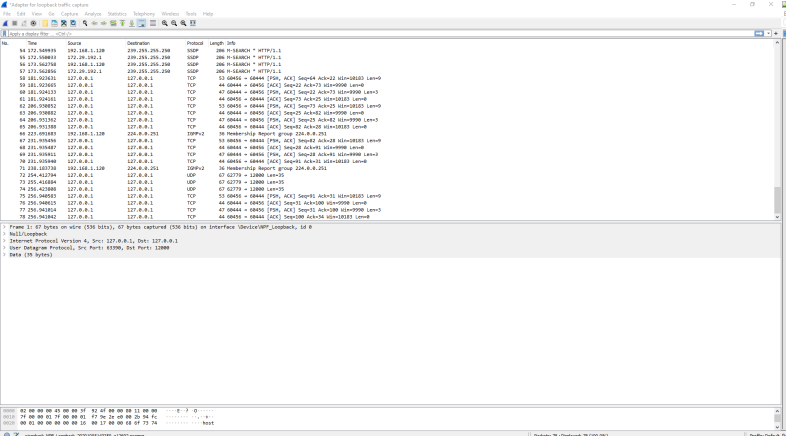
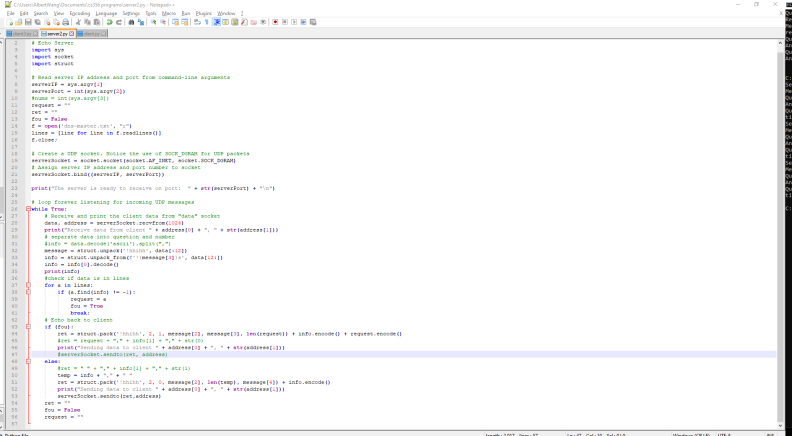
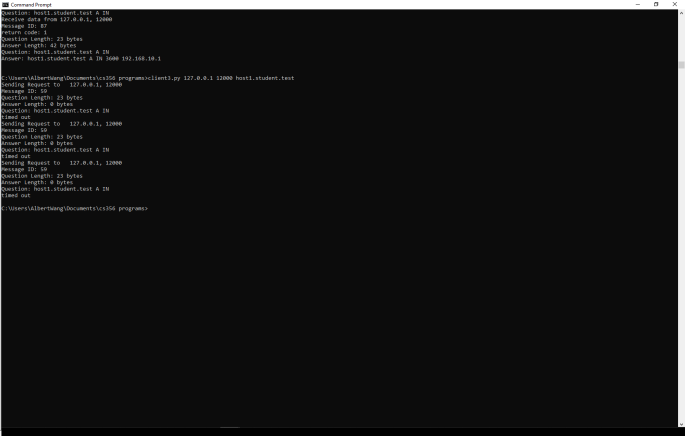

Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
94 172.569935 192.168.1.128 239.255.255.250 SSDP 286 M-SEARCH * HTTP/1.1
95 172.598810 172.20.192.1 239.255.255.250 SSDP 286 M-SEARCH * HTTP/1.1
96 173.562758 192.168.1.128 239.255.255.250 SSDP 286 M-SEARCH * HTTP/1.1
97 173.562856 172.20.192.1 239.255.255.250 SSDP 286 M-SEARCH * HTTP/1.1
58 181.923631 127.0.0.1 127.0.0.1 TCP 53 60456 → 60444 [PSH, ACK] Seq=64 Ack=22 Win=10183 Len=9
59 181.923665 127.0.0.1 127.0.0.1 TCP 44 60444 → 60456 [ACK] Seq=22 Ack=73 Win=9998 Len=0
60 181.924131 127.0.0.1 127.0.0.1 TCP 44 60444 → 60456 [FIN, ACK] Seq=22 Ack=73 Win=9998 Len=3
61 181.924161 127.0.0.1 127.0.0.1 TCP 44 60456 → 60444 [ACK] Seq=73 Ack=25 Win=10183 Len=0
62 186.936852 127.0.0.1 127.0.0.1 TCP 53 60456 → 60444 [PSH, ACK] Seq=73 Ack=25 Win=10183 Len=9
63 186.936882 127.0.0.1 127.0.0.1 TCP 44 60444 → 60456 [ACK] Seq=25 Ack=82 Win=9998 Len=0
64 186.953362 127.0.0.1 127.0.0.1 TCP 47 60444 → 60456 [PSH, ACK] Seq=25 Ack=82 Win=9998 Len=3
65 186.953388 127.0.0.1 127.0.0.1 TCP 44 60456 → 60444 [ACK] Seq=82 Ack=28 Win=10183 Len=0
66 223.691683 192.168.1.128 224.0.0.251 IGMPv2 46 Membership Report group 224.0.0.251
67 231.939496 127.0.0.1 127.0.0.1 TCP 53 60456 → 60444 [PSH, ACK] Seq=82 Ack=28 Win=10183 Len=9
68 231.939487 127.0.0.1 127.0.0.1 TCP 44 60444 → 60456 [ACK] Seq=28 Ack=91 Win=9998 Len=0
69 231.939511 127.0.0.1 127.0.0.1 TCP 47 60444 → 60456 [PSH, ACK] Seq=28 Ack=91 Win=9998 Len=3
70 231.939540 127.0.0.1 127.0.0.1 TCP 44 60456 → 60444 [ACK] Seq=91 Ack=31 Win=10183 Len=0
71 238.183738 192.168.1.128 224.0.0.251 IGMPv2 46 Membership Report group 224.0.0.251
72 254.412794 127.0.0.1 127.0.0.1 UDP 67 62779 → 12000 Len=35
73 255.412889 127.0.0.1 127.0.0.1 UDP 67 62779 → 12000 Len=35
74 256.428885 127.0.0.1 127.0.0.1 UDP 67 62779 → 12000 Len=35
75 256.944883 127.0.0.1 127.0.0.1 TCP 53 60456 → 60444 [PSH, ACK] Seq=91 Ack=31 Win=10183 Len=9
76 256.944913 127.0.0.1 127.0.0.1 TCP 44 60444 → 60456 [ACK] Seq=31 Ack=100 Win=9998 Len=0
77 256.943814 127.0.0.1 127.0.0.1 TCP 47 60444 → 60456 [PSH, ACK] Seq=31 Ack=100 Win=9998 Len=3
78 256.943841 127.0.0.1 127.0.0.1 TCP 44 60456 → 60444 [ACK] Seq=100 Ack=34 Win=10183 Len=0
Frame 1: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface \Device\NPF_Loopback, id 0
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 63390, Dst Port: 12000
Data (35 bytes)
0000 02 00 00 00 45 00 00 3f 92 4f 00 00 80 11 00 00 ····E··? ·O····
0010 7f 00 00 01 7f 00 00 01 f7 9e 2e e0 00 2b 2a 4c ···· ···· ·····+*L
0020 00 01 00 00 00 00 00 1c 00 17 00 00 68 6f 73 74 ···· ···· ····host
Packets: 78 · Displayed: 78 (100.0%)
Profile: Default
nmshark_NPF_Loopback_20210051143358_a12092.pcapng


C:\Users\AlbertWang\Documents\cs356 programs\server2.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
client.py server2.py client.py
# Echo Server
import sys
import socket
import struct
# Read server IP address and port from command-line arguments
serverIP = sys.argv[1]
serverPort = int(sys.argv[2])
#cuse = int(sys.argv[3])
request = ""
ret = ""
fcc = False
f = open('the-server.txt', "r")
lines = [line for line in f.readlines()]
f.close()
# Create a UDP socket. Notice the use of SOCK_DGRAM for UDP packets
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Assign server IP address and port number to socket
serverSocket.bind((serverIP, serverPort))
print("The server is ready to receive on port: " + str(serverPort) + "\n")
# Loop forever listening for incoming UDP messages
while True:
# Receive and print the client data from "data" socket
data, address = serverSocket.recvfrom(1024)
print("Receive data from client " + address[0] + ", " + str(address[1]))
# separate data into question and number
$info = data.decode('ascii').split(",")
message = struct.unpack('!HHhhh', data[:12])
info = struct.unpack_from(f'!{message[3]}s', data[12:])
info = info[0].decode()
print(info)
#check if data is in lines
for a in lines:
if (a.find(info) != -1):
request = a
fcc = True
break;
# Echo back to client
if (fcc):
ret = struct.pack('!HHhhh', 2, 1, message[2], message[3], len(request)) + info.encode() + request.encode()
#ret = request + "," + info[1] + "," + str(t)
print("Sending data to client " + address[0] + ", " + str(address[1]))
#serverSocket.sendto(ret, address)
else:
#cut = " " + ",," + info[1] + "," + str(t)
temp = info + "," + " "
ret = struct.pack('!HHhhh', 2, 0, message[2], len(temp), message[3]) + info.encode()
print("Sending data to client " + address[0] + ", " + str(address[1]))
serverSocket.sendto(ret,address)
ret = ""
fcc = False
request = ""
length : 2,017 lines : 57
Ln : 47 Col : 10 Sel : 0 | 0
Windows (CR LF) UTF-8 INS


Command Prompt
Host:host1.student.test A IN
Receive data from 127.0.0.1, 12000
Message ID: 87
Return code: 1
Question Length: 23 bytes
Answer Length: 42 bytes
Question: host1.student.test A IN
Answer: host1.student.test A IN 3600 192.168.10.1
C:\Users\AlbertWang\Documents\cs356 programs>client3.py 127.0.0.1 12000 host1.student.test
Sending Request to 127.0.0.1, 12000
Message ID: 87
Question Length: 23 bytes
Answer Length: 0 bytes
Question: host1.student.test A IN
Timed out
Sending Request to 127.0.0.1, 12000
Message ID: 16
Question Length: 23 bytes
Answer Length: 0 bytes
Question: host1.student.test A IN
Timed out
Sending Request to 127.0.0.1, 12000
Message ID: 19
Question Length: 23 bytes
Answer Length: 0 bytes
Question: host1.student.test A IN
Timed out
C:\Users\AlbertWang\Documents\cs356 programs>