Solutions 1: Wireshark, Ping, and Traceroute

The HiddenLife of Networks: Network Measurements

Deadline: 27/10/2023
Professor: Maria Papadopouli
TA: Katerina Lionta
mailing list: hy335a-list@csd.uoc.gr
TA: klionta@csd.uoc.gr

The objective of this assignment is to get familiar with passive network measurements, examine the format of packets, and start becoming more familiar with various network protocols (e.g., UDP, ICMP) that we have been discussing in class. Specifically, we will experiment with Wireshark, ping, traceroute, and examine packets, their IP addresses, size and estimate simple statistics.

You will employ **Wireshark**, a popular open-source packet capture and network analysis tool, widely used by network administrators, security professionals, and developers to analyze, troubleshoot, and capture network traffic.

Part 1: Wireshark (50pts)

Objective:

1.1 Open the Wireshark and choose the Wi-Fi interface (if you are connected with an Ethernet cable choose the interface that you observe traffic). Visit some websites like:

https://edition.cnn.com/
https://www.bbc.com/

Wait 5 minutes to ensure you have captured a sufficient number of packets.

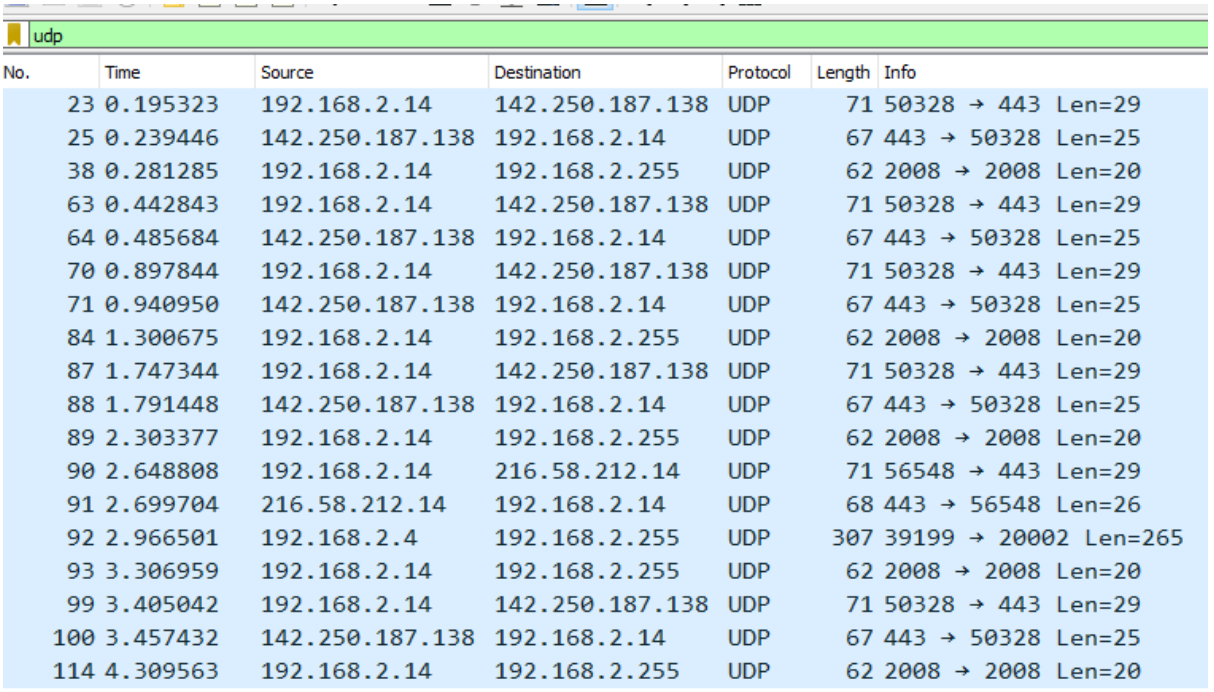
a. How many packets have you captured? (5 pts)

I captured 118 packets.

110	4.106840	192.168.2.14	52.85.158.54	TCP	55 56697 → 443 [ACK] Seq=1 Ack=1 Win=514 Len=1 [TCP segment of a reassembled PDU]
111	4.175945	192.168.2.14	104.16.87.20	TCP	55 56721 → 443 [ACK] Seq=1 Ack=1 Win=514 Len=1 [TCP segment of a reassembled PDU]
112	4.196344	52.85.158.54	192.168.2.14	TCP	66 443 → 56697 [ACK] Seq=1 Ack=2 Win=133 Len=0 SLE=1 SRE=2
113	4.199131	104.16.87.20	192.168.2.14	TCP	66 443 → 56721 [ACK] Seq=1 Ack=2 Win=8 Len=0 SLE=1 SRE=2
114	4.309563	192.168.2.14	192.168.2.255	UDP	62 2008 → 2008 Len=20
115	4.323048	192.168.2.14	146.75.3.5	TCP	55 56698 → 443 [ACK] Seq=1 Ack=1 Win=513 Len=1 [TCP segment of a reassembled PDU]
116	4.354069	146.75.3.5	192.168.2.14	TCP	66 443 → 56698 [ACK] Seq=1 Ack=2 Win=290 Len=0 SLE=1 SRE=2
117	4.438913	192.168.2.14	52.85.158.86	TCP	55 56726 → 443 [ACK] Seq=1 Ack=1 Win=511 Len=1 [TCP segment of a reassembled PDU]
118	4.463048	52.85.158.86	192.168.2.14	TCP	66 443 → 56726 [ACK] Seq=1 Ack=2 Win=133 Len=0 SLE=1 SRE=2

b. ***UDP*** *(User Datagram Protocol) is a connectionless transport layer protocol in the Internet Protocol (IP). UDP does not establish a connection before sending data and does not guarantee delivery or order of packets. It is often used for real-time applications, such as streaming media and online gaming, where low latency and quick data transmission are more critical than ensuring every piece of data arrives intact.*

How many **UDP** packets have you captured? (5pts)



udp

No.	Time	Source	Destination	Protocol	Length	Info
23	0.195323	192.168.2.14	142.250.187.138	UDP	71	50328 → 443 Len=29
25	0.239446	142.250.187.138	192.168.2.14	UDP	67	443 → 50328 Len=25
38	0.281285	192.168.2.14	192.168.2.255	UDP	62	2008 → 2008 Len=20
63	0.442843	192.168.2.14	142.250.187.138	UDP	71	50328 → 443 Len=29
64	0.485684	142.250.187.138	192.168.2.14	UDP	67	443 → 50328 Len=25
70	0.897844	192.168.2.14	142.250.187.138	UDP	71	50328 → 443 Len=29
71	0.940950	142.250.187.138	192.168.2.14	UDP	67	443 → 50328 Len=25
84	1.300675	192.168.2.14	192.168.2.255	UDP	62	2008 → 2008 Len=20
87	1.747344	192.168.2.14	142.250.187.138	UDP	71	50328 → 443 Len=29
88	1.791448	142.250.187.138	192.168.2.14	UDP	67	443 → 50328 Len=25
89	2.303377	192.168.2.14	192.168.2.255	UDP	62	2008 → 2008 Len=20
90	2.648808	192.168.2.14	216.58.212.14	UDP	71	56548 → 443 Len=29
91	2.699704	216.58.212.14	192.168.2.14	UDP	68	443 → 56548 Len=26
92	2.966501	192.168.2.4	192.168.2.255	UDP	307	39199 → 20002 Len=265
93	3.306959	192.168.2.14	192.168.2.255	UDP	62	2008 → 2008 Len=20
99	3.405042	192.168.2.14	142.250.187.138	UDP	71	50328 → 443 Len=29
100	3.457432	142.250.187.138	192.168.2.14	UDP	67	443 → 50328 Len=25
114	4.309563	192.168.2.14	192.168.2.255	UDP	62	2008 → 2008 Len=20

c. *An* ***IP (Internet Protocol)*** *address is a numerical label assigned to each device participating in a computer network that uses the Internet Protocol for communication. An IPv4 address is typically represented as a series of four numbers separated by dots (a.b.c.d). Each number is called an octet and represents 8 bits.*

Which is your **IP address**? How many packets were sent from your IP? How many packets were received by your IP? (10pts)

Include screenshots in your report and export the results of the Wireshark to a CSV file (File-> Export Packet Dissections-> As CSV…).

To find your IP address, in the following platforms:

- **Windows**: Open the command prompt type `ipconfig /all` and find the IPv4 Address field
- **Linux**: Open the terminal and type `hostname -I` and your IP address will be displayed just below the command.

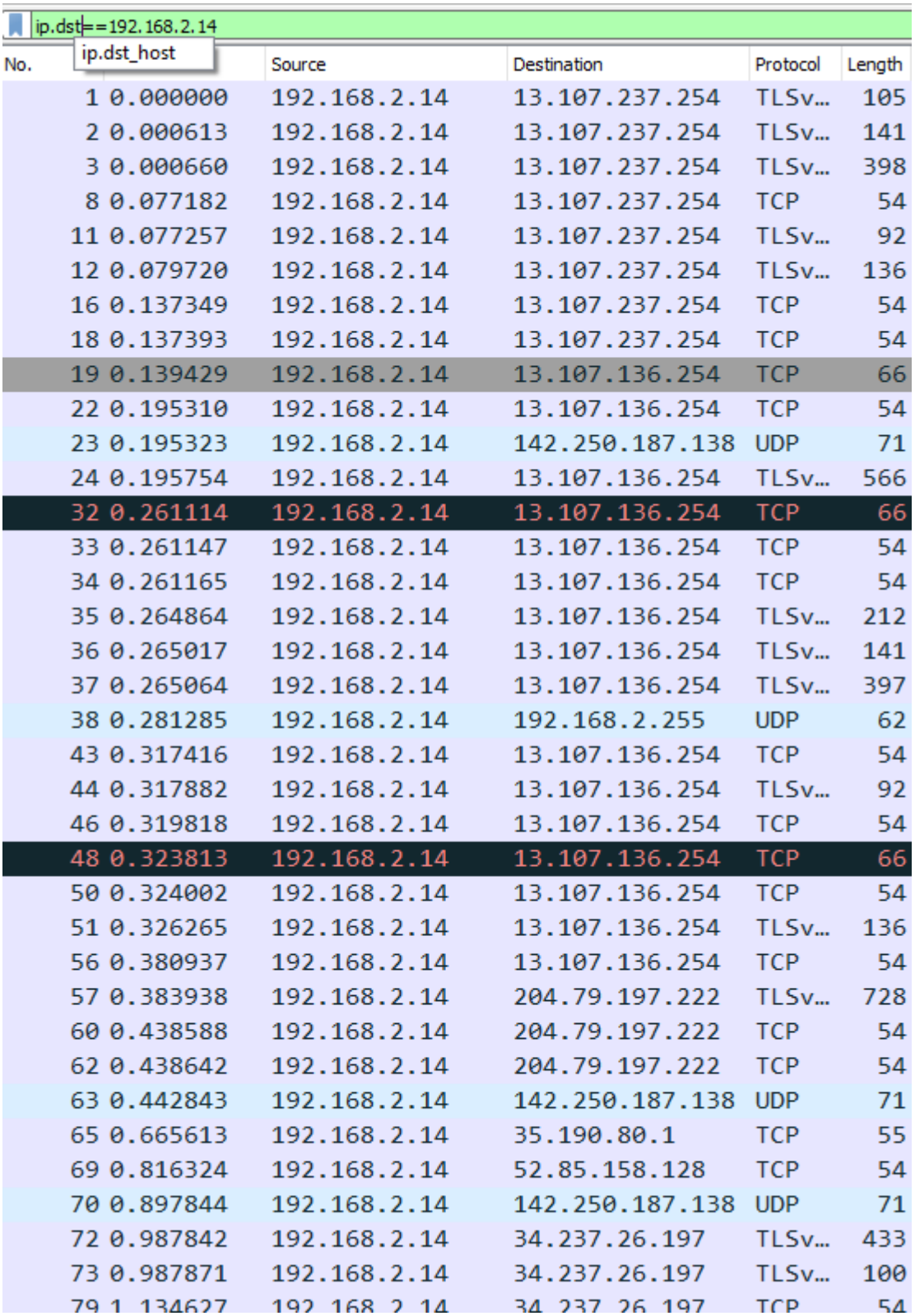
```
IPv4 Address. . . . . . . . . . . : 192.168.2.14(Preferred)
```

To find how many packets were sent from my IP address I filtered with the command ip.src==192.168.2.14.

ip.src==192.168.2.14

No.	Time	Source	Destination	Protocol	Length
36	0.265017	192.168.2.14	13.107.136.254	TLSv...	141
37	0.265064	192.168.2.14	13.107.136.254	TLSv...	397
38	0.281285	192.168.2.14	192.168.2.255	UDP	62
43	0.317416	192.168.2.14	13.107.136.254	TCP	54
44	0.317882	192.168.2.14	13.107.136.254	TLSv...	92
46	0.319818	192.168.2.14	13.107.136.254	TCP	54
48	0.323813	192.168.2.14	13.107.136.254	TCP	66
50	0.324002	192.168.2.14	13.107.136.254	TCP	54
51	0.326265	192.168.2.14	13.107.136.254	TLSv...	136
56	0.380937	192.168.2.14	13.107.136.254	TCP	54
57	0.383938	192.168.2.14	204.79.197.222	TLSv...	728
60	0.438588	192.168.2.14	204.79.197.222	TCP	54
62	0.438642	192.168.2.14	204.79.197.222	TCP	54
63	0.442843	192.168.2.14	142.250.187.138	UDP	71
65	0.665613	192.168.2.14	35.190.80.1	TCP	55
69	0.816324	192.168.2.14	52.85.158.128	TCP	54
70	0.897844	192.168.2.14	142.250.187.138	UDP	71
72	0.987842	192.168.2.14	34.237.26.197	TLSv...	433
73	0.987871	192.168.2.14	34.237.26.197	TLSv...	100
79	1.134627	192.168.2.14	34.237.26.197	TCP	54
80	1.136474	192.168.2.14	34.237.26.197	TLSv...	96
81	1.215615	192.168.2.14	18.168.235.151	TLSv...	108
84	1.300675	192.168.2.14	192.168.2.255	UDP	62
86	1.345941	192.168.2.14	18.168.235.151	TCP	54
87	1.747344	192.168.2.14	142.250.187.138	UDP	71
89	2.303377	192.168.2.14	192.168.2.255	UDP	62
90	2.648808	192.168.2.14	216.58.212.14	UDP	71
93	3.306959	192.168.2.14	192.168.2.255	UDP	62
99	3.405042	192.168.2.14	142.250.187.138	UDP	71
102	3.483484	192.168.2.14	63.140.62.164	TCP	54
105	3.558557	192.168.2.14	63.140.62.164	TCP	54
106	3.806000	192.168.2.14	146.75.3.5	TCP	55
108	4.006720	192.168.2.14	224.0.0.252	IGMP...	46
109	4.006799	192.168.2.14	239.255.255.250	IGMP...	46
110	4.106840	192.168.2.14	52.85.158.54	TCP	55

To find how many packets were received by my IP I filtered with the command ip.dst==192..168.2.14.



No.		Source	Destination	Protocol	Length
1	0.000000	192.168.2.14	13.107.237.254	TLSv…	105
2	0.000613	192.168.2.14	13.107.237.254	TLSv…	141
3	0.000660	192.168.2.14	13.107.237.254	TLSv…	398
8	0.077182	192.168.2.14	13.107.237.254	TCP	54
11	0.077257	192.168.2.14	13.107.237.254	TLSv…	92
12	0.079720	192.168.2.14	13.107.237.254	TLSv…	136
16	0.137349	192.168.2.14	13.107.237.254	TCP	54
18	0.137393	192.168.2.14	13.107.237.254	TCP	54
19	0.139429	192.168.2.14	13.107.136.254	TCP	66
22	0.195310	192.168.2.14	13.107.136.254	TCP	54
23	0.195323	192.168.2.14	142.250.187.138	UDP	71
24	0.195754	192.168.2.14	13.107.136.254	TLSv…	566
32	0.261114	192.168.2.14	13.107.136.254	TCP	66
33	0.261147	192.168.2.14	13.107.136.254	TCP	54
34	0.261165	192.168.2.14	13.107.136.254	TCP	54
35	0.264864	192.168.2.14	13.107.136.254	TLSv…	212
36	0.265017	192.168.2.14	13.107.136.254	TLSv…	141
37	0.265064	192.168.2.14	13.107.136.254	TLSv…	397
38	0.281285	192.168.2.14	192.168.2.255	UDP	62
43	0.317416	192.168.2.14	13.107.136.254	TCP	54
44	0.317882	192.168.2.14	13.107.136.254	TLSv…	92
46	0.319818	192.168.2.14	13.107.136.254	TCP	54
48	0.323813	192.168.2.14	13.107.136.254	TCP	66
50	0.324002	192.168.2.14	13.107.136.254	TCP	54
51	0.326265	192.168.2.14	13.107.136.254	TLSv…	136
56	0.380937	192.168.2.14	13.107.136.254	TCP	54
57	0.383938	192.168.2.14	204.79.197.222	TLSv…	728
60	0.438588	192.168.2.14	204.79.197.222	TCP	54
62	0.438642	192.168.2.14	204.79.197.222	TCP	54
63	0.442843	192.168.2.14	142.250.187.138	UDP	71
65	0.665613	192.168.2.14	35.190.80.1	TCP	55
69	0.816324	192.168.2.14	52.85.158.128	TCP	54
70	0.897844	192.168.2.14	142.250.187.138	UDP	71
72	0.987842	192.168.2.14	34.237.26.197	TLSv…	433
73	0.987871	192.168.2.14	34.237.26.197	TLSv…	100
79	1.134627	192.168.2.14	34.237.26.197	TCP	54

1.2 The two datasets you will use are in this drive folder datasets_CS335a. These are two output files of specific Wireshark runs in CSV format. The fields of the datasets are:

No.: the serial number of the packet
Time: the time of the transmission/receiving of the packet (starts from 0, the moment that the capturing started) in seconds
Source: the source IP address
Destination: the destination IP address
Protocol: the protocol used
Length: the length of the packets in bytes
Info: extra information about the packet (header fields, flags etc)

Use the *pandas library* to open and process the CSV file (example: https://www.geeksforgeeks.org/convert-csv-to-pandas-dataframe/).

Develop a script that accepts a CSV file name as an input through a command line argument.

The script should load the data from the CSV file into a DataFrame. (5pts)

1.3 Then generate various plots based on this data. Use the Matplotlib library (https://matplotlib.org/stable/gallery/index) to illustrate your data. Generate the following plots for the two datasets (DATASE1, DATASET2):

a. Create a bar plot that illustrates the top 2 destination IP addresses based on the number of packets they have received. This means you will generate one bar for each of the two IP addresses that received the highest number of packets. (5pts)

b. Generate a pie chart that visually represents the distribution of protocols(except UDP) used for sending the packets. This pie chart should display the percentage of the different protocols employed in the packet communication. (5pts)

c. Produce a bar plot that visually represents the number of packets received by the IP address 63.111.11.187 per second (only the seconds that the IP receives packets included in the plot). This bar plot should show how the packet count for this specific IP address varies over time in terms of seconds. (5pts)

d. Generate a histogram that visually represents the distribution of packet lengths. Which is the packet size of the most packets (find the median)? (10pts)

Include in your deliverable the script and include in your report the plots that you generate from the script.
Don't hesitate to utilize any resources that you find helpful. You are not obligated to use the suggested libraries.

Part 2: Ping (25 pts)

The **ping** command is a network utility used to test the reachability of a host on an Internet Protocol (IP) network and to measure the round-trip time for messages sent from the source to the destination. It sends a series of **ICMP** (Internet Control Message Protocol) Echo Request messages to the target host and reports the time it takes for each message to be acknowledged.
The Internet Control Message Protocol (**ICMP**) is a network layer protocol within the Internet Protocol (IP) suite. It is responsible for sending error messages, diagnostics, and operational information between network devices. ICMP packets are encapsulated within IP packets and serve various purposes, including error reporting, network testing

2.1 Explain precisely and briefly what is the **Round Trip Time (RTT)**. (5pts)

Is the duration of time it takes for a data packet to travel from the sender to the receiver and back again.

2.2 Use the ping command to measure the RTTs in the following 3 hosts:

www.google.com
www.youtube.com
www.facebook.com
www.twitter.com

For the packets from which you received a response, estimate and report the minimum, average, and maximum round trip times in milliseconds. (10pts)

Include screenshots of the answers that you received in your report.

```
C:\WINDOWS\system32>ping www.google.com

Pinging www.google.com [142.250.187.100] with 32 bytes of data:
Reply from 142.250.187.100: bytes=32 time=38ms TTL=111
Reply from 142.250.187.100: bytes=32 time=33ms TTL=111
Reply from 142.250.187.100: bytes=32 time=31ms TTL=111
Reply from 142.250.187.100: bytes=32 time=31ms TTL=111

Ping statistics for 142.250.187.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 31ms, Maximum = 38ms, Average = 33ms
```

```
C:\WINDOWS\system32>ping www.youtube.com

Pinging youtube-ui.l.google.com [216.58.212.14] with 32 bytes of data:
Reply from 216.58.212.14: bytes=32 time=36ms TTL=111
Reply from 216.58.212.14: bytes=32 time=36ms TTL=111
Reply from 216.58.212.14: bytes=32 time=36ms TTL=111
Reply from 216.58.212.14: bytes=32 time=35ms TTL=111

Ping statistics for 216.58.212.14:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 35ms, Maximum = 36ms, Average = 35ms
```

```
C:\WINDOWS\system32>ping www.facebook.com

Pinging star-mini.c10r.facebook.com [157.240.9.35] with 32 bytes of data:
Reply from 157.240.9.35: bytes=32 time=36ms TTL=49
Reply from 157.240.9.35: bytes=32 time=36ms TTL=49
Reply from 157.240.9.35: bytes=32 time=38ms TTL=49
Reply from 157.240.9.35: bytes=32 time=37ms TTL=49

Ping statistics for 157.240.9.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 36ms, Maximum = 38ms, Average = 36ms
```

```
C:\WINDOWS\system32>ping www.twitter.com

Pinging twitter.com [104.244.42.129] with 32 bytes of data:
Reply from 104.244.42.129: bytes=32 time=60ms TTL=49
Reply from 104.244.42.129: bytes=32 time=71ms TTL=49
Reply from 104.244.42.129: bytes=32 time=62ms TTL=49
Reply from 104.244.42.129: bytes=32 time=61ms TTL=49

Ping statistics for 104.244.42.129:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 60ms, Maximum = 71ms, Average = 63ms
```

Hosts	Min RTT(ms)	Max RTT(ms)	Average RTT(ms)
Google (142.250.187.100)	31	38	33
Youtube (216.58.212.14)	35	36	35
Facebook (157.240.9.35)	36	38	36
Twitter (104.244.42.129)	60	71	63

2.3 By default the Ping command sends 4 packets of 64 bytes [56 bytes(payload) + 8 bytes (header of the ICMP packet)] to the destination.

Run in the terminal the command `ping -h` to find the suitable flag in order to specify the packet size. Now send pings with 56, 512, and 1024 byte packets to the 4 hosts above. Estimate and report the minimum, average, and maximum round trip times in milliseconds for each of the 12 pings. Why are the minimum round-trip times to the same hosts different when using 56, 512, and 1024 byte packets? (10pts)

Hosts	56 bytes	512 bytes	1024-bytes
Google	35ms	37ms	39ms
Youtube	32ms	34ms	37ms
Facebook	36ms	38ms	43ms
Twitter	60ms	62ms	63ms

As the bits of the packet increase, so does the RTT. More bits, more processing, more delay.

```
C:\WINDOWS\system32>ping -l 56 google.com

Pinging google.com [142.250.187.110] with 56 bytes of data:
Reply from 142.250.187.110: bytes=56 time=37ms TTL=111
Reply from 142.250.187.110: bytes=56 time=36ms TTL=111
Reply from 142.250.187.110: bytes=56 time=35ms TTL=111
Reply from 142.250.187.110: bytes=56 time=43ms TTL=111

Ping statistics for 142.250.187.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 35ms, Maximum = 43ms, Average = 37ms
```

```
C:\WINDOWS\system32>ping -l 512 google.com

Pinging google.com [142.250.187.110] with 512 bytes of data:
Reply from 142.250.187.110: bytes=68 (sent 512) time=38ms TTL=111
Reply from 142.250.187.110: bytes=68 (sent 512) time=37ms TTL=111
Reply from 142.250.187.110: bytes=68 (sent 512) time=44ms TTL=111
Reply from 142.250.187.110: bytes=68 (sent 512) time=49ms TTL=111

Ping statistics for 142.250.187.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 37ms, Maximum = 49ms, Average = 42ms
```

```
C:\WINDOWS\system32>ping -l 1024 google.com

Pinging google.com [142.250.187.110] with 1024 bytes of data:
Reply from 142.250.187.110: bytes=68 (sent 1024) time=50ms TTL=111
Reply from 142.250.187.110: bytes=68 (sent 1024) time=39ms TTL=111
Reply from 142.250.187.110: bytes=68 (sent 1024) time=51ms TTL=111
Reply from 142.250.187.110: bytes=68 (sent 1024) time=40ms TTL=111

Ping statistics for 142.250.187.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 39ms, Maximum = 51ms, Average = 45ms
```

```
C:\WINDOWS\system32>ping -l 56 facebook.com

Pinging facebook.com [157.240.9.35] with 56 bytes of data:
Reply from 157.240.9.35: bytes=56 time=36ms TTL=49
Reply from 157.240.9.35: bytes=56 time=36ms TTL=49
Reply from 157.240.9.35: bytes=56 time=37ms TTL=49
Reply from 157.240.9.35: bytes=56 time=39ms TTL=49

Ping statistics for 157.240.9.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 36ms, Maximum = 39ms, Average = 37ms
```

```
C:\WINDOWS\system32>ping -l 512 facebook.com

Pinging facebook.com [157.240.9.35] with 512 bytes of data:
Reply from 157.240.9.35: bytes=512 time=38ms TTL=49
Reply from 157.240.9.35: bytes=512 time=40ms TTL=49
Reply from 157.240.9.35: bytes=512 time=38ms TTL=49
Reply from 157.240.9.35: bytes=512 time=46ms TTL=49

Ping statistics for 157.240.9.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 38ms, Maximum = 46ms, Average = 40ms
```

```
C:\WINDOWS\system32>ping -l 1024 facebook.com

Pinging facebook.com [157.240.9.35] with 1024 bytes of data:
Reply from 157.240.9.35: bytes=1024 time=120ms TTL=49
Reply from 157.240.9.35: bytes=1024 time=50ms TTL=49
Reply from 157.240.9.35: bytes=1024 time=43ms TTL=49
Reply from 157.240.9.35: bytes=1024 time=45ms TTL=49

Ping statistics for 157.240.9.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 43ms, Maximum = 120ms, Average = 64ms
```

```
C:\WINDOWS\system32>ping -l 56 youtube.com

Pinging youtube.com [142.250.184.142] with 56 bytes of data:
Reply from 142.250.184.142: bytes=56 time=32ms TTL=111
Reply from 142.250.184.142: bytes=56 time=34ms TTL=111
Reply from 142.250.184.142: bytes=56 time=32ms TTL=111
Reply from 142.250.184.142: bytes=56 time=33ms TTL=111

Ping statistics for 142.250.184.142:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 32ms, Maximum = 34ms, Average = 32ms
```

```
C:\WINDOWS\system32>ping -l 512 youtube.com

Pinging youtube.com [142.250.184.142] with 512 bytes of data:
Reply from 142.250.184.142: bytes=68 (sent 512) time=37ms TTL=111
Reply from 142.250.184.142: bytes=68 (sent 512) time=35ms TTL=111
Reply from 142.250.184.142: bytes=68 (sent 512) time=34ms TTL=111
Reply from 142.250.184.142: bytes=68 (sent 512) time=48ms TTL=111

Ping statistics for 142.250.184.142:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 34ms, Maximum = 48ms, Average = 38ms
```

```
C:\WINDOWS\system32>ping -l 1024 youtube.com

Pinging youtube.com [142.250.184.142] with 1024 bytes of data:
Reply from 142.250.184.142: bytes=68 (sent 1024) time=44ms TTL=111
Reply from 142.250.184.142: bytes=68 (sent 1024) time=37ms TTL=111
Reply from 142.250.184.142: bytes=68 (sent 1024) time=45ms TTL=111
Reply from 142.250.184.142: bytes=68 (sent 1024) time=38ms TTL=111

Ping statistics for 142.250.184.142:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 37ms, Maximum = 45ms, Average = 41ms
```

```
C:\WINDOWS\system32>ping -l 56 twitter.com

Pinging twitter.com [104.244.42.129] with 56 bytes of data:
Reply from 104.244.42.129: bytes=56 time=87ms TTL=49
Reply from 104.244.42.129: bytes=56 time=73ms TTL=49
Reply from 104.244.42.129: bytes=56 time=60ms TTL=49
Reply from 104.244.42.129: bytes=56 time=60ms TTL=49

Ping statistics for 104.244.42.129:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 60ms, Maximum = 87ms, Average = 70ms
```

```
C:\WINDOWS\system32>ping -l 512 twitter.com

Pinging twitter.com [104.244.42.129] with 512 bytes of data:
Reply from 104.244.42.129: bytes=512 time=75ms TTL=49
Reply from 104.244.42.129: bytes=512 time=62ms TTL=49
Reply from 104.244.42.129: bytes=512 time=74ms TTL=49
Reply from 104.244.42.129: bytes=512 time=65ms TTL=49

Ping statistics for 104.244.42.129:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 62ms, Maximum = 75ms, Average = 69ms
```

```
C:\WINDOWS\system32>ping -l 1024 twitter.com

Pinging twitter.com [104.244.42.129] with 1024 bytes of data:
Reply from 104.244.42.129: bytes=1024 time=76ms TTL=49
Reply from 104.244.42.129: bytes=1024 time=67ms TTL=49
Reply from 104.244.42.129: bytes=1024 time=63ms TTL=49
Reply from 104.244.42.129: bytes=1024 time=80ms TTL=49

Ping statistics for 104.244.42.129:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 63ms, Maximum = 80ms, Average = 71ms
```

Part 3: Traceroute (25pts)

The **traceroute** command is a network diagnostic tool that helps identify the route taken by data packets from a source to a destination across an Internet Protocol (IP) network.

3.1 Run:

For Linux:

```
traceroute www.google.com
```

For Windows:

```
tracert    www.google.com
```

What is the number of hops that the packets had to pass through to reach their destination? (5pts)
Include a screenshot of the answer in your report.

```
C:\WINDOWS\system32>tracert    www.google.com

Tracing route to www.google.com [142.250.187.100]
over a maximum of 30 hops:

  1     4 ms    11 ms     3 ms  vodafone.station [192.168.2.1]
  2    15 ms    16 ms    16 ms  loopback2004.med01.dsl.hol.gr [62.38.0.170]
  3    14 ms    14 ms    15 ms  62.38.98.173
  4    21 ms    20 ms    20 ms  10.119.5.233
  5    24 ms    24 ms    25 ms  10.119.9.165
  6    22 ms    22 ms    21 ms  10.119.2.221
  7    24 ms    24 ms    24 ms  10.119.0.105
  8    24 ms    24 ms    23 ms  62.38.97.150
  9     *        *        *     Request timed out.
 10    26 ms    24 ms    25 ms  ae3-100-ucr1.atm.cw.net [195.89.103.89]
 11    37 ms    34 ms    34 ms  ae24-xcr1.sof.cw.net [195.2.16.5]
 12    33 ms    34 ms    34 ms  72.14.218.54
 13    86 ms    34 ms    50 ms  216.239.59.239
 14    33 ms    33 ms    33 ms  142.251.52.81
 15    33 ms    33 ms    34 ms  sof02s44-in-f4.1e100.net [142.250.187.100]

Trace complete.
```

To reach the destination, the packets had to pass 15 hops.

3.2 Can you provide an explanation of how the **Time to live (TTL)** field is utilized by the traceroute command and what issue it aims to resolve? (10pts)

Each packet has a Time-to-Live value, as a packet travels through each hop, its Time-to-Live value decreases by 1. When the TTL value of a packet hits zero, the packet is no longer forwarded. This TTL value feature is in place to ensure that packets don't endlessly roam the Internet.

3.3 What is the minimum TTL required for the packets to reach their destination in 3.1? (5pts)

The Traceroute command has a default TTL setting of 30 for the packets it sends. This means that if a packet cannot reach its destination within 30 hops, it will be rejected. Based on the screenshot provided, we can determine that the packets in question reached their destination after passing at least 15 hops. Therefore, we can conclude that the minimum TTL for each of the three packets in this instance was 30-15=15. A more general answer is:

minTTL = TTL_of_the_packet - number_of_hops_of_the_packet

3.4 What could be the potential reasons for the following output received from the traceroute command (write at least two reasons)? (5pts)

```
2     *        *        *     Request timed out.
3     *        *        *     Request timed out.
4     *        *        *     Request timed out.
5     *        *        *     Request timed out.
```

1. Firewall: configured to not return traceroute replies
2. Network congestion, low priority of ICMP packets