

Task 1.1 step 1

```
Command Prompt
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Robby>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 2:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::b9ca:8b72:1de3:2588%42
    IPv4 Address. . . . . : 192.168.56.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 13:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 14:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :
```

IPv4 Route Table

=====

Active Routes:

Network	Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	0.0.0.0	192.168.0.1	192.168.0.130	35
127.0.0.0	255.0.0.0	255.0.0.0	On-link	127.0.0.1	331
127.0.0.1	255.255.255.255	255.255.255.255	On-link	127.0.0.1	331
127.255.255.255	255.255.255.255	255.255.255.255	On-link	127.0.0.1	331
192.168.0.0	255.255.255.0	255.255.255.0	On-link	192.168.0.130	291
192.168.0.130	255.255.255.255	255.255.255.255	On-link	192.168.0.130	291
192.168.0.255	255.255.255.255	255.255.255.255	On-link	192.168.0.130	291
192.168.56.0	255.255.255.0	255.255.255.0	On-link	192.168.56.1	330
192.168.56.1	255.255.255.255	255.255.255.255	On-link	192.168.56.1	330
192.168.56.255	255.255.255.255	255.255.255.255	On-link	192.168.56.1	330
224.0.0.0	240.0.0.0	240.0.0.0	On-link	127.0.0.1	331
224.0.0.0	240.0.0.0	240.0.0.0	On-link	192.168.56.1	330
224.0.0.0	240.0.0.0	240.0.0.0	On-link	192.168.0.130	291
255.255.255.255	255.255.255.255	255.255.255.255	On-link	127.0.0.1	331
255.255.255.255	255.255.255.255	255.255.255.255	On-link	192.168.56.1	330
255.255.255.255	255.255.255.255	255.255.255.255	On-link	192.168.0.130	291

=====

Persistent Routes:

None

Capturing from Wi-Fi 3 (arp)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
821	5621.807426	ba:bf:6d:71:76:86	Broadcast	ARP	60	Who has 192.168.0.89? Tel
822	5639.037624	Technico_ff:3e:5d	TP-Link_5c:53:1b	ARP	60	Who has 192.168.0.130? Te
823	5639.037634	TP-Link_5c:53:1b	Technico_ff:3e:5d	ARP	42	192.168.0.130 is at 54:af
824	5669.244234	Technico_ff:3e:5d	TP-Link_5c:53:1b	ARP	60	Who has 192.168.0.130? Te
825	5669.244241	TP-Link_5c:53:1b	Technico_ff:3e:5d	ARP	42	192.168.0.130 is at 54:af
826	5673.825525	Ring_33:8b:b4	Broadcast	ARP	60	Who has 192.168.0.1? Tell
827	5699.452094	Technico_ff:3e:5d	TP-Link_5c:53:1b	ARP	60	Who has 192.168.0.130? Te
828	5699.452105	TP-Link_5c:53:1b	Technico_ff:3e:5d	ARP	42	192.168.0.130 is at 54:af
829	5704.979495	TP-Link_5c:53:1b	Microsof_e0:b7:0b	ARP	42	Who has 192.168.0.89? Tel
830	5705.026852	Microsof_e0:b7:0b	TP-Link_5c:53:1b	ARP	60	192.168.0.89 is at 2c:54:
831	5729.660066	Technico_ff:3e:5d	TP-Link_5c:53:1b	ARP	60	Who has 192.168.0.130? Te
832	5729.660077	TP-Link_5c:53:1b	Technico_ff:3e:5d	ARP	42	192.168.0.130 is at 54:af

< >

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: Technico_ff:3e:5d (ac:4c:a5:ff:3e:5d), Dst: TP-Link_5c:53:1b (54:af:97:5c:53:1b)
> Address Resolution Protocol (request)

0000 54 af 97 5c 53 1b ac 4c a5 ff 3e 5d 08 06 00
0010 08 00 06 04 00 01 ac 4c a5 ff 3e 5d c0 a8 00
0020 00 00 00 00 00 00 c0 a8 00 82 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00

< >

Wi-Fi 3: <live capture in progress> | Packets: 832 · Displayed: 832 (100.0%) | Profile: Default

```
Administrator: Command Prompt
> arp -a .... Displays the arp table.

C:\Windows\system32>arp -a

Interface: 192.168.0.130 --- 0x2
Internet Address      Physical Address      Type
192.168.0.1           ac-4c-a5-ff-3e-5d     dynamic
192.168.0.12          b0-f7-c4-9b-0b-b3     dynamic
192.168.0.46          08-84-9d-c4-4a-cb     dynamic
192.168.0.85          8c-49-62-b6-a4-6e     dynamic
192.168.0.89          2c-54-91-e0-b7-0b     dynamic
192.168.0.188         2c-64-1f-28-19-06     dynamic
192.168.0.255         ff-ff-ff-ff-ff-ff     static
224.0.0.2             01-00-5e-00-00-02     static
224.0.0.22           01-00-5e-00-00-16     static
224.0.0.251          01-00-5e-00-00-fb     static
224.0.0.252          01-00-5e-00-00-fc     static
239.255.3.22         01-00-5e-7f-03-16     static
239.255.255.250      01-00-5e-7f-ff-fa     static
239.255.255.251      01-00-5e-7f-ff-fb     static
255.255.255.255      ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x10
Internet Address      Physical Address      Type
192.168.56.255        ff-ff-ff-ff-ff-ff     static
224.0.0.2             01-00-5e-00-00-02     static
224.0.0.22           01-00-5e-00-00-16     static
224.0.0.251          01-00-5e-00-00-fb     static
224.0.0.252          01-00-5e-00-00-fc     static
239.255.3.22         01-00-5e-7f-03-16     static
239.255.255.250      01-00-5e-7f-ff-fa     static
239.255.255.251      01-00-5e-7f-ff-fb     static

C:\Windows\system32>arp -d

C:\Windows\system32>arp -a

Interface: 192.168.0.130 --- 0x2
Internet Address      Physical Address      Type
192.168.0.1           ac-4c-a5-ff-3e-5d     dynamic
192.168.0.46          08-84-9d-c4-4a-cb     dynamic
192.168.0.85          8c-49-62-b6-a4-6e     dynamic
192.168.0.188         2c-64-1f-28-19-06     dynamic
224.0.0.2             01-00-5e-00-00-02     static
224.0.0.22           01-00-5e-00-00-16     static
224.0.0.251          01-00-5e-00-00-fb     static
224.0.0.252          01-00-5e-00-00-fc     static
239.255.3.22         01-00-5e-7f-03-16     static
239.255.255.250      01-00-5e-7f-ff-fa     static
239.255.255.251      01-00-5e-7f-ff-fb     static

Interface: 192.168.56.1 --- 0x10
Internet Address      Physical Address      Type
224.0.0.2             01-00-5e-00-00-02     static
224.0.0.22           01-00-5e-00-00-16     static
224.0.0.251          01-00-5e-00-00-fb     static
239.255.3.22         01-00-5e-7f-03-16     static
239.255.255.250      01-00-5e-7f-ff-fa     static
239.255.255.251      01-00-5e-7f-ff-fb     static

C:\Windows\system32>sudo arp -d 192.168.0.1 arp -a
```

```
228 1036.434839 Ring_33:8b:b4 Broadcast ARP 60 Who has 192.168.0.1? Tell 192.168.0.92
```

Address before arp -d



Address no longer available after arp -d



Step 2

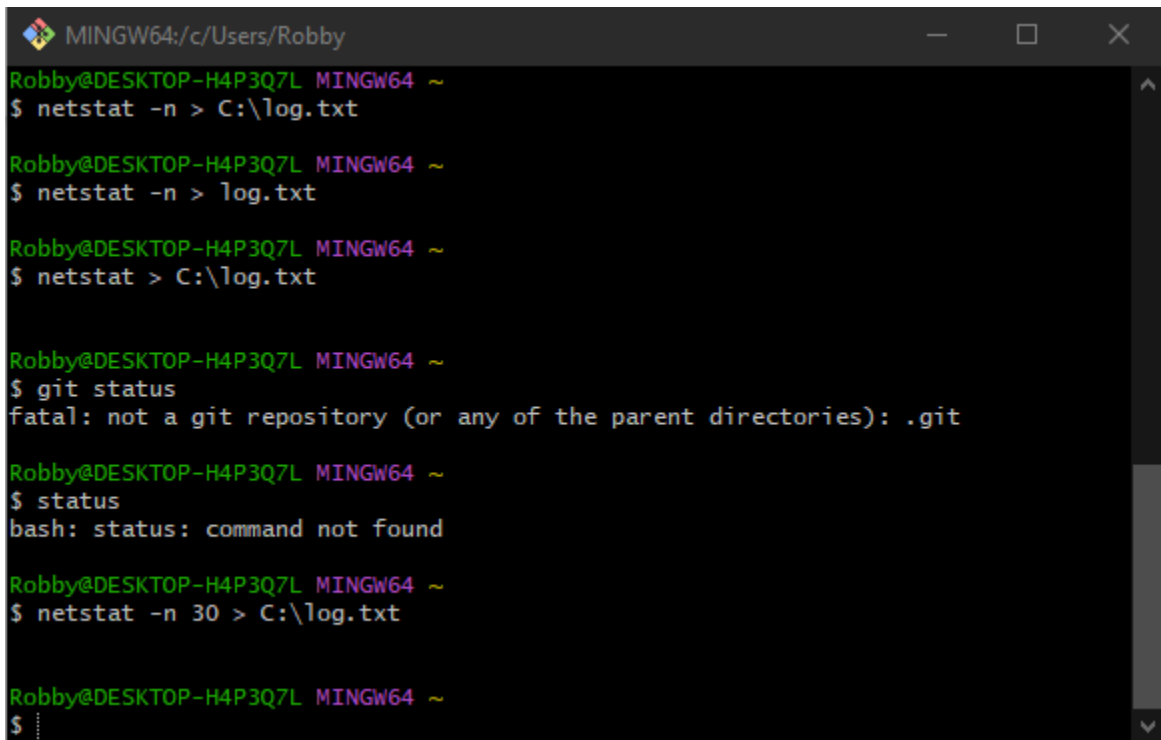
```
> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{06C3E17F-78B4-4E12-9529-...}
> Ethernet II, Src: Technico_ff:3e:5d (ac:4c:a5:ff:3e:5d), Dst: TP-Link_5c:53:1b (54:af:97:5c:53:1b)
  Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Technico_ff:3e:5d (ac:4c:a5:ff:3e:5d)
    Sender IP address: 192.168.0.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.0.130

> Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{06C3E17F-78B4-4E12-9529-...}
> Ethernet II, Src: TP-Link_5c:53:1b (54:af:97:5c:53:1b), Dst: Technico_ff:3e:5d (ac:4c:a5:ff:3e:5d)
  Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: TP-Link_5c:53:1b (54:af:97:5c:53:1b)
    Sender IP address: 192.168.0.130
    Target MAC address: Technico_ff:3e:5d (ac:4c:a5:ff:3e:5d)
    Target IP address: 192.168.0.1
```

Step 3

1. for a request opcode 1. A reply is opcode 2
2. according to <https://kevincurran.org/com320/labs/wireshark/lab-arp.pdf> , the ARP header contains 28 bytes for both the header and the reply.
3. 00:00:00_00:00:00
4. The hex value for upper level protocol is (0x0806)

Task 1.2



```
MINGW64:/c/Users/Robby
Robby@DESKTOP-H4P3Q7L MINGW64 ~
$ netstat -n > C:\log.txt

Robby@DESKTOP-H4P3Q7L MINGW64 ~
$ netstat -n > log.txt

Robby@DESKTOP-H4P3Q7L MINGW64 ~
$ netstat > C:\log.txt

Robby@DESKTOP-H4P3Q7L MINGW64 ~
$ git status
fatal: not a git repository (or any of the parent directories): .git

Robby@DESKTOP-H4P3Q7L MINGW64 ~
$ status
bash: status: command not found

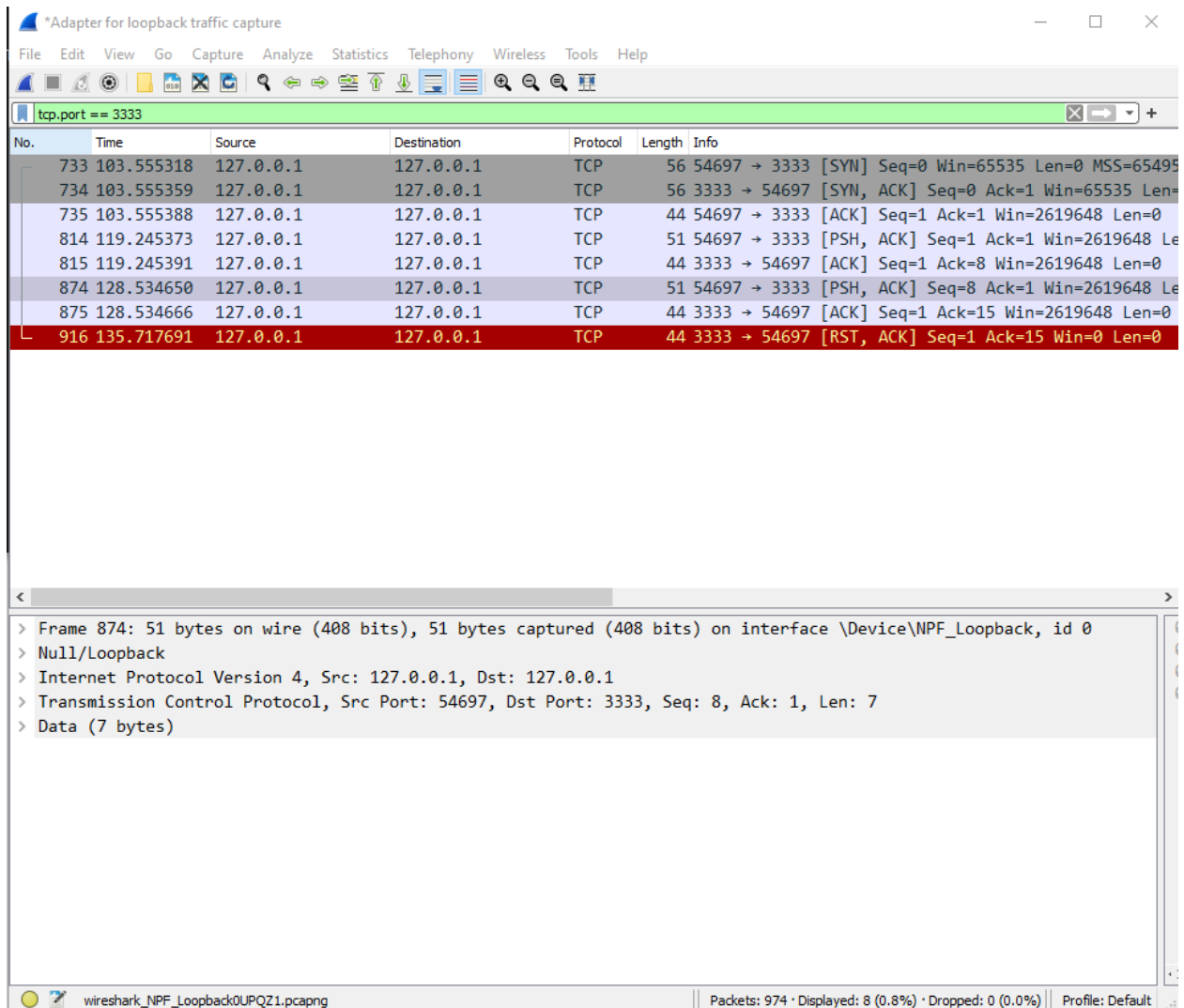
Robby@DESKTOP-H4P3Q7L MINGW64 ~
$ netstat -n 30 > C:\log.txt

Robby@DESKTOP-H4P3Q7L MINGW64 ~
$
```

I wasn't able to run the watch command or the grep command. So I decided for the sake of finishing the assignment to run the above command and set a timer for 10 minutes. The extracted txt file I attempted to import into excel, but it didn't import correctly and I was unable to create a graph from the results. I will also be attaching the txt file to the submission on canvas.

Task 1.3

Step 1:



*Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 3333

No.	Time	Source	Destination	Protocol	Length	Info
733	103.555318	127.0.0.1	127.0.0.1	TCP	56	54697 → 3333 [SYN] Seq=0 Win=65535 Len=0 MSS=65495
734	103.555359	127.0.0.1	127.0.0.1	TCP	56	3333 → 54697 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
735	103.555388	127.0.0.1	127.0.0.1	TCP	44	54697 → 3333 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
814	119.245373	127.0.0.1	127.0.0.1	TCP	51	54697 → 3333 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=0
815	119.245391	127.0.0.1	127.0.0.1	TCP	44	3333 → 54697 [ACK] Seq=1 Ack=8 Win=2619648 Len=0
874	128.534650	127.0.0.1	127.0.0.1	TCP	51	54697 → 3333 [PSH, ACK] Seq=8 Ack=1 Win=2619648 Len=0
875	128.534666	127.0.0.1	127.0.0.1	TCP	44	3333 → 54697 [ACK] Seq=1 Ack=15 Win=2619648 Len=0
916	135.717691	127.0.0.1	127.0.0.1	TCP	44	3333 → 54697 [RST, ACK] Seq=1 Ack=15 Win=0 Len=0

< >

> Frame 874: 51 bytes on wire (408 bits), 51 bytes captured (408 bits) on interface \Device\NPF_Loopback, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 54697, Dst Port: 3333, Seq: 8, Ack: 1, Len: 7

> Data (7 bytes)

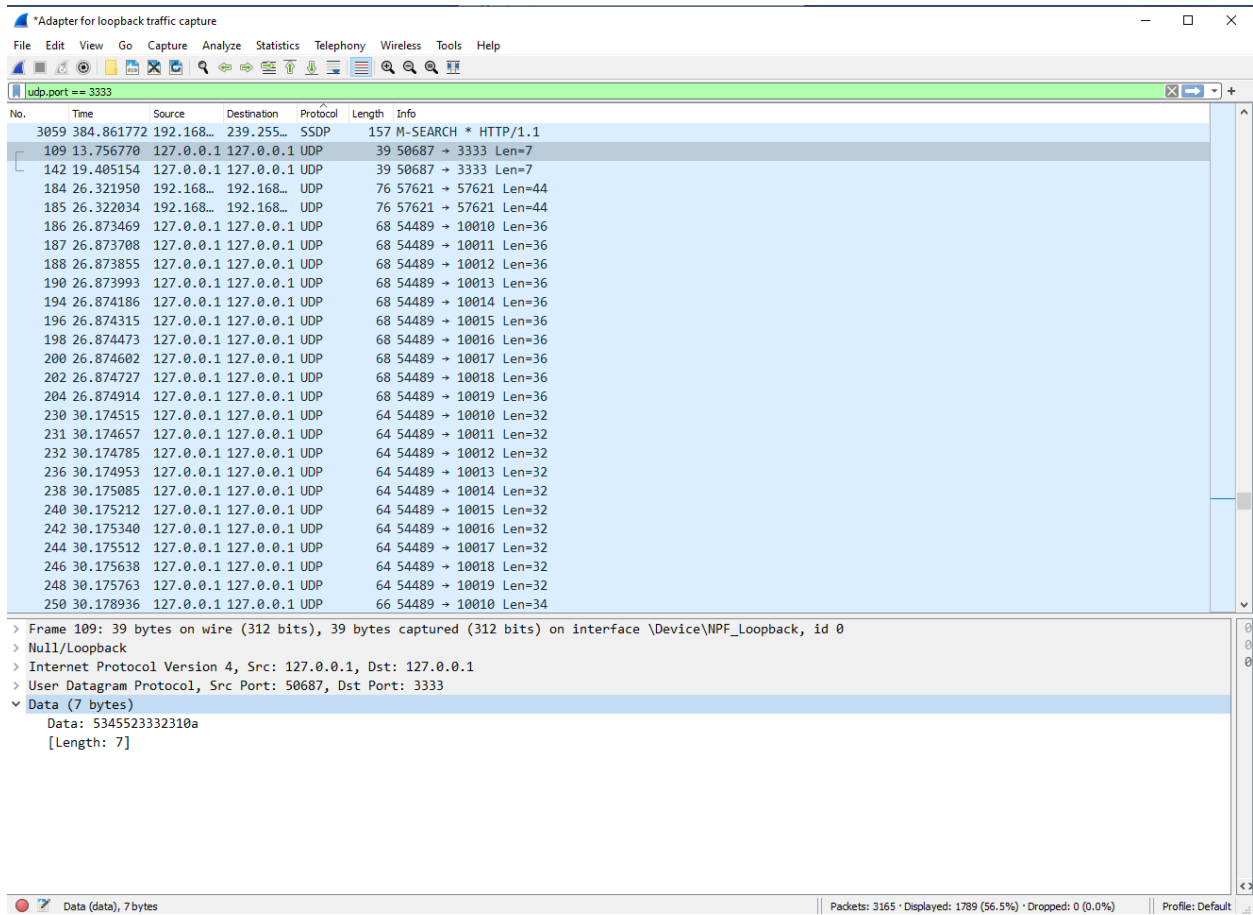
wireshark_NPF_Loopback0UPQZ1.pcapng | Packets: 974 · Displayed: 8 (0.8%) · Dropped: 0 (0.0%) | Profile: Default

Answers:

- Both commands used were given in the assignment document. The first command used was `ncat -k -l 3333`. This sets up the monitoring system for the computer specifically at the port 3333. The `-k` allows it to accept multiple connections in listen mode and the `-l` binds the ncat and listens for incoming connections. The second command used was `ncat 127.0.0.1 3333`. This command has the computer connect to a specific address with a specific port. The port for the 1st and second command are the same. The second command is used to send data that will then be picked up by 1st command.
- According to wireshark, 3 frames. Frames 733,734,735
- Adding up the packet length given of the three frames yields 156 bytes
- According to wireshark, the entire process needed 974 bytes.
- It totaled 14 bytes

- f) Im not sure if I used the wrong data for this one or answer c, but it also shows 974 bytes
- g) We had 14 bytes of data transferred manually, 960 bytes of data was sent automatically.

Step 2:



- a) The commands are very similar to the ones used in the first part however this one uses UDP instead of TCP. I also had to get rid of the -k as it wouldn't allow me to use the command with it.
- b) To be honest, im not really sure. My UDP readings seemed to have more than just my program running through it.
- c) Im not sure
- d) According to wireshark 1536 bytes
- e) Im not sure
- f) If it was the same as before it should be about 14 bytes
- g) Well if my readings are correct(which more than likely they are not) it would mean that we only sent 14 bytes while the system went through about 1500 bytes
- h) It looks like the data that they show is around the same its just structured differently. Udp seems to spread out its information whereas TCP is more consolidated.

Task 1.4

#	Country	Town	Lat	Lon	IP	Hostname	Latency (ms)	DNS Lookups	Distance (km)
1	United States	Mesa	33.4393	-111.77	2600:880C:1:1::1	(None)	3	995	0
2	United States	(Unknown)	37.751	-97.822	2600:880C:1:1::1	(None)	18	22	1349
3	*	*	37.751	-97.822	*	*	0	0	0
4	United States	(Unknown)	37.751	-97.822	2001:578:1:1::1	(None)	11	21	0
5	United States	(Unknown)	37.751	-97.822	2001:578:1:1::1	(None)	24	61	0
6	United States	(Unknown)	37.751	-97.822	2620:11a:1:1::1	(None)	21	71	0
7	United States	(Unknown)	37.751	-97.822	2a04:4e42:1:1::1	(None)	24	16	0
#	Country	Town	Lat	Lon	IP	Hostname	Latency (ms)	DNS Lookups	Distance (km)
1	United States	Los Angeles	34.0544	-118.244	199.232.98.1	(None)	27	10004	0

Answers: it looks like the fastest was the one I did off my home network. When I connected to a public network, it was as if it didn't work or just needed the one IP address to work. The Second one on the public network seemed to have the fewest loops assuming the program worked correctly.

Task 1.5

1.5.1:

The image displays a Windows desktop environment with three application windows:

- Terminal Window (Left):** Running a Java application named `SocketServer`. It shows the server listening on port 8888 and receiving connections from a client. The output includes:


```

      Task :SocketServer
      Server ready for a connection
      Server waiting for a connection
      Received the String SER321
      java.net.SocketException: Connection reset
      at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:320)
      at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:347)
      at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:800)
      at java.base/java.net.Socket$SocketInputStream.read(Socket.java:966)
      at java.base/java.net.Socket$SocketInputStream.read(Socket.java:961)
      at java.base/java.io.ObjectInputStream$BlockDataInputStream.peek(ObjectInputStream.java:2905)
      at java.base/java.io.ObjectInputStream$BlockDataInputStream.peek(ObjectInputStream.java:3232)
      at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.java:3242)
      at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1708)
      at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:538)
      at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:496)
      at SocketServer.main(SocketServer.java:50)
      
```
- Terminal Window (Right):** Running a Java application named `SocketClient`. It shows the client attempting to connect to the server on port 8888. The output includes:


```

      Task :SocketClient
      Please enter a String to send to the Server (enter "exit" to quit)":
      <-----> 75% EXECUTING [14s]
      <-----> 75% EXECUTING [28s]uit":
      java.util.InputMismatchException[55]
      at java.base/java.util.Scanner.throwFor(Scanner.java:943)
      at java.base/java.util.Scanner.next(Scanner.java:1598)
      at java.base/java.util.Scanner.nextInt(Scanner.java:2263)
      at java.base/java.util.Scanner.nextInt(Scanner.java:2217)
      at SocketClient.main(SocketClient.java:42)
      
```
- Wireshark Window:** Capturing network traffic on the `tcp.port == 8888` filter. The packet list shows a SYN packet from `127.0.0.1` to `127.0.0.1` on port 8888. The packet details pane shows the frame structure:
 - Frame 389: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface `\Device\NPF_{...}`, id 0
 - Null/Loopback
 - Internet Protocol Version 4, Src: `127.0.0.1`, Dst: `127.0.0.1`
 - Transmission Control Protocol, Src Port: 3623, Dst Port: 8888, Seq: 1, Ack: 1, Len: 4
 - Data (4 bytes): `aced0005`

1.5.2: As far as gradle goes, I did have to make changes to the client host as I had to change the host from local to the AWS server IP address.

1.5.3: it would not work automatically. You would first need to make sure that the AWS client class has the correct information in order to access the local Server. this would include the correct server port and host.

1.5.4: in order to access the local server from an outside client AWS class, the AWS client would need the local IP address of the server at the very least to be able to connect to the modem running the server. the client would also then need to have the correct access information including the host and the socket number. Otherwise it would not work.

<https://drive.google.com/file/d/1zDBTFkSqz3HvPAX2wfkY4gleiCBTQ4Wp/view?usp=share> link