192 [ACK] Seq=1 Ack=404 Win=65152 Len=0 TSval=265026…
101 1.124457335  127.0.0.1   127.0.0.1    OCSP     867 Response
102 1.124475168  127.0.0.1   127.0.0.1    TCP       68 41192 → 1234 [ACK] Seq=404 Ack=800 Win=64768 Len=0 TSval=2650…
```

```
    ▸ [SEQ/ACK analysis]
    ▸ [Timestamps]
      TCP payload (799 bytes)
▾ Hypertext Transfer Protocol
    ▸ HTTP/1.1 200 OK\r\n
      Accept-Ranges: bytes\r\n
      Age: 4558\r\n
      Cache-Control: max-age=102868\r\n
      Content-Type: application/ocsp-response\r\n
      Date: Tue, 20 Oct 2020 01:26:29 GMT\r\n
      Etag: "5f8d19cb-1d7"\r\n
      Expires: Wed, 21 Oct 2020 06:00:57 GMT\r\n
      Last-Modified: Mon, 19 Oct 2020 04:44:59 GMT\r\n
      Server: ECS (blr/D186)\r\n
      X-Cache: HIT\r\n
    ▸ Content-Length: 471\r\n
      \r\n
      [HTTP response 1/2]
      [Time since request: 0.088911538 seconds]
      [Request in frame: 83]
      [Next request in frame: 411]
      [Request URI: http://ocsp.digicert.com/]
      File Data: 471 bytes
▸ Online Certificate Status Protocol
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
```

wireshark_any_20201020065618_MHq8qt.pcapng
Packets: 10162 · Displayed: 22 (0.2%)
Profile:



Oct 20 07:05
Wireshark · Follow TCP Stream (tcp.stream eq 6) · any

```
POST http://ocsp.digicert.com/ HTTP/1.1
Host: ocsp.digicert.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/ocsp-request
Content-Length: 83
Connection: keep-alive

0Q0O0M0K0I0..+........._.z....'.5...C....
....a..1a./(..F8.,.......i(../[..[....-/HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 4558
Cache-Control: max-age=102868
Content-Type: application/ocsp-response
Date: Tue, 20 Oct 2020 01:26:29 GMT
Etag: "5f8d19cb-1d7"
Expires: Wed, 21 Oct 2020 06:00:57 GMT
Last-Modified: Mon, 19 Oct 2020 04:44:59 GMT
Server: ECS (blr/D186)
X-Cache: HIT
Content-Length: 471

0...
......0....+.....0......0...0........a..1a./(..F8.,......20201019044459Z0s0q0I0         ..+........._.z....'.
5...C....
....a..1a./(..F8.,.......i(../[..[....-/....20201019044459Z....20201026035959Z0
.           *.H..
..........ew6..o..pR...F....u...3.,..7..^......[c...4$...@)bz.....fZ.wMy./...D-...
.?M.........p.<.z{...6.......a
```

2 client pkts, 1 server pkt, 2 turns.

Entire conversation (1,605 bytes) ▾ Show and save data as ASCII ▾ Stream 6

Find:

Report:

The proxy sits between the client (usually web browser) and the server (web server). In our simple case, the client sends all its requests to the proxy instead of sending requests directly to the server. The proxy then opens a connection to the server, and passes on the client's request. Then when the proxy receives the reply from the server, it sends that reply back to the client. There are several reasons we use proxy for our browser: Performance (the proxy caches the pages that it fetched), Content Filtering and Transformation (block access to certain domain, reformat web pages), and Privacy.

In the main function, we create a socket to listen requests from client (web browser). The port of the socket is the command argument of the program. Since the proxy needs to handle multiple clients at the same time, we need to implement multi-threading for it. Whenever the proxy received a request from client, it creates a thread to handle the request t**hread.start_new_thread(proxy_thread, (conn, client_addr)).**

The proxy_thread function firstly parse the web server URL and port (if the port is not defined, default port 80 will be used). For example, the first line of the request from client is **GET http://www.amazon.com/ HTTP/1.1** we need to parse the URL **www.amazon.com**. When the URL is ready, the proxy just create a connection to server using the URL, send the request to it to receive back resulted web page and then send the web page to web browser.