**NETWORK ARCHITECTURE – I**

**HOMEWORK #3**

**Name : Shreyaa Sridhar**

**Student ID : 16251190**

**Email ID :** [ssn2m@mail.umkc.edu](mailto:ssn2m@mail.umkc.edu)

**Question 1**

Consider sending a large file from a host to another over a TCP connection that has no loss.

1. Suppose TCP uses AIMD for its congestion control without slow start. Assuming cwnd increases by 1 MSS every time a batch of ACKs is received and assuming approximately constant RTTs, how long does it take for cwnd increase from 1 MSS to 8 MSS?

**Answer :**

It takes 1 RTT to increase to 2 MSS , 2 RTTs to increase to 3 MSS, 3 RTTs to increase to 4 MSS, 4 RTTs to increase to 5 MSS, 5 RTTs to increase to 6 MSS, 6 RTTs to increase to 7 MSS, 7 RTTs to increase to 8 MSS.

Therefore , it takes **7 RTTs** for cwnd to increase from 1 MSS to 8 MSS.

1. What is the average throughput in terms of MSS and RTT for this connection up through time = 6 RTT (consider 1st 6 RTT after the start of the scenario)?

**Answer :**

In the first RTT, 1 MSS was sent ; in the second RTT, 2 MSS was sent; in the third RTT, 3 MSS was sent; in the fourth RTT , 4 MSS was sent; in the fifth RTT , 5 MSS was sent and in the 6th RTT , 6 MSS was sent.

Thus up to 6 RTT, 1 + 2 + 3 + 4 + 5 + 6 = 21 MSS were sent and acknowledged.

Thus the average throughput upto 6 RTT is

21 MSS / 6 RTT = **3.5 MSS / RTT**

**Question 2**

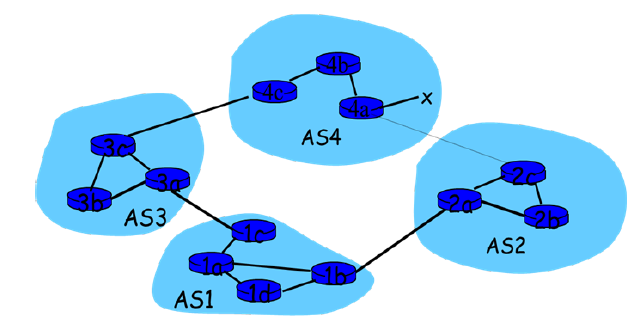
Consider the network shown below. Suppose AS3 and AS2 are running OSPF for their intra-AS routing protocol. Suppose AS1 and AS4 are running RIP for their intra-AS routing protocol. Suppose eBGP and iBGP are used for the inter-AS routing protocol. Initially suppose there is no physical link between AS2 and AS4.

(a) Router 3b learns about prefix x from which routing protocol: OSPF, RIP, eBGP or iBGP?

(b) Router 1c learns about prefix x from which routing protocol?

(c) Router 1b learns about prefix x from which routing protocol?

(d) Router 2a learns about prefix x from which routing protocol?



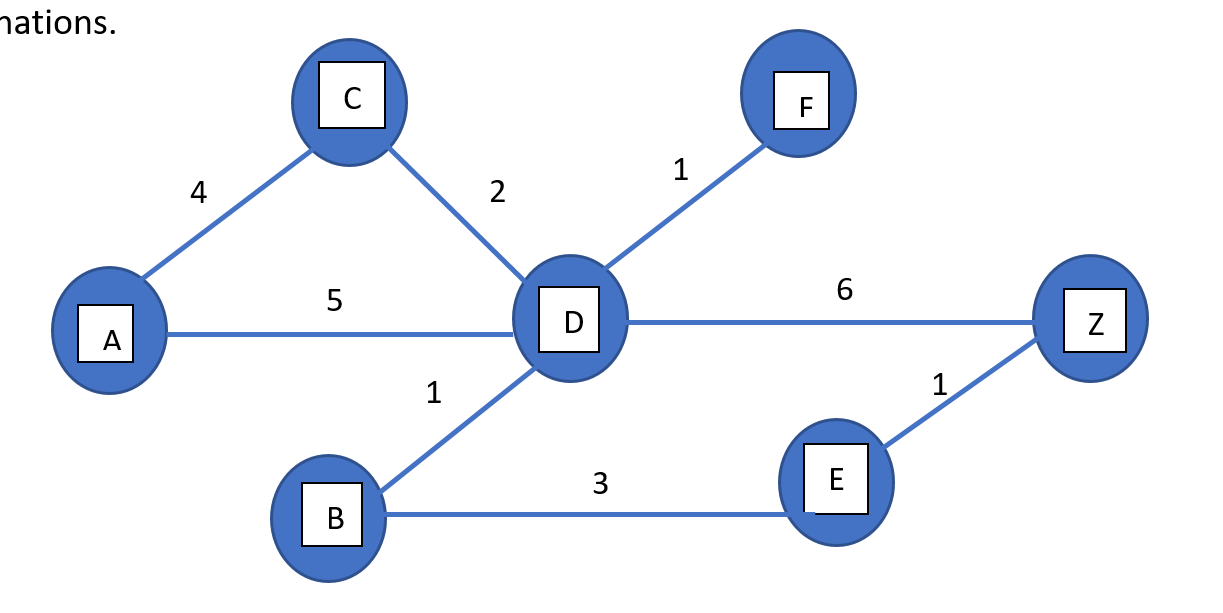
**Answer :**

1. Router 3b learns about prefix x from 3c of its own AS , where 3c learns from 4c of AS4. Since 3b learns from 3c it uses **iBGP routing protocol**.
2. Router 1c learns from 3a of AS3, which in turn learns from 3c within its own AS. Therefore it uses **eGBP routing protocol** to learn about prefix x from 3a.
3. Router 1b learns from 1a or 1d which in turn learns from 1c, therefore it uses **iBGP routing protocol** to learn about prefix x from 1a.
4. Router 2a learns from 1b of AS1 , therefore it uses **eGBP routing protocol** to learn about prefix x from 1b.

**Question 3 :**

Consider the network shown below (the labels are the cost on the links).

Show the operation of Dijkstra’s (Link State) algorithm for computing the shortest path from A to all destinations.



**Answer :**

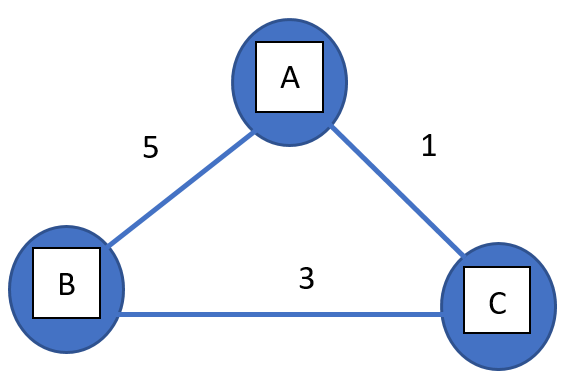
Source is ‘A’

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | N’ | D(b)  P(b) | D(c)  P(c) | D(d)  P(d) | D(e)  P(e) | D(f)  P(f) | D(z)  P(z) |
| 0 | a | ∞ | 4,a | 5,a | ∞ | ∞ | ∞ |
| 1 | ac | ∞ |  | 5,a | ∞ | ∞ | ∞ |
| 2 | acd | 6,d |  |  | ∞ | 6,d | 11,d |
| 3 | acdb |  |  |  | 9,b | 6,d | 11,d |
| 4 | acdbf |  |  |  | 9,b |  | 11,d |
| 5 | acdbfe |  |  |  |  |  | 10,e |
| 6 | acdbfez |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Shortest path diagram** | **Resulting Forwarding Table**   |  |  | | --- | --- | | Destination | Link | | b | (a,d) | | c | (a,c) | | d | (a,d) | | e | (a,d) | | f | (a,d) | | z | (a,d) | |

**Question 4**

Show the distance table for all nodes (A, B and C) that would be computed by the distance vector algorithm.



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Node A | A | B | C | | A | **0** | **5** | **1** | | B | **∞** | **∞** | **∞** | | C | **∞** | **∞** | **∞** | | |  |  |  |  | | --- | --- | --- | --- | | Node A | A | B | C | | A | **0** | **4** | **1** | | B | **5** | **0** | **3** | | C | **1** | **3** | **0** | | |  |  |  |  | | --- | --- | --- | --- | | Node A | A | B | C | | A | **0** | **4** | **1** | | B | **5** | **0** | **3** | | C | **1** | **3** | **0** | | |
| |  |  |  |  | | --- | --- | --- | --- | | Node B | A | B | C | | A | **∞** | **∞** | **∞** | | B | **5** | **0** | **3** | | C | **∞** | **∞** | **∞** | | |  |  |  |  | | --- | --- | --- | --- | | Node B | A | B | C | | A | **0** | **5** | **1** | | B | **5** | **0** | **3** | | C | **1** | **3** | **0** | | |  |  |  |  | | --- | --- | --- | --- | | Node B | A | B | C | | A | **0** | **4** | **1** | | B | **5** | **0** | **3** | | C | **1** | **3** | **0** | |
| |  |  |  |  | | --- | --- | --- | --- | | Node C | A | B | C | | A | **∞** | **∞** | **∞** | | B | **∞** | **∞** | **∞** | | C | **1** | **3** | **0** | | |  |  |  |  | | --- | --- | --- | --- | | Node C | A | B | C | | A | **0** | **5** | **1** | | B | **5** | **0** | **3** | | C | **1** | **3** | **0** | | |  |  |  |  | | --- | --- | --- | --- | | Node C | A | B | C | | A | **0** | **4** | **1** | | B | **5** | **0** | **3** | | C | **1** | **3** | **0** | | |

Da(a) = 0 , Db(b) = 0 , Dc(c) = 0

Da(b) = min { c(a,b)+Db(b) , c(a,c)+Dc(b) } = min { 5+0 , 1+3 } = 4

Da(c) = min { c(a,c)+Dc(c) , c(a,b)+Db(c) } = min { 1+0 , 5+3 } = 1

Db(a) = min { c(b,a)+Da(a) , c(b,c)+Dc(b) } = min { 5+0 , 3+3 } = 5

Db(c) = min { c(b,c)+Dc(c) , c(b,a)+Da(c) } = min { 3+0 , 5+1 } = 3

Dc(a) = min { c(c,a)+Da(a) , c(c,b)+Db(a) } = min { 1+0 , 3+5 } = 1

Dc(b) = min { c(c,b)+Db(b) , c(c,a)+Da(b) } = min { 3+0 , 1+5 } = 3

**Question 5**

**Laboratory Homework Part 1-1: using netstat**

**netstat** is a command line tool that displays current TCP/IP network connections and protocol statistics. For more detail, refer http://www.ss64.com/nt/netstat.html.

Answer to the following questions using proper options.

1. Use netstat on your local host to find current UDP sessions and TCP connections. How many of them do you find and what port numbers are used?

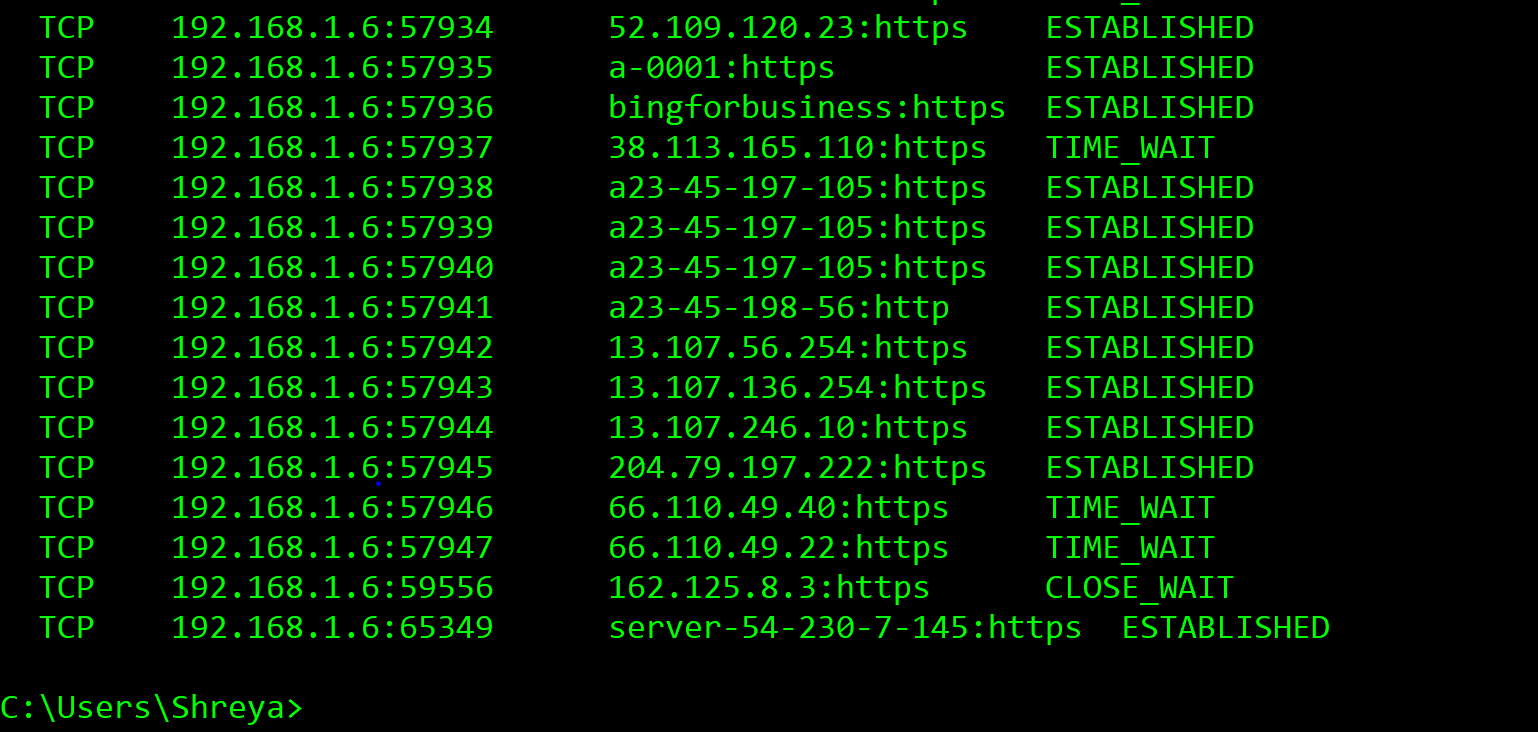
**Answer :**

**Case 1 : TCP Connections**

There are 52 TCP connections . Port numbers are 135, 445, 5357, 17500, 49664, 49665, 49666, 49667, 49668, 49671, 49672, 843, 17600, 49673, 49674, 53146, 53147, 59545, 59546, 59553, 59554, 139, 5040, 53129, 54352, 56133, 56134, 57577, 57604, 57714, 57916, 57925, 57928, 57929, 57930, 57933, 57934, 57935, 57936, 57937, 57938, 57939, 57940, 57941, 57942, 57943, 57944, 57945, 57946, 57947, 59556 and 65349.

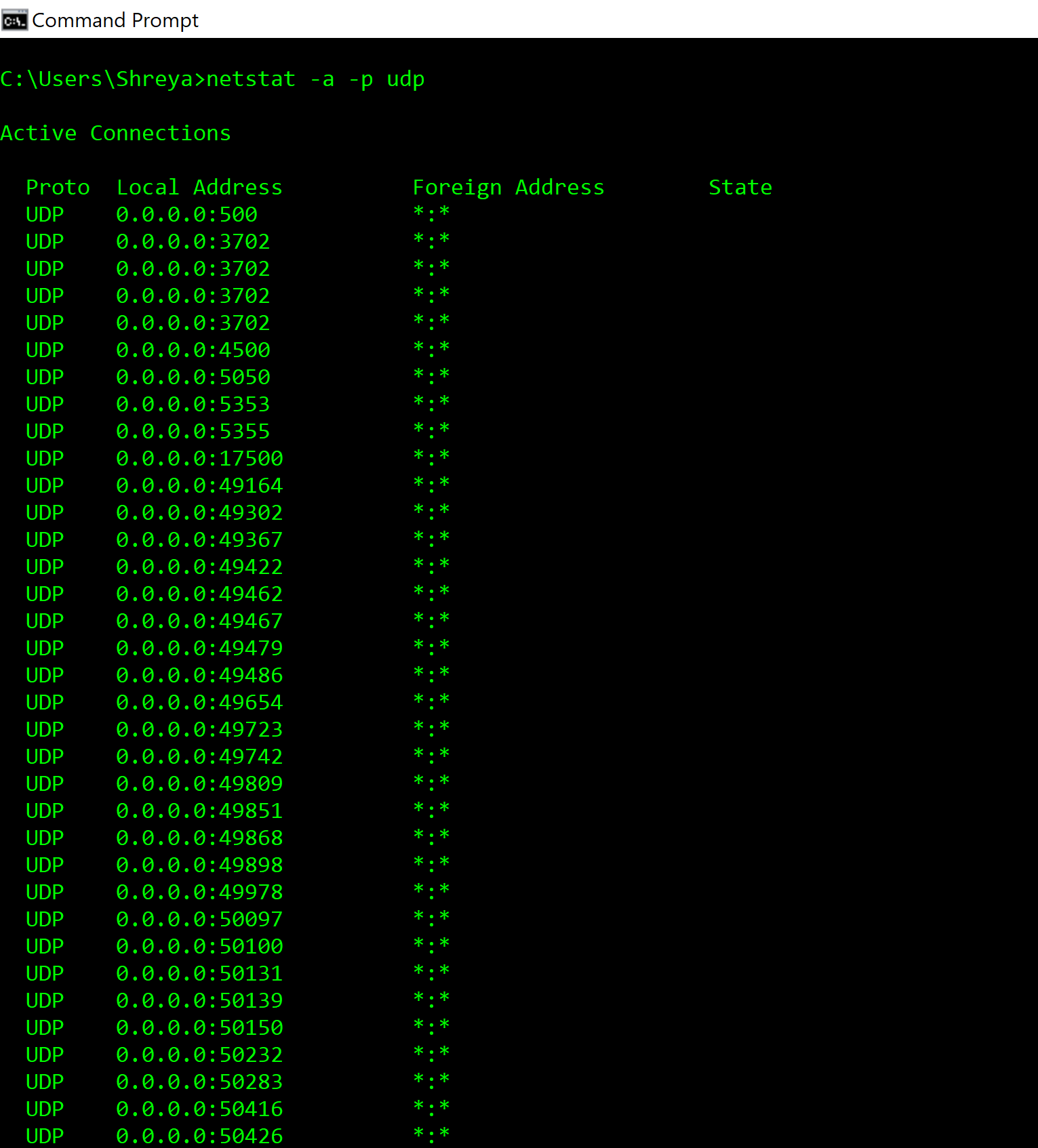




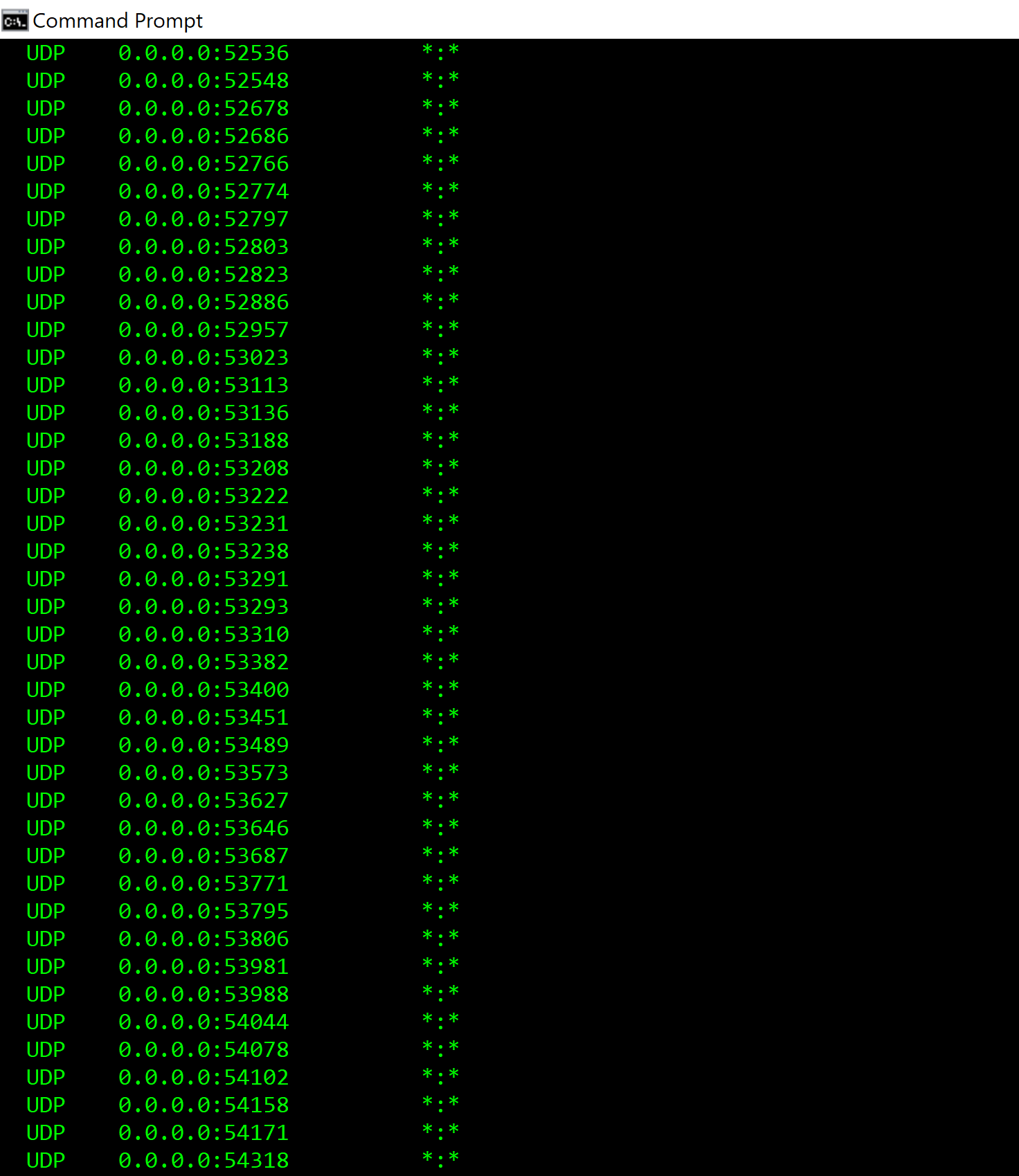


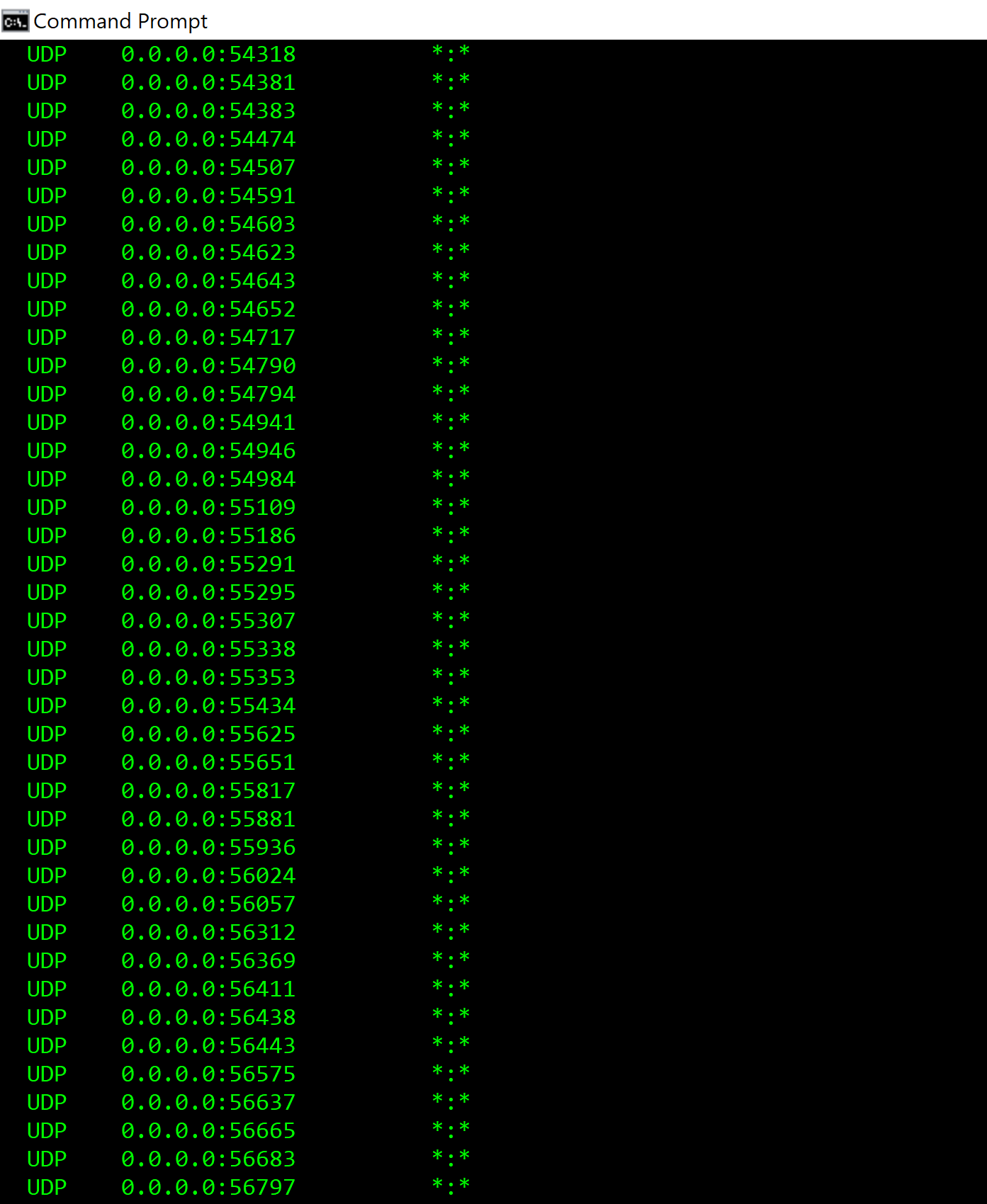
**Case 2 : UDP Connections**

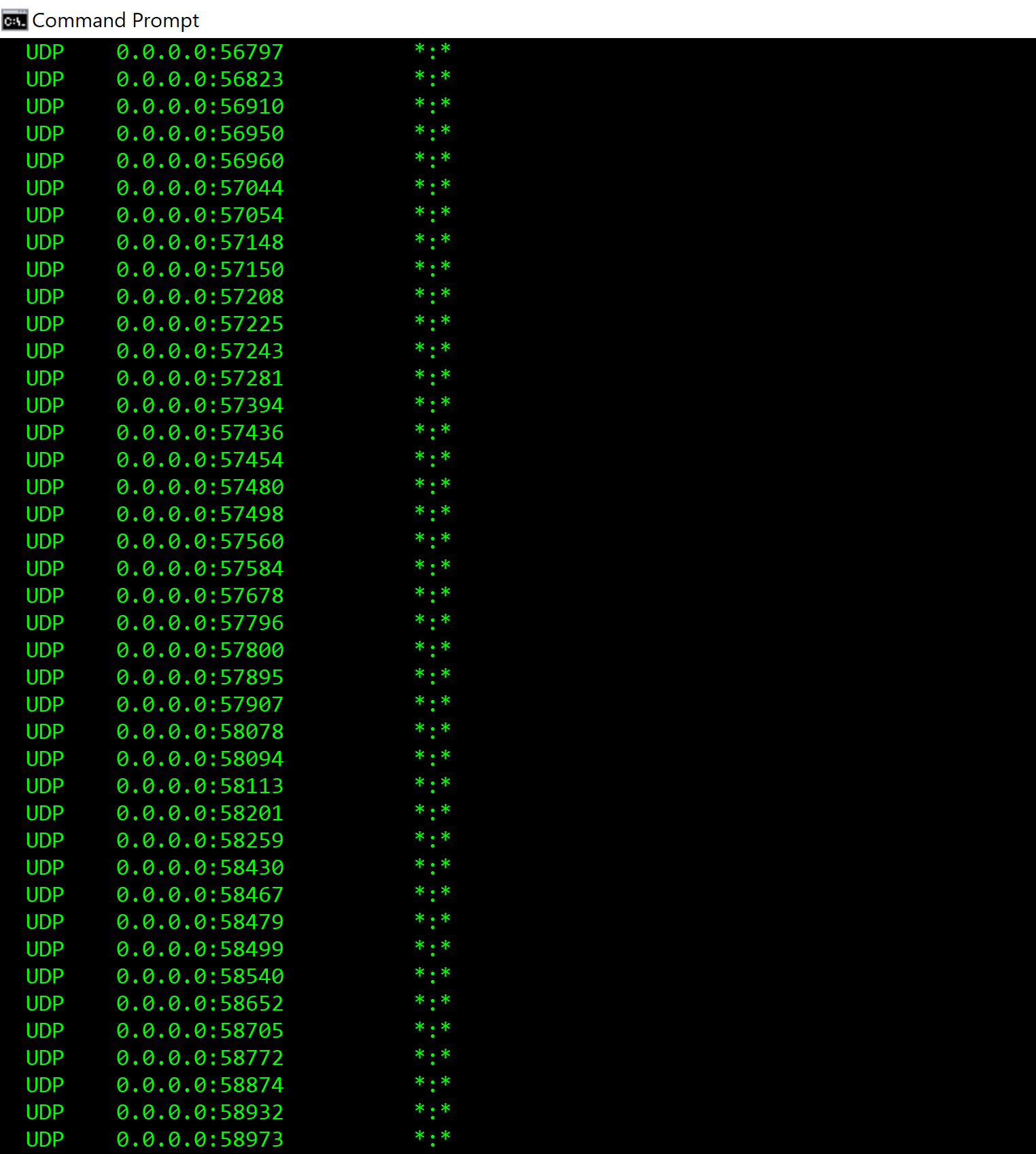
There are 322 UDP connections. Port numbers are 500, 3702, 4500, 5050, 5353, 5355, 17500, 49164, 49302, 49367, 49422, 49462, 49467, 49479, 49486, 49654, 49723, 49742, 49809, 49851, 49868, 49898, 49978, 50097, 50100, 50131, 50139, 50150, 50232, 50283, 50416, 50426, 50426, 50490, 50503, 50666, 50725, 50727, 50813, 50922, 50955, 50971, 50989, 51084, 51109, 51109, 51195, 51211, 51258, 51285, 51351, 51388, 51451, 51505, 51545, 51590, 51644, 61666, 51723, 51782, 52037, 52145, 52223, 52229, 52288, 52306, 52325, 52368, 52387, 52418, 52433, 52442, 52512, 52536, 52548, 52678, 52686, 52766, 52774,52797, 52803, 52823, 52886, 52957, 53023, 53113, 53136, 53188, 53208, 53222, 53231, 53238, 53291, 53293, 53310, 53382, 53400, 53451, 53489, 53573, 53627, 53646, 53687, 53771, 53795, 53806, 53981, 53988, 54044, 54078, 54102, 54158, 54171, 54318, 54381, 54383, 54474, 54507, 54591, 54603, 54623, 54643, 54652, 54717, 54790, 54794, 54941, 54946, 54984, 55109, 55186, 55291, 55295, 55307, 55338, 55353, 55434, 55625, 55651, 55817, 55881,55936, 56024, 56057, 56312, 56369, 56411, 56438, 56443, 56575, 56637, 56665, 56683, 56797, 56823, 56910, 56950, 56960, 57044, 57054, 57148, 57150, 57208, 57225, 57243, 57281, 57394,57436, 57454, 57480, 57498, 57560, 57584, 57678, 57796, 57800, 57895, 57907, 58078, 58094, 58113, 58201, 58259, 58430, 58467, 58479, 58499, 58540, 58652, 58705, 58772, 58874, 58932, 58973, 58976, 59052, 59069, 59083, 59114, 59134, 59170, 59244, 59247, 59253, 59270, 59312, 59333, 59433, 59434, 59464, 59562, 59587, 59592, 59608, 59613, 59616, 59626, 59634, 59666, 59692, 59715, 59717, 59807, 59835, 59898, 59942, 60085, 60133, 60176, 60263, 60346, 60522, 60686, 60707, 60749, 60755, 60808, 60898, 60960, 61056, 61112, 61208, 61250, 61257, 61305, 61332, 61475, 61712, 61727, 61730, 61831, 61842, 61905, 61976, 62000, 62142, 62184, 62196, 62252, 62259, 62270, 62288, 62319, 62374, 62382, 62407, 62430, 62483, 62518, 62572, 62630, 62730, 62865, 62870, 62907, 62919, 62932, 62942, 62970, 62997, 63170, 63224, 63228, 63253, 63363, 63378, 63580, 63626, 63817, 63901, 64162, 64243, 64383, 64417, 64496, 64499, 64506, 64639, 64707, 64708, 64736, 64755, 64901, 64912, 65042, 65085, 65086, 65111, 65160, 65244, 65287, 65318, 65350, 65378, 1900, 58211, 60161, 60558, 60860, 137, 138 and 58210.

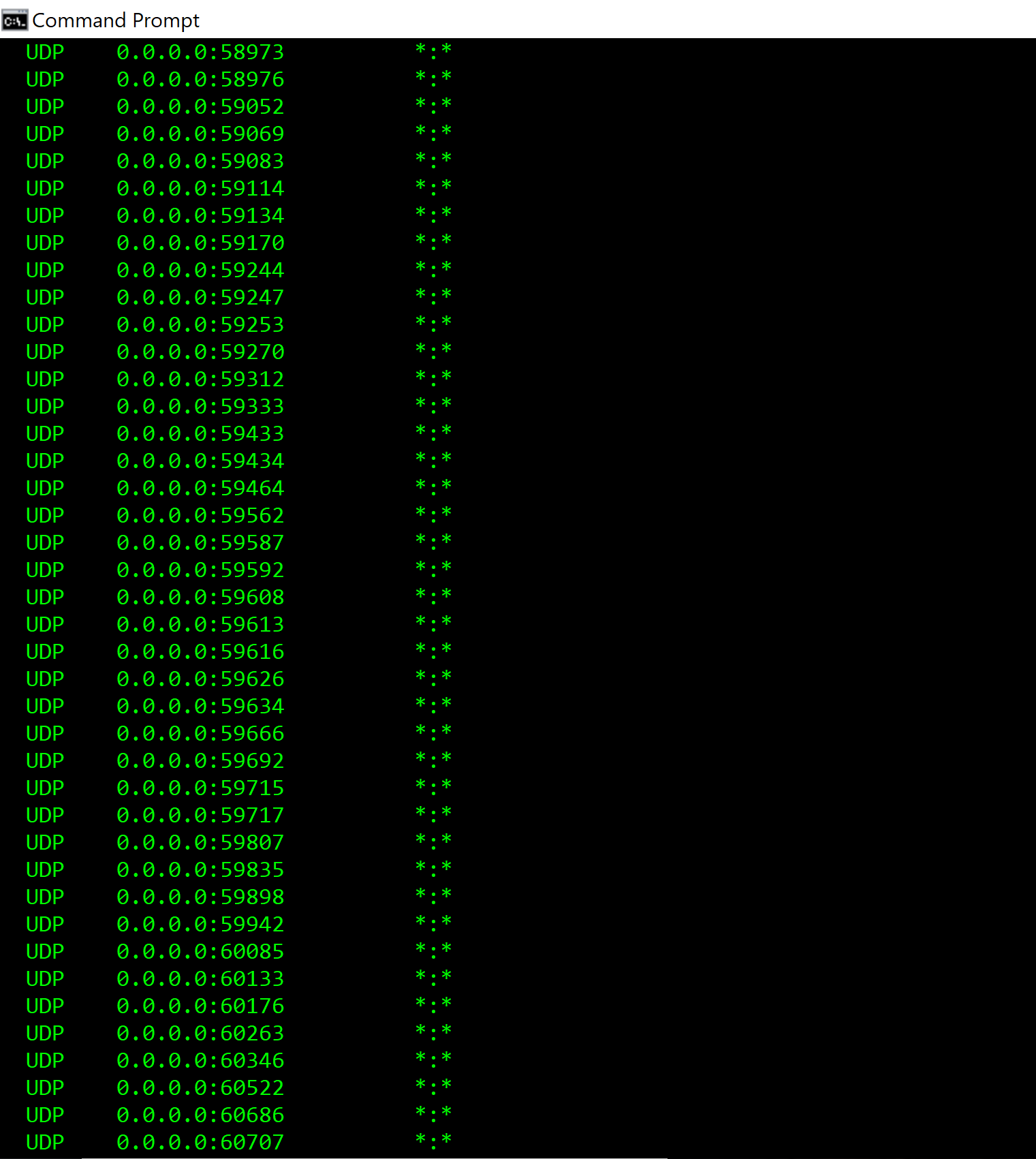


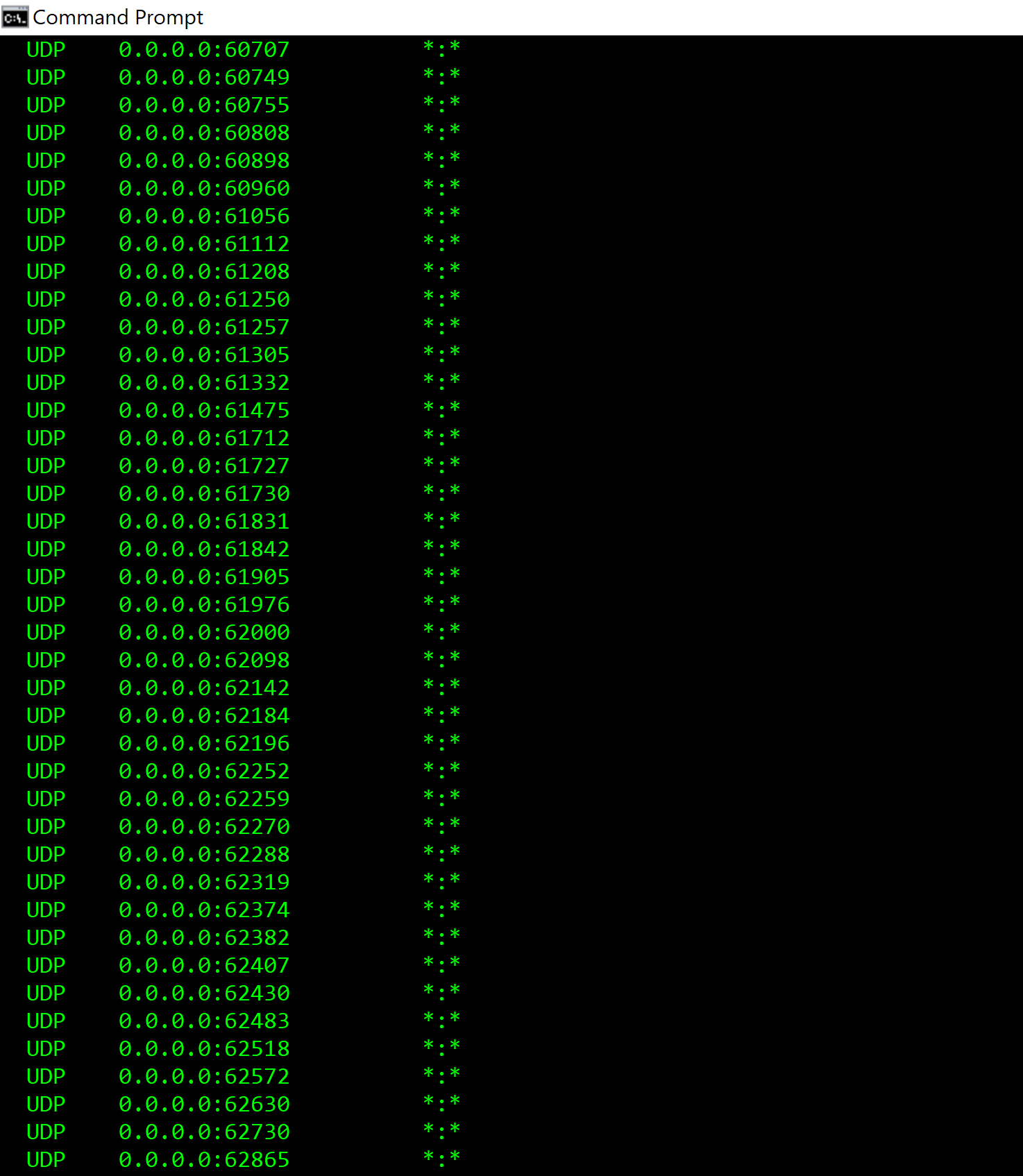


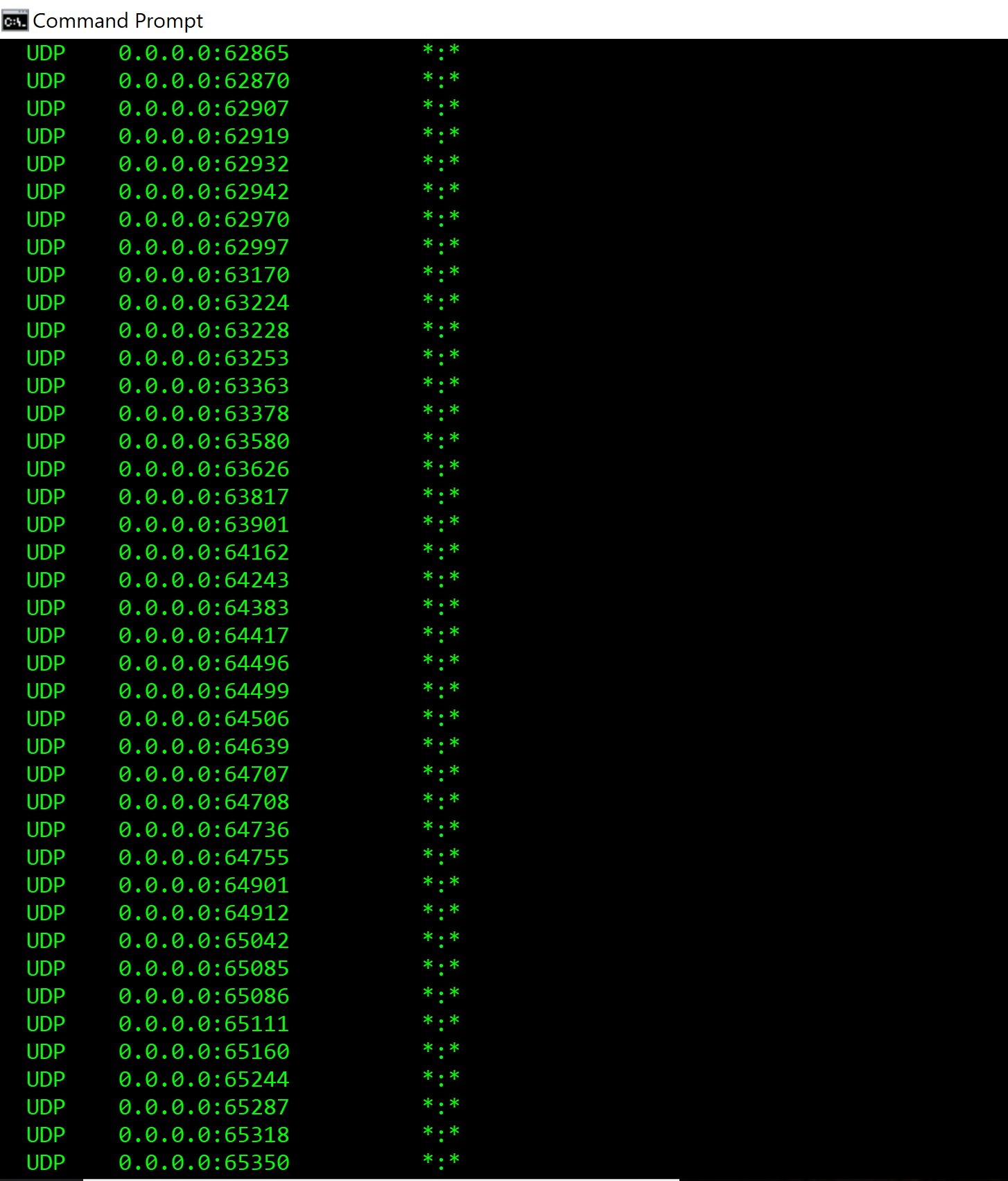


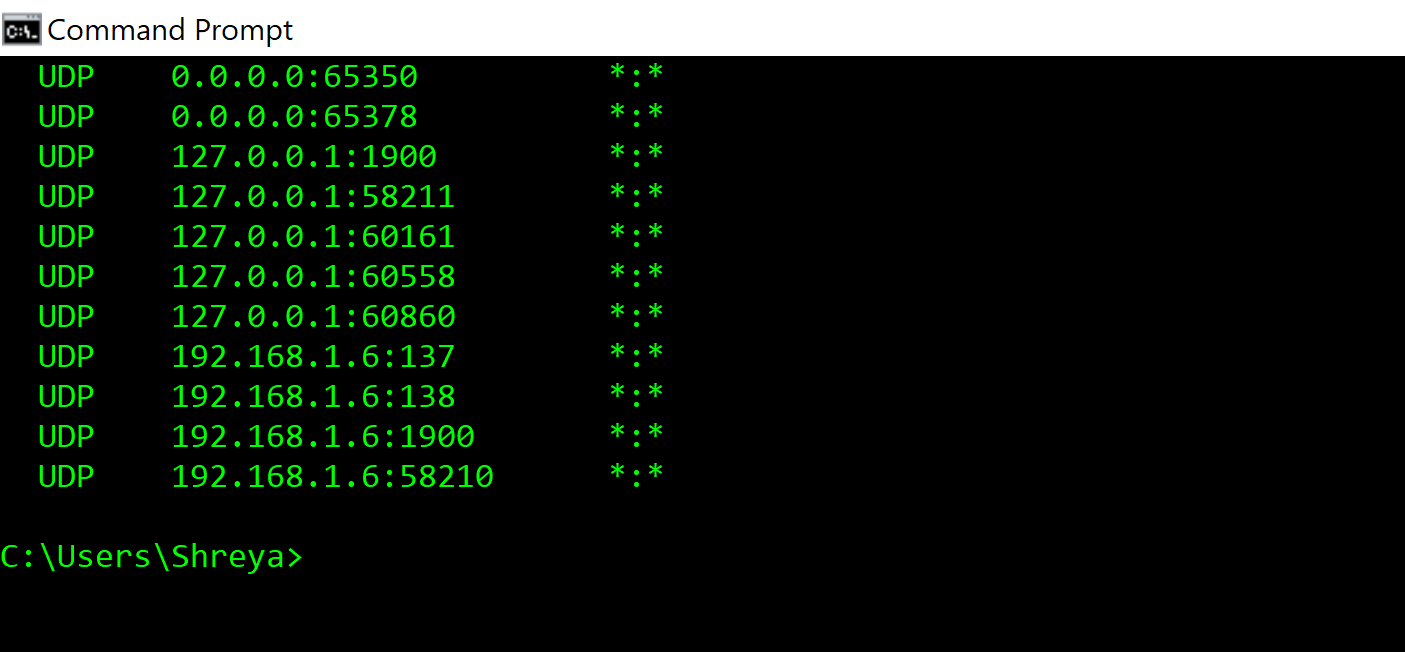










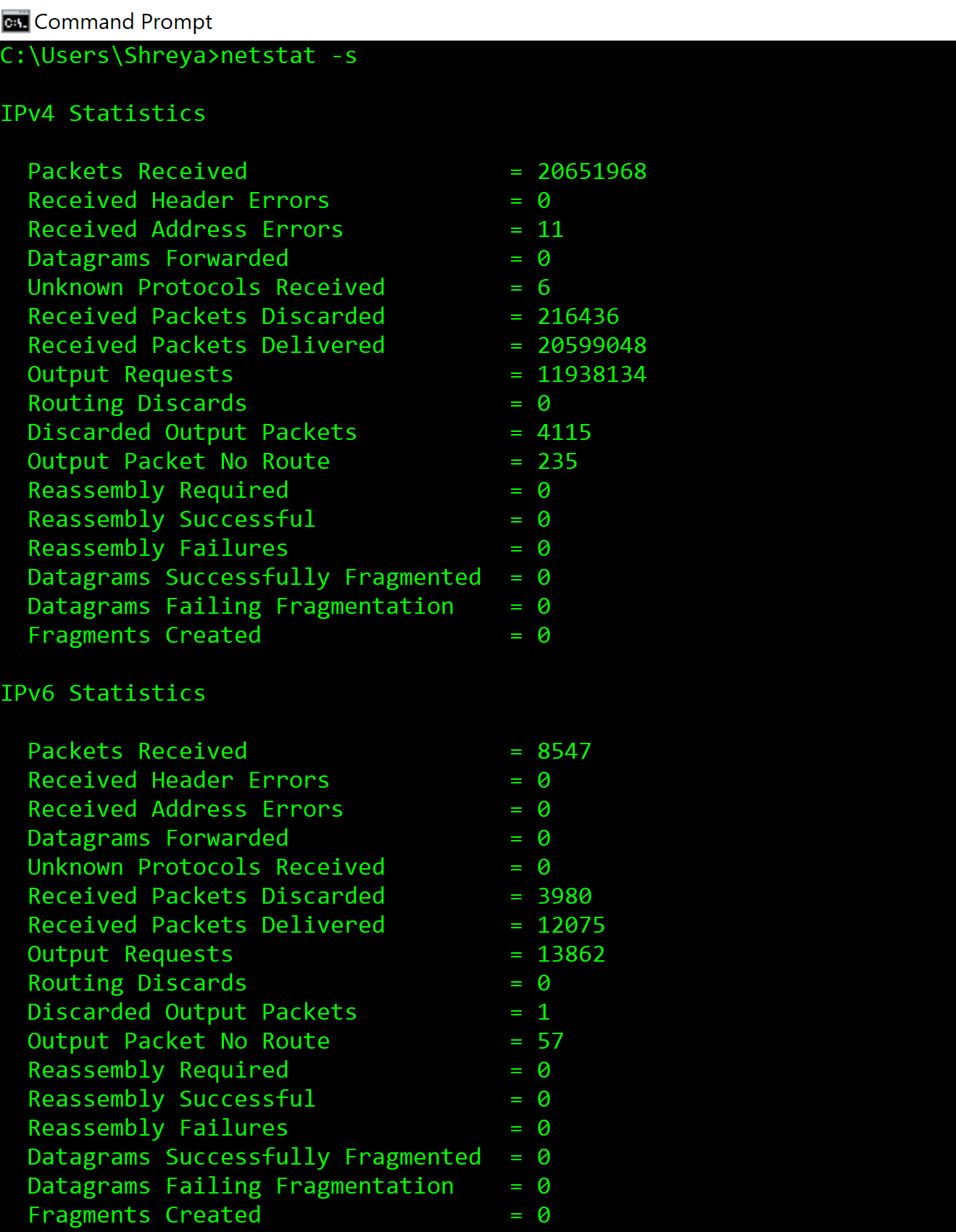


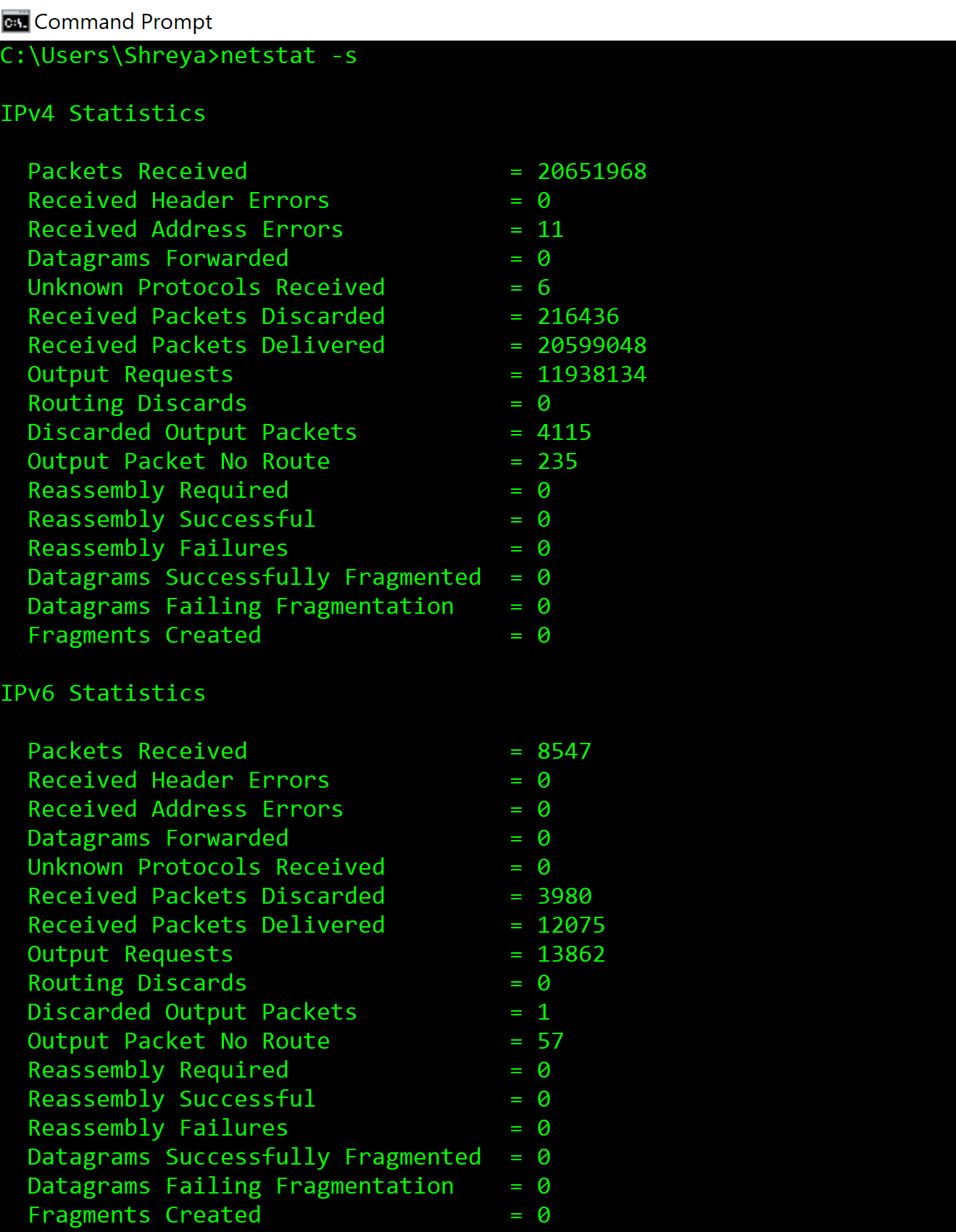
1. Find out per-protocol (IP, ICMP, TCP and UDP) statistics (using –s option).

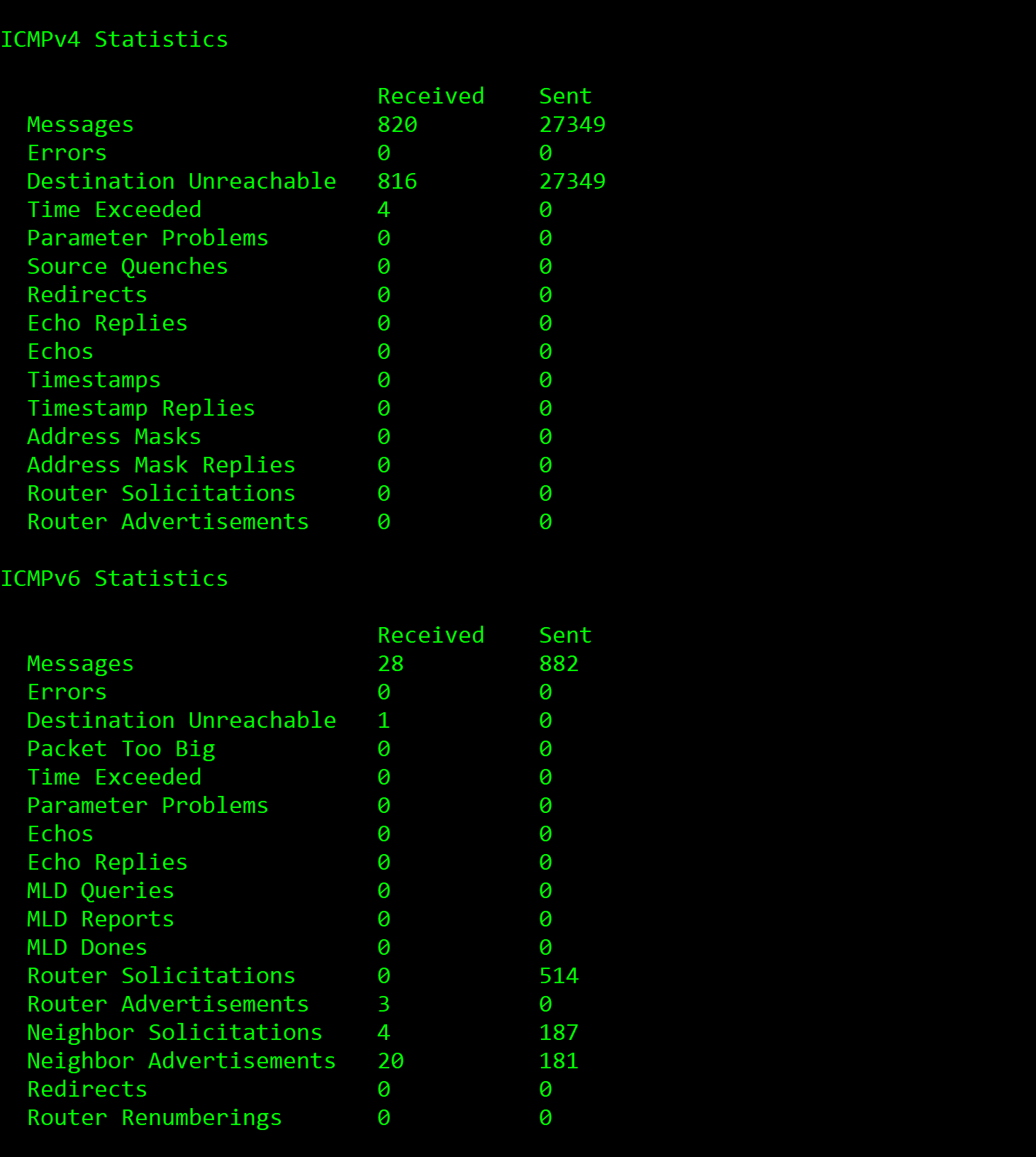
Then try ping or tracesroute to a well-known server (eg. www.google.com). Now check per-protocol (IP, ICMP, TCP and UDP) statistics again. Summarize your findings.

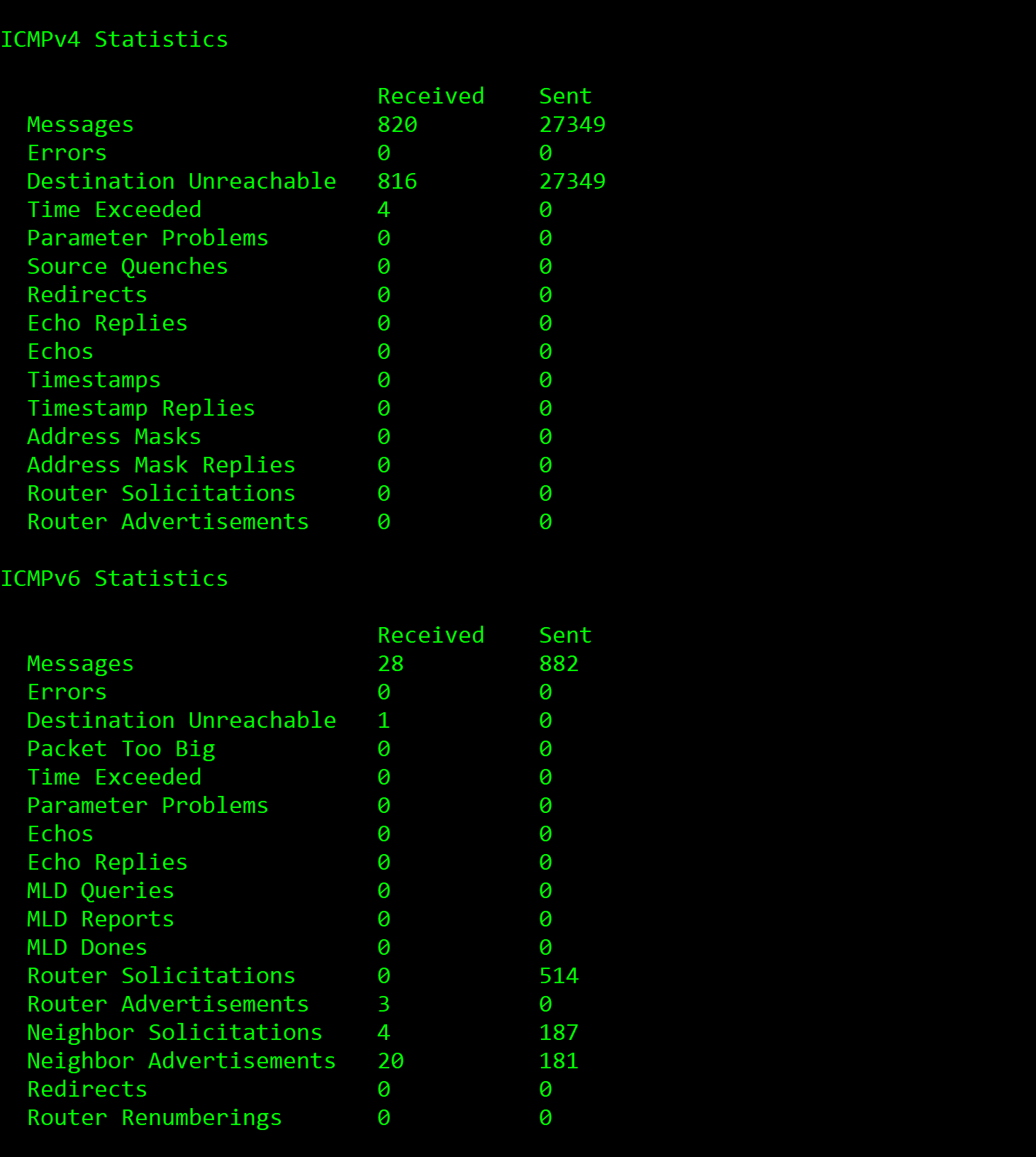
**Answer :**

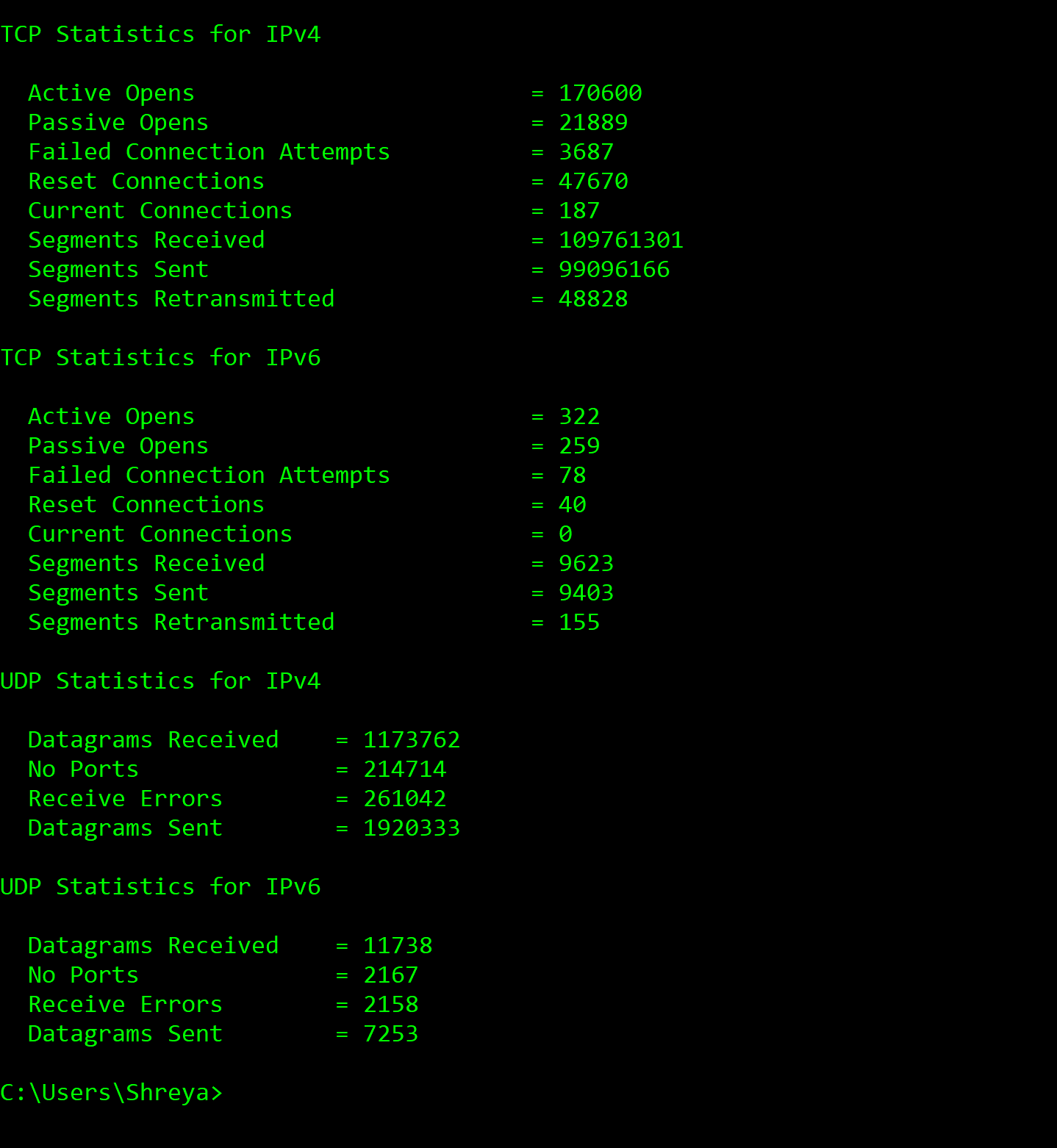
Run command **netstat -s**



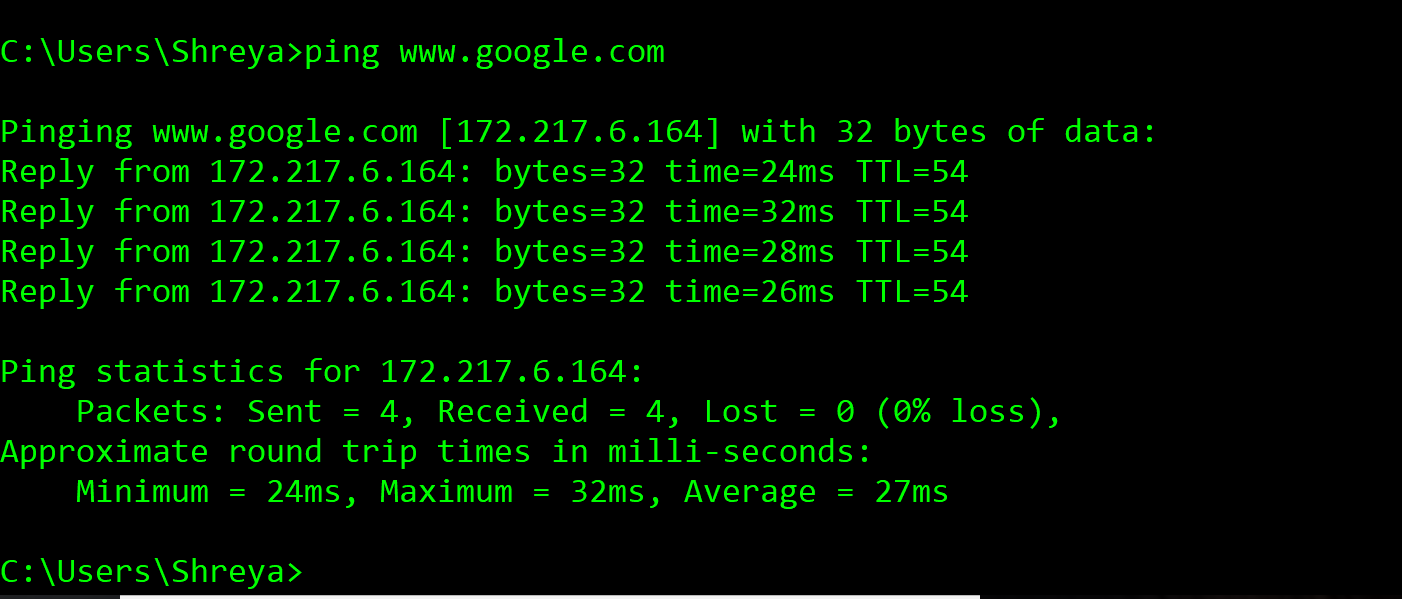




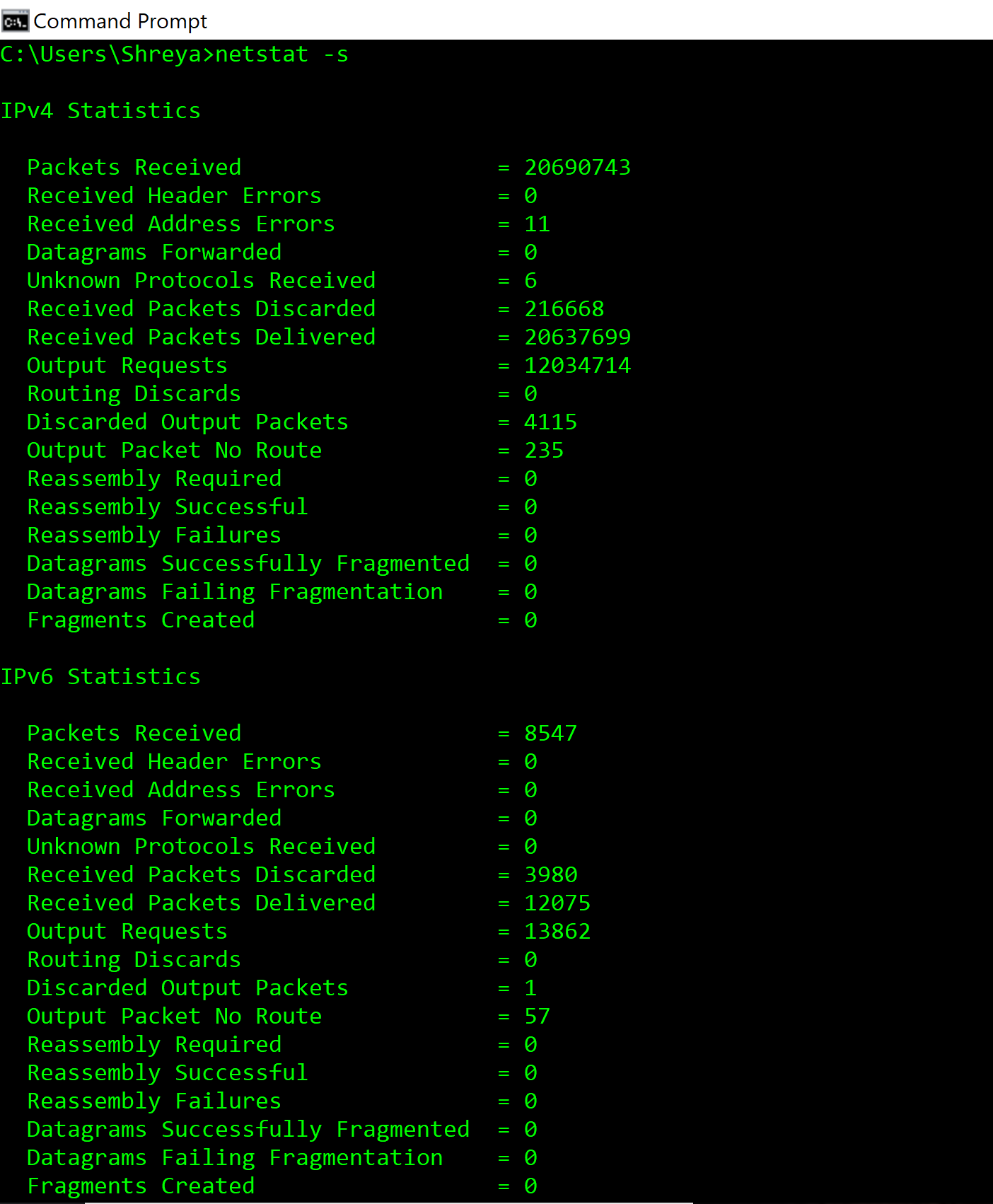


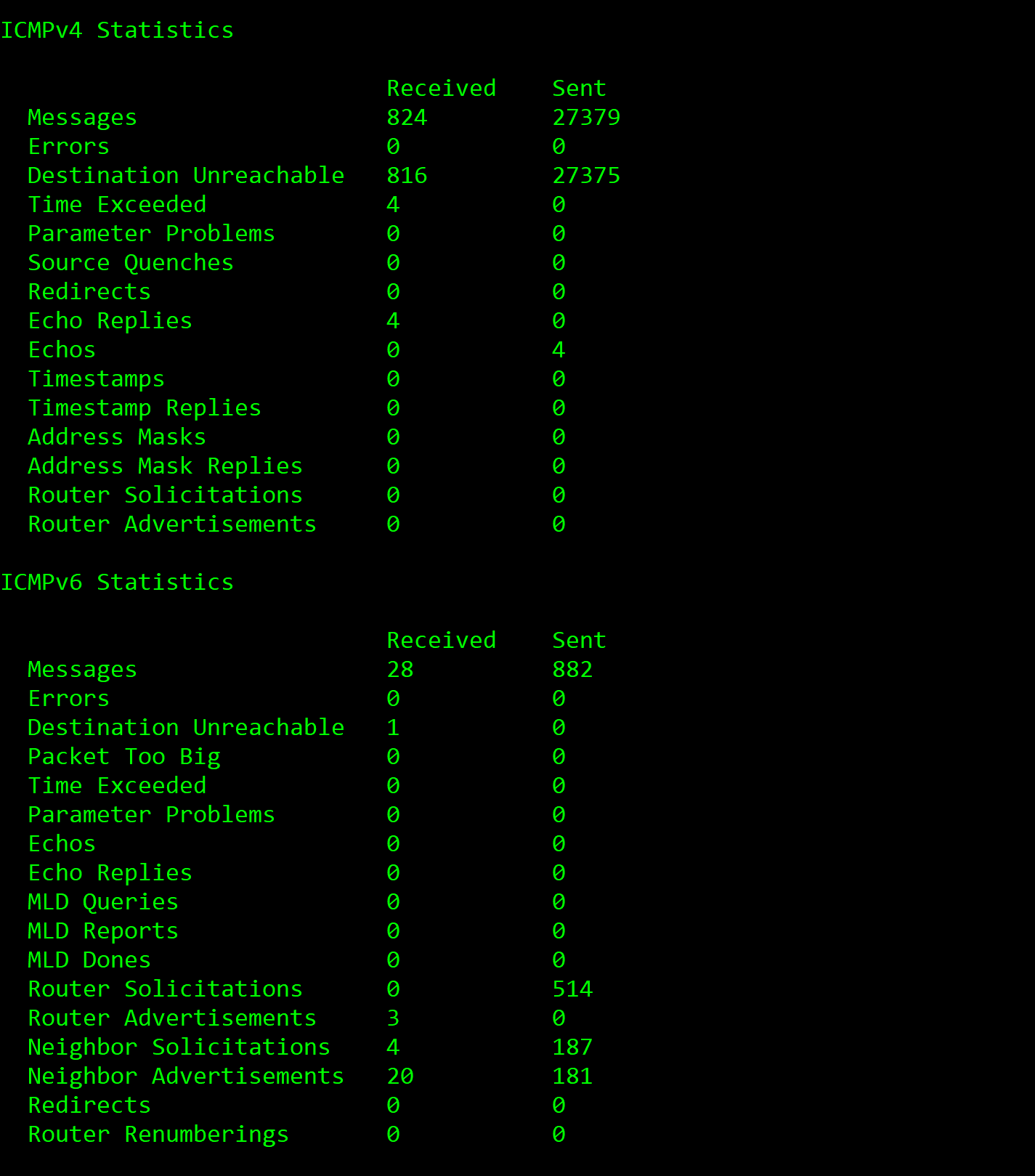


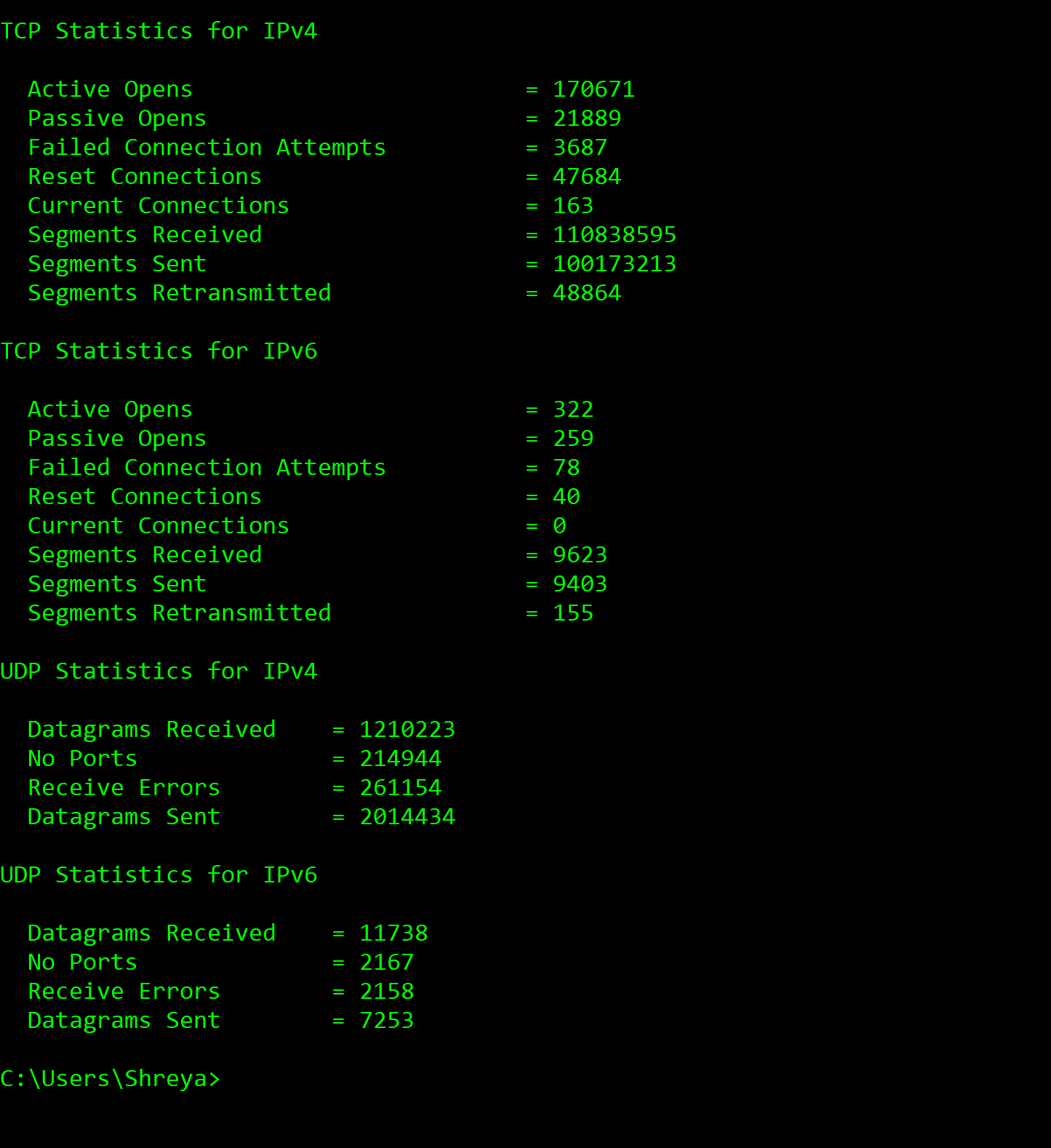
After this I have run the ping command **ping www.google.com**

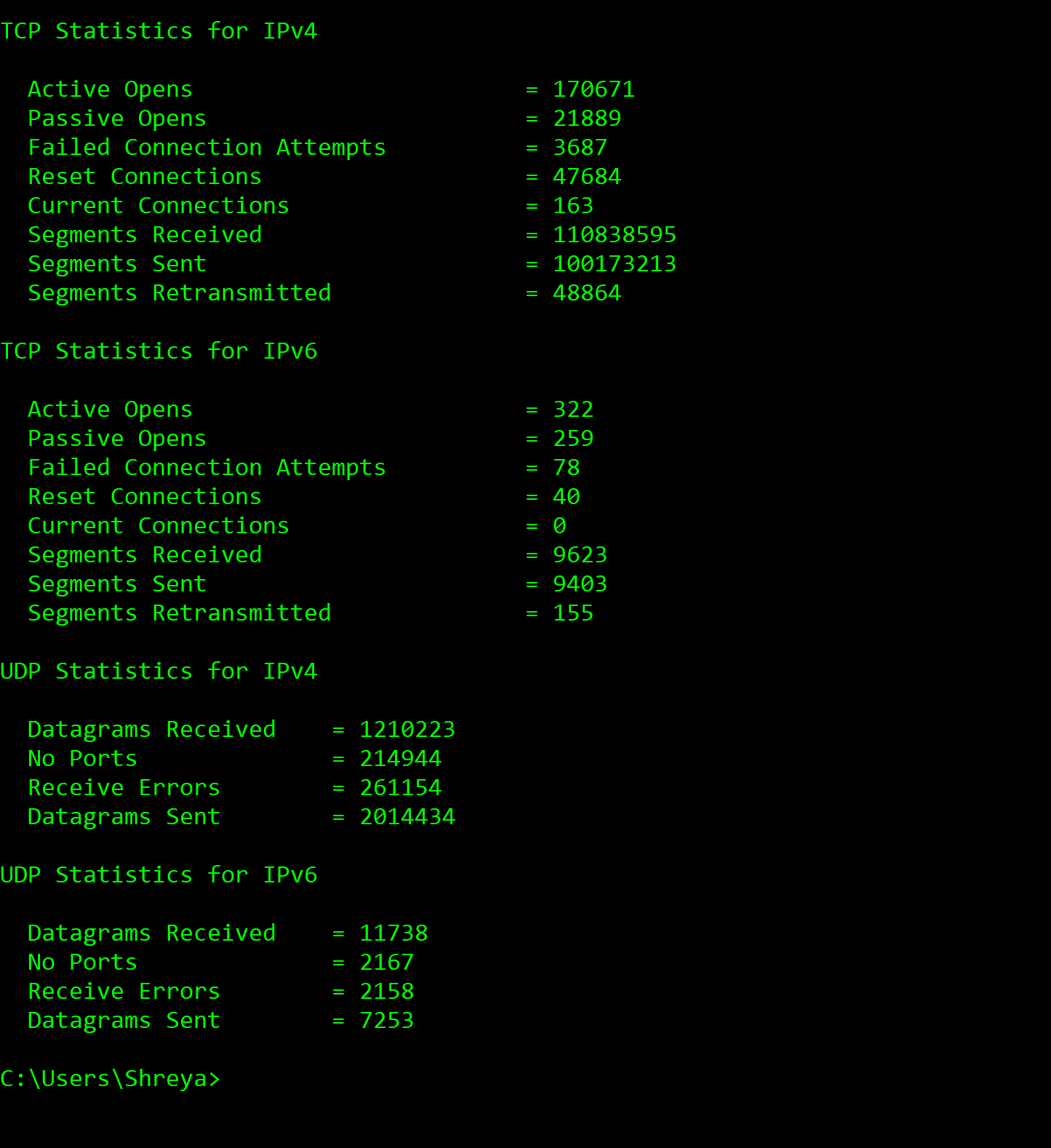


Then re run the **netstat -s** command to check for the changes.





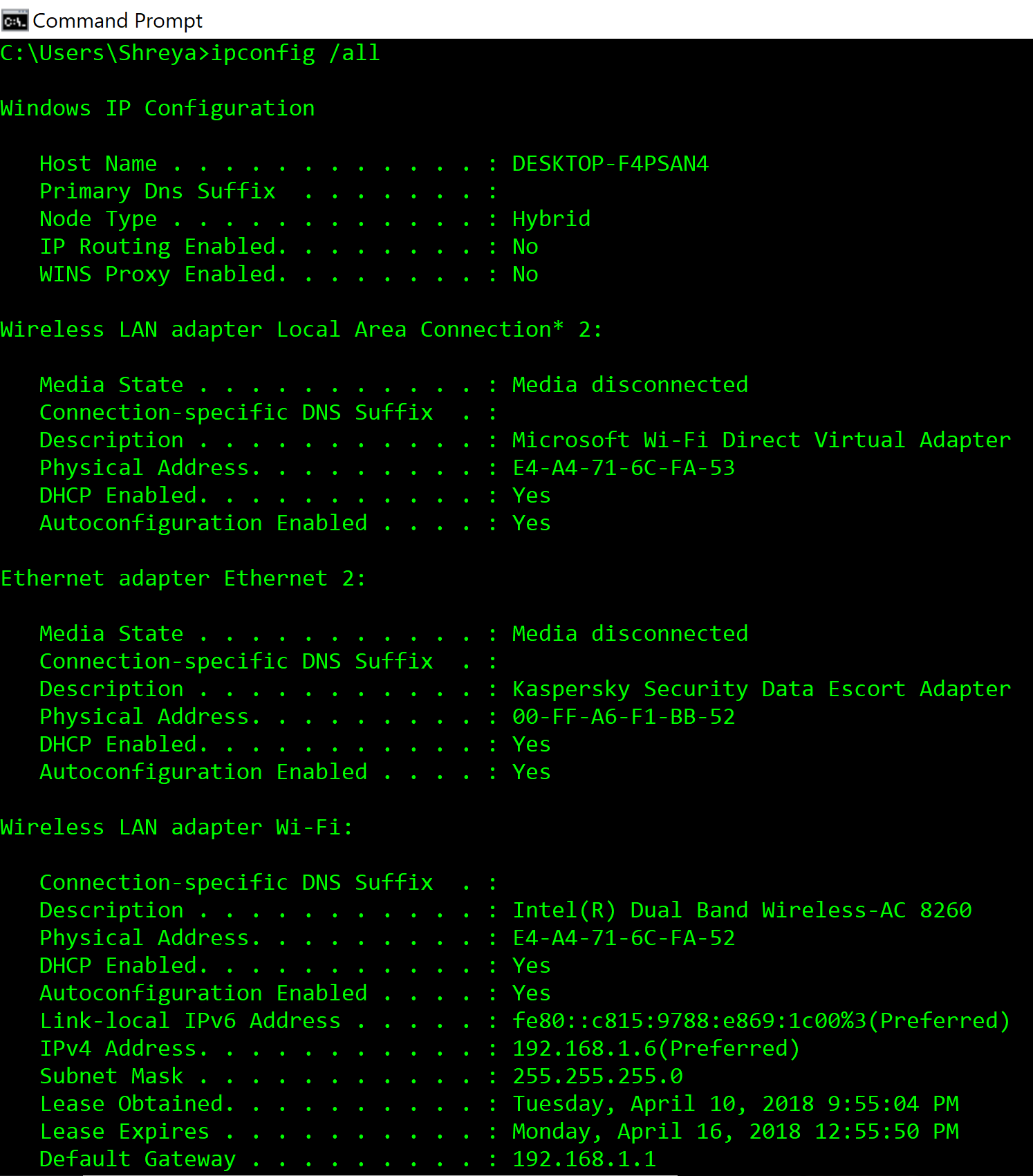


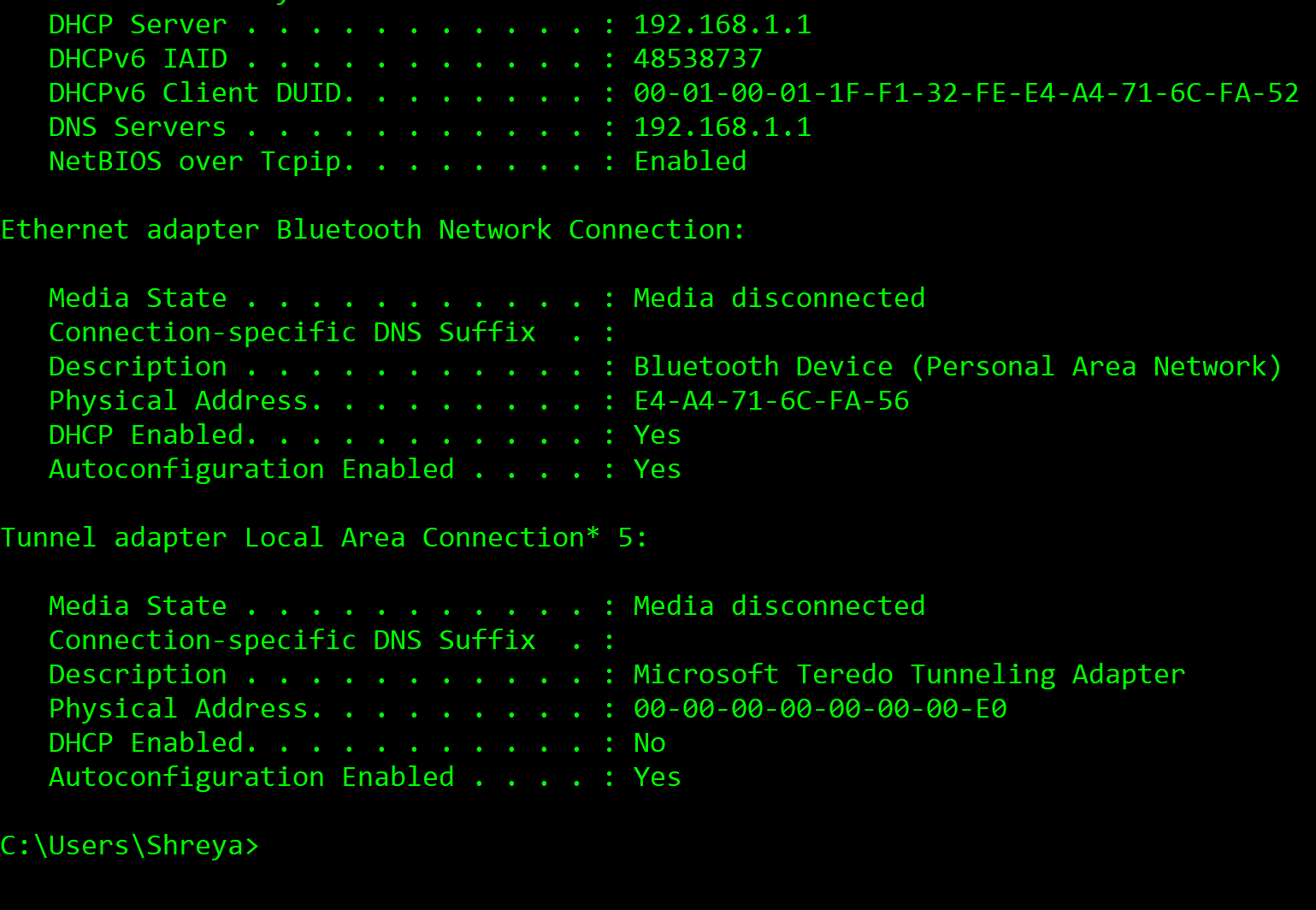


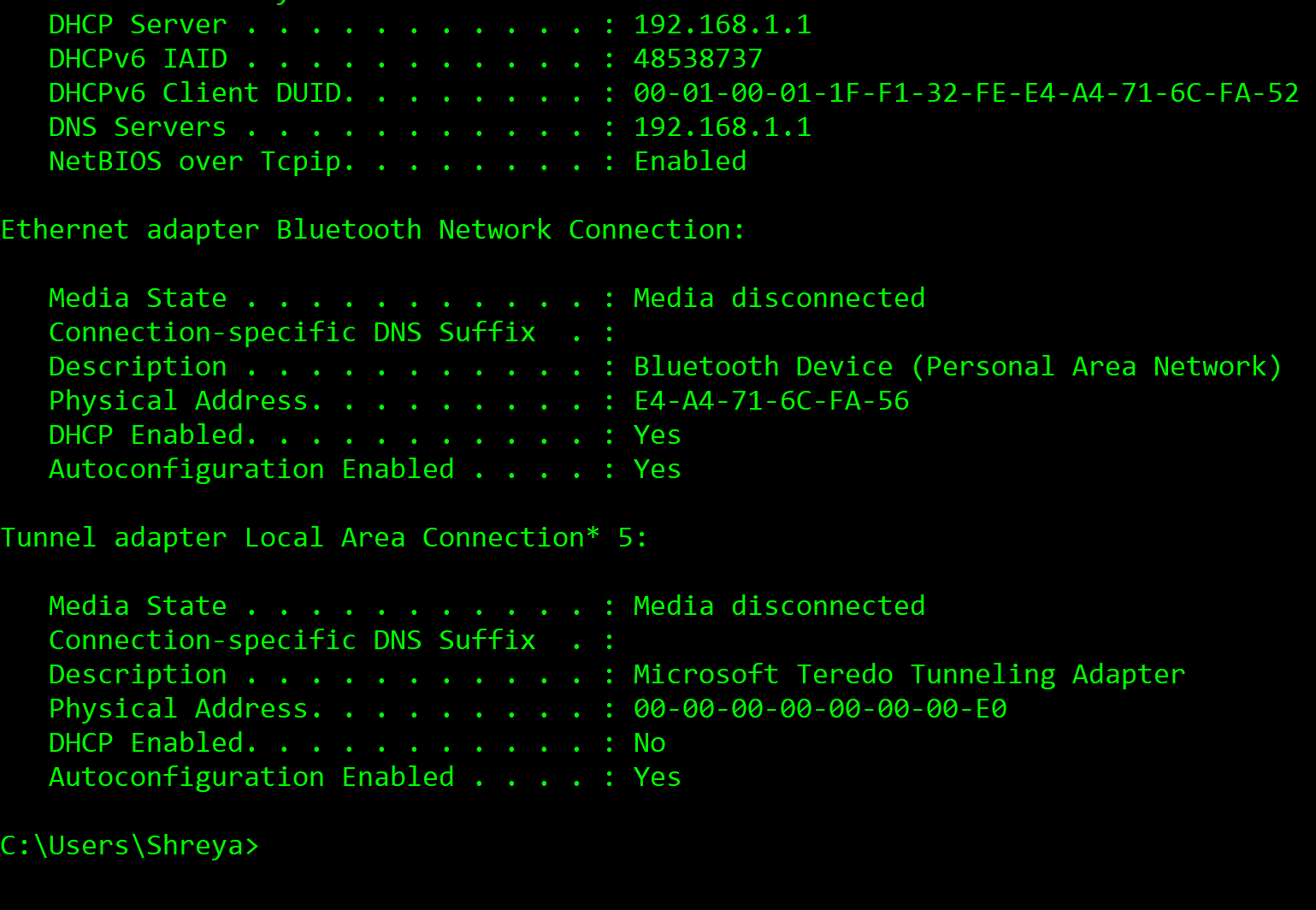
**Laboratory Homework Part 1-2: using ipconfig(Windows)**

ipconfig is a command line tool used to control the network connections on Windows machines. The Linux/Unix equivalent of ipconfig is ifconfig. For more detail, refer <http://www.ss64.com/nt/ipconfig.html>.

Run command **ipconfig /all** to get all details.







Answer to the following questions after trying various options.

1. What are the Physical and IP addresses of the host?

**Answer :**

Physical Address : E4-A4-71-6C-FA-52

Link-local IPV6 address : fe80::c815:9788:e869:1c00

IPV4 address : 192.168.1.6

1. How many bits are for the subnet mask? What is the subnet (not subnet mask) of the host?

**Answer :**

Subnet mask of 255.255.255.0 in binary form can be represented as 11111111.11111111.11111111.00000000. There are 24-bits for the subnet mask 255.255.255.0.

Subnetting is a process of designating some high-order bits from the host part and grouping them with the network mask to form the subnet mask. Here, all the hosts in subnet will have the IP addresses in the form 192.168.1.xxx.

Subnet ID of this network is 192.168.1.0.

**Laboratory Homework Part 2: Wireshark**

In this part of the homework, you investigate the IP protocol, focusing on the IP datagram. You will inspect the various fields in the IP datagram, and study IP fragmentation in detail.

Download a packet trace file of this homework from the Blackboard, Assignment Section.

In your trace, you should be able to see the series of ICMP Echo Request (Windows machine) (the UDP segment in the case of Linux/Unix) sent by the user's computer and the ICMP Time Exceeded messages returned to the user's computer by the intermediate routers.

**Answer to the questions below.**

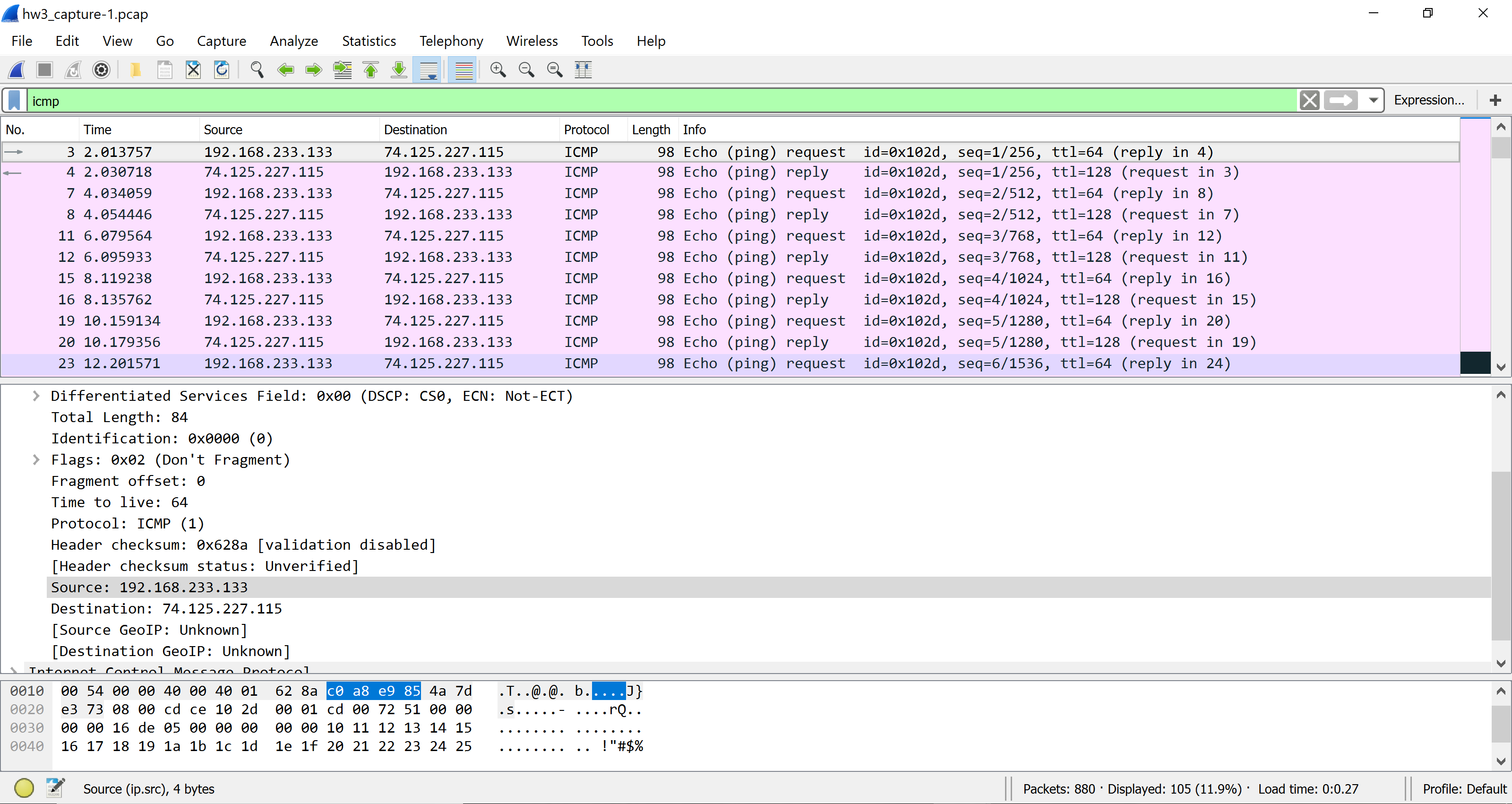
**IP Datagram**

1. Select the first ICMP Echo Request message sent by the computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of the user's computer?

**Answer :**

IP address of source user’s computer is **192.168.233.133**

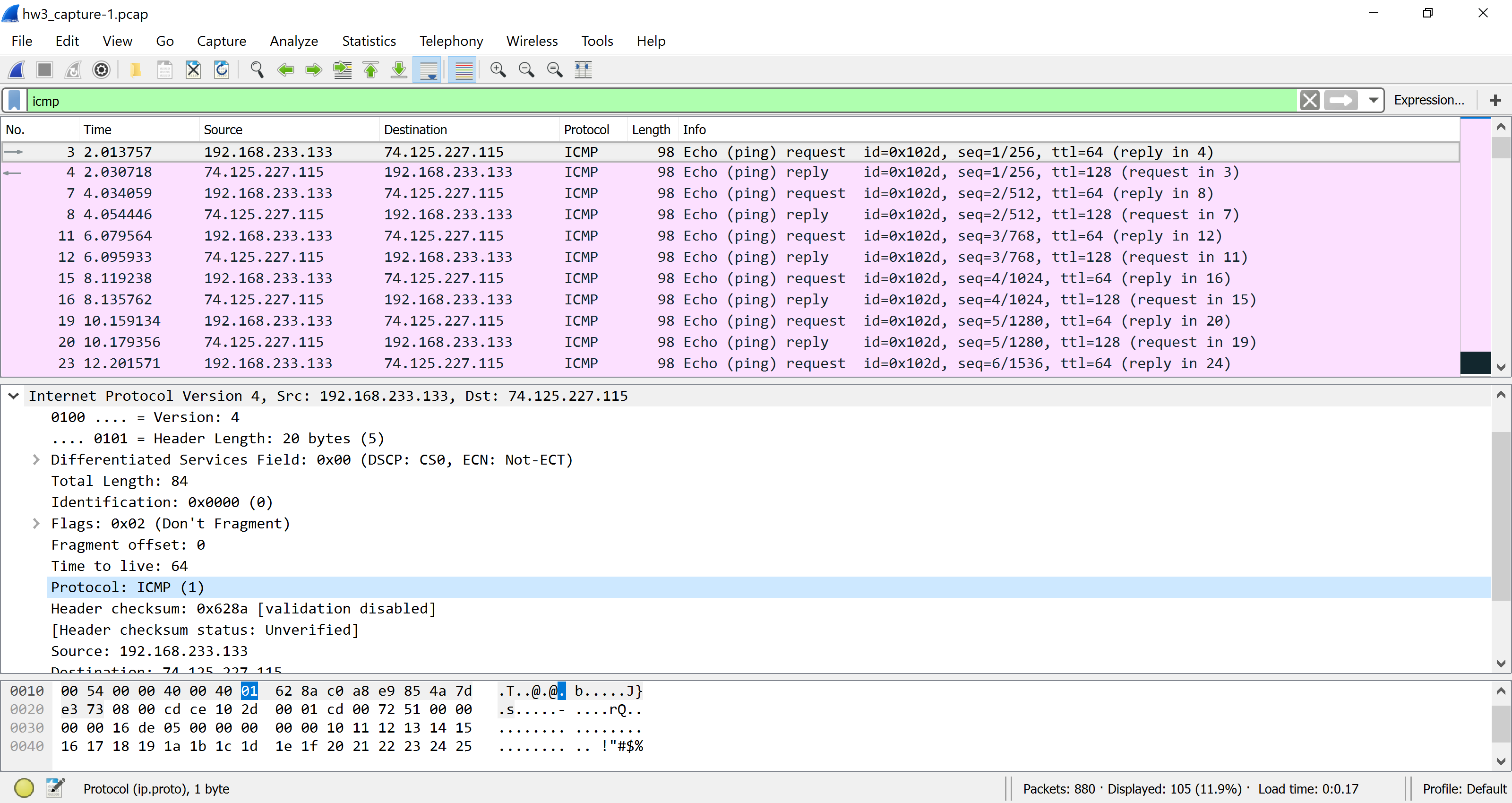
IP address od destination user’s computer is **74.125.227.115**



1. Within the IP packet header, what is the value in the upper layer protocol field?

**Answer :**

The value of upper layer protocol field is ICMP (1) , which is 1.



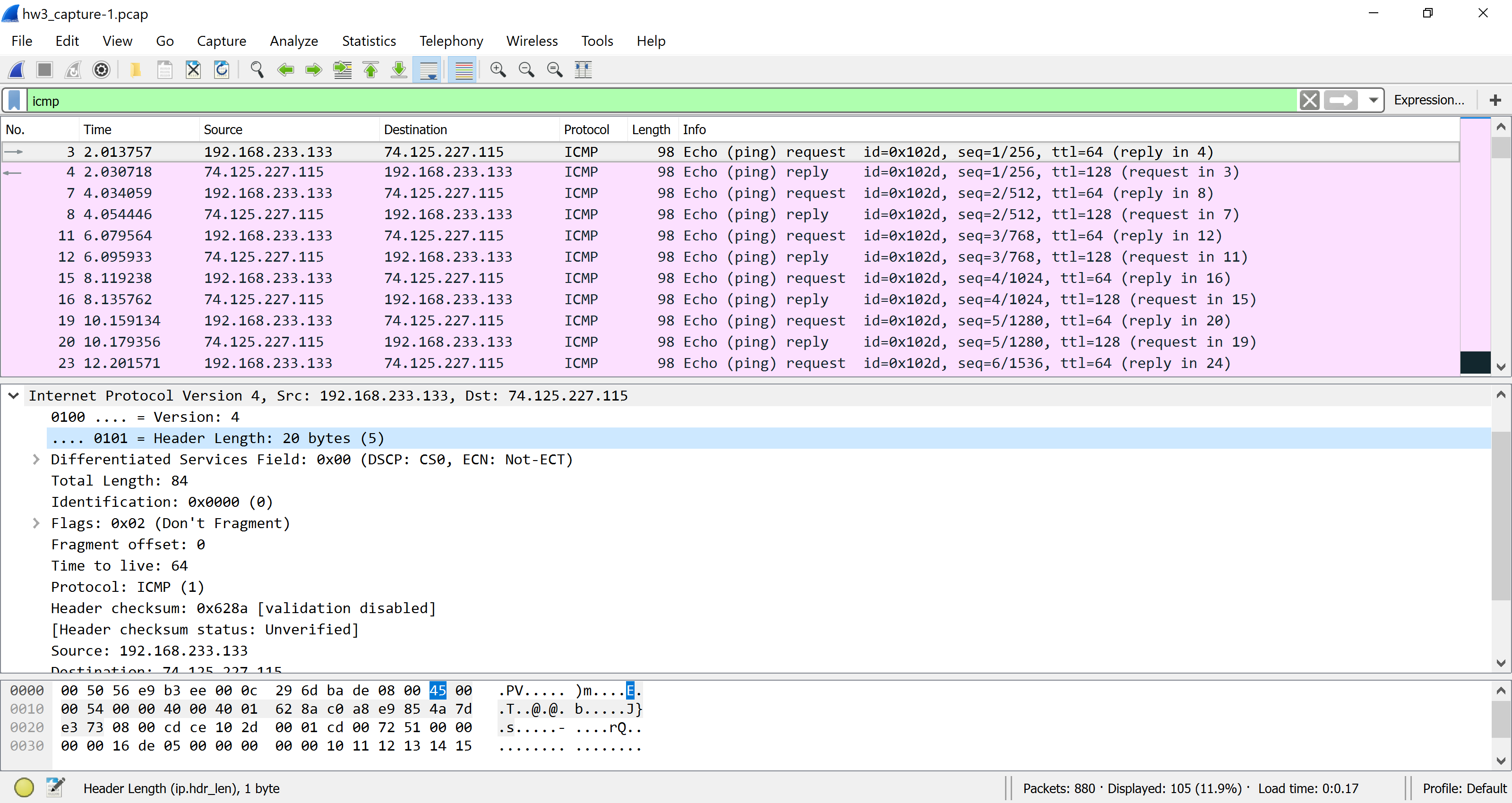
1. How many bytes are in the IP header? How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.

**Answer :**

Number of bytes in the IP header are 20 bytes.

Number of bytes in the payload of IP datagram =Total length – IP header bytes= 84 – 20= 64

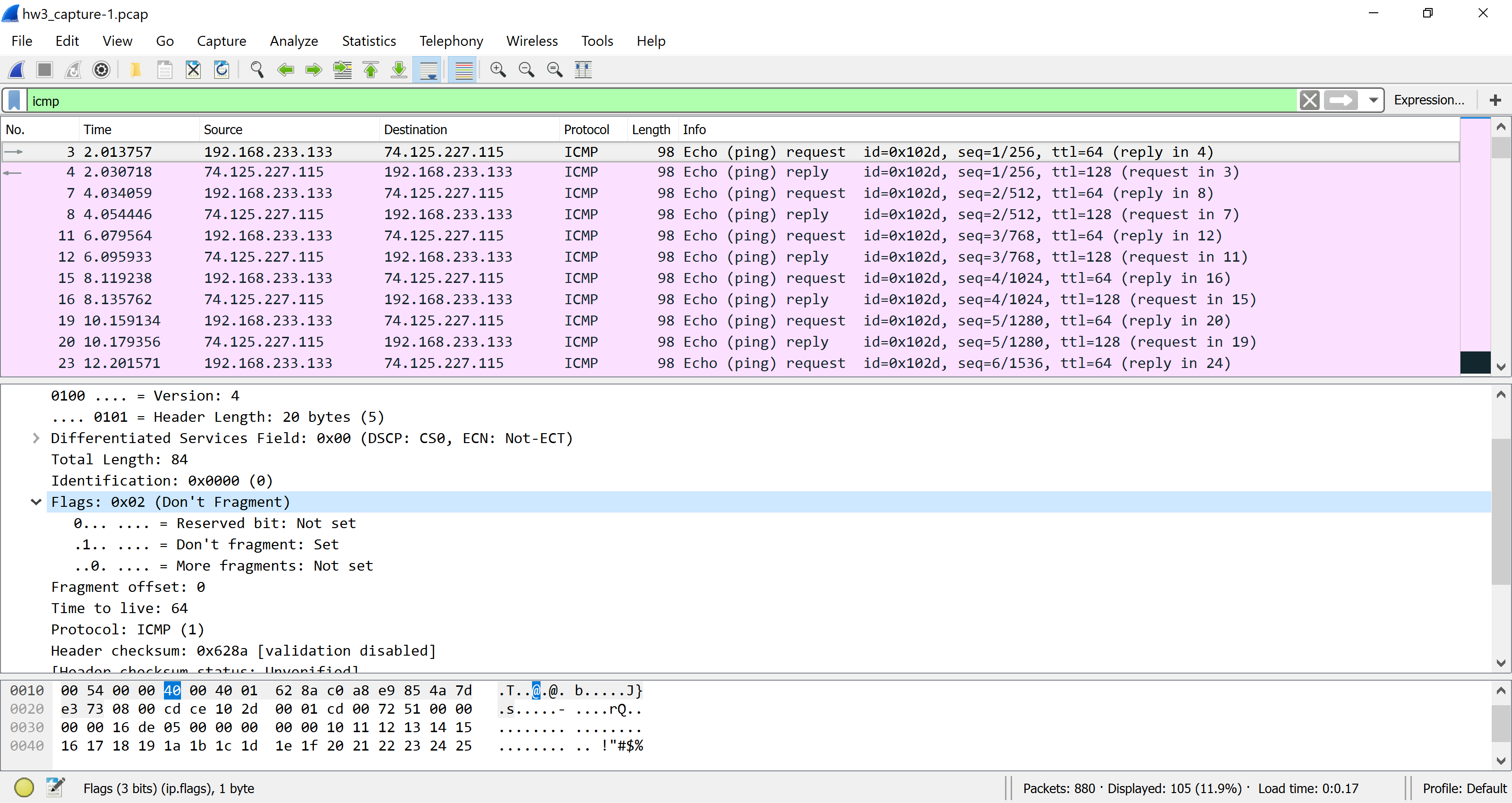
Therefore, there are **64 bytes** in the payload of IP datagram.



1. Has this IP datagram been fragmented? Explain how you determined whether the datagram has been fragmented.

**Answer :**

The fragment offset is 0 so therefore the IP datagram has not been fragmented.



Next, sort the traced packets per IP source address by clicking on the Source column header; a small downward pointing arrow should appear next to the word Source. If the arrow points up, click on the Source column header again. Select the first ICMP Echo Request message sent by the computer, and expand the Internet Protocol portion in the “details of selected packet header” window. In the “listing of captured packets” window, you should see all the subsequent ICMP messages (perhaps with additional interspersed packets sent other protocols running on the computer) below this first ICMP.

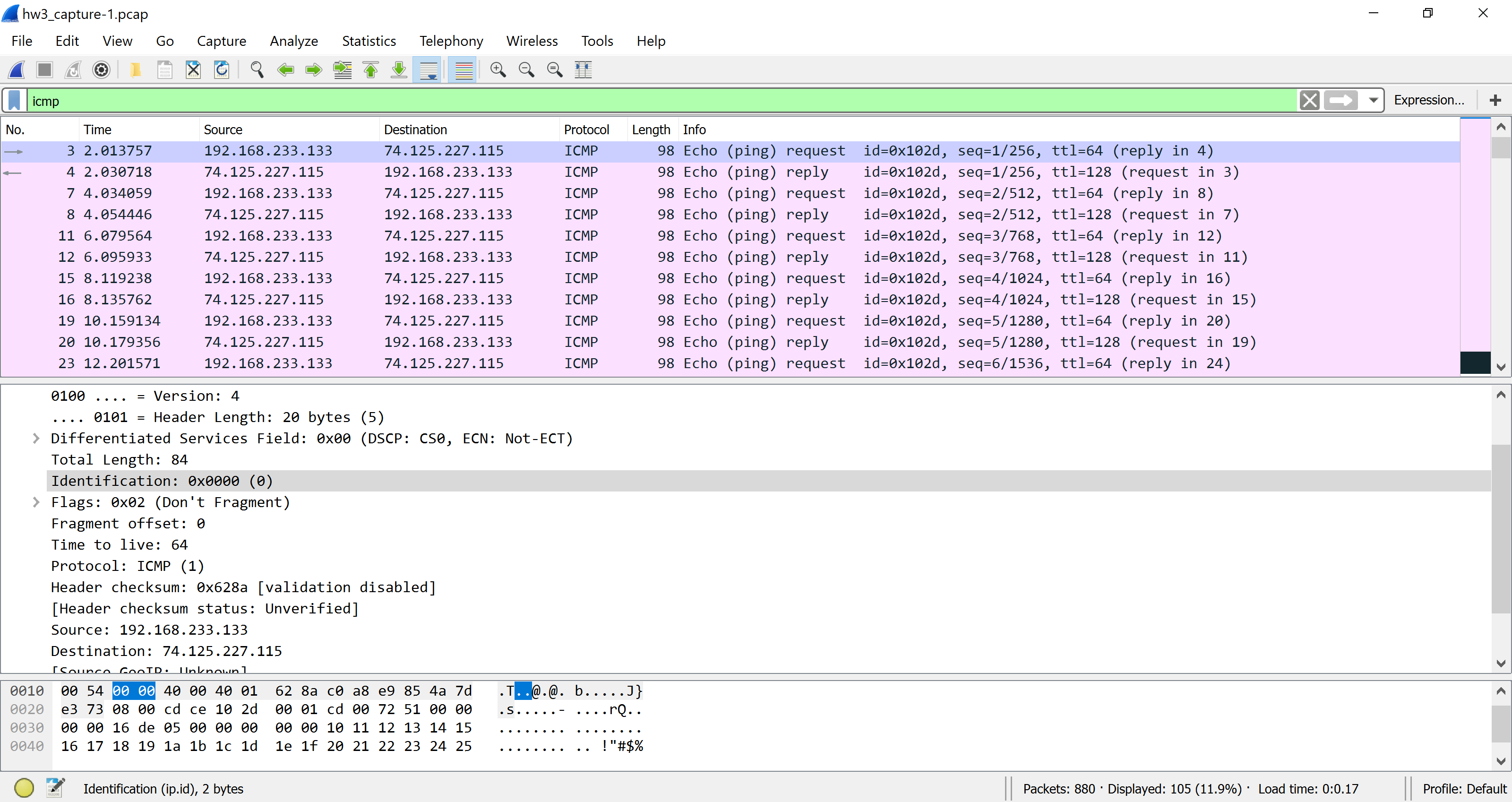
Use the down arrow to move through the ICMP messages sent by the computer.

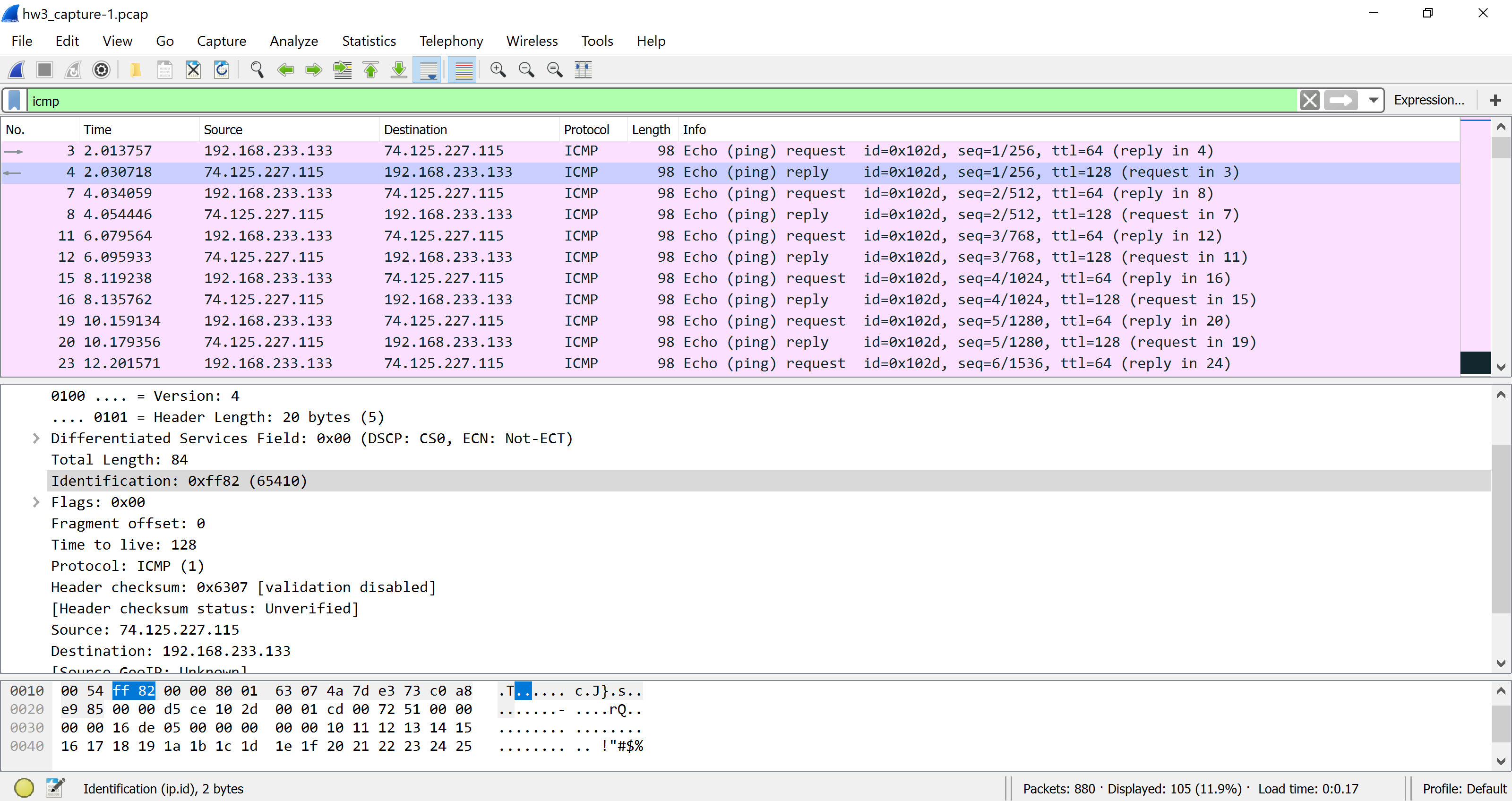
1. Which fields in the IP datagram always change from one datagram to the next within this series of ICMP messages sent by the computer?

**Answer :**

**Identification** field, **header checksum** field and **Time to live** field always change from one datagram to the next within this series of ICMP messages sent by the computer.

Here I have considered packets 3 and 4 , where packet 3 is ICMP echo request and packet 4 is ICMP echo reply. It shows that the identification , time to live and header checksum field changes .





1. Which fields stay constant? Which of the fields must stay constant? Which fields must change? Why?

**Answer :**

The fields that stay constant across the IP datagrams are Version, header length, source IP, destination IP, differentiated services and flags.

The fields that must stay constant are Version, header length, source IP, destination IP, differentiated services and flags.

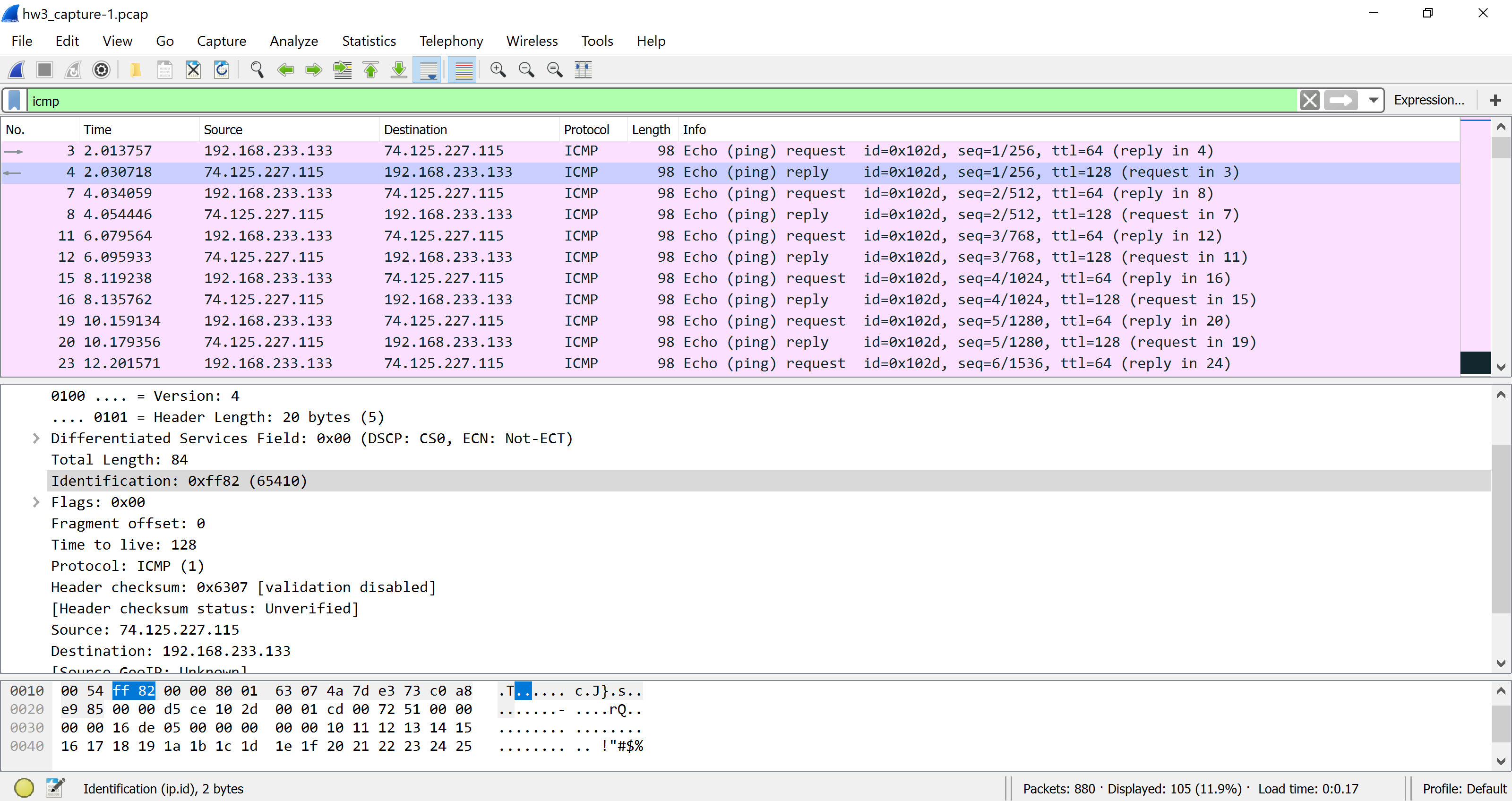
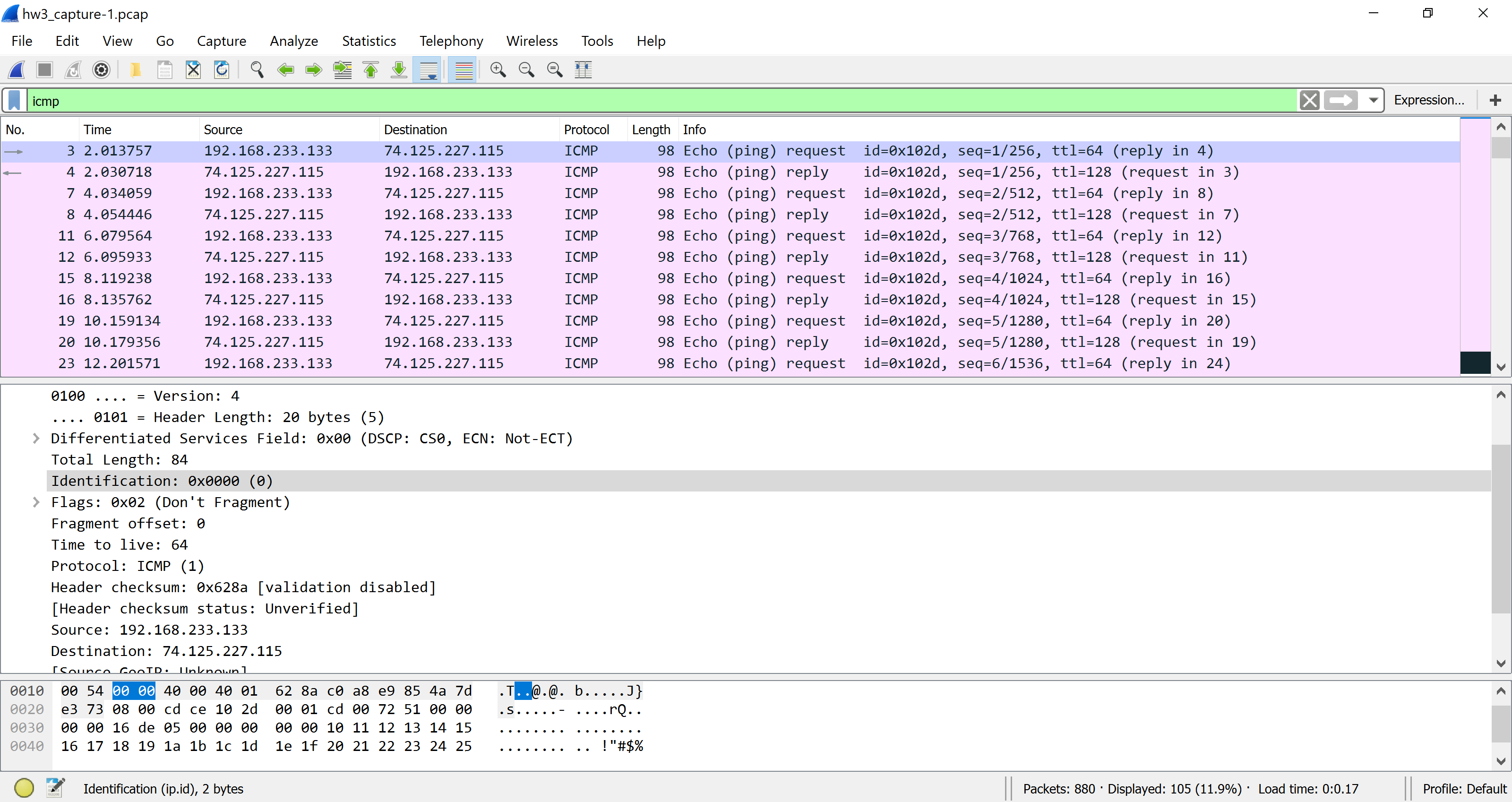
The fields that must change are Identification, Time to live and Header checksum.

If you notice the packets 3 and 4 ,

version is 4 in both the packets which is because we are using IPV4

Header length is 20 bytes because they are ICMP packets

Source and Destination IP addresses , where we are sending from same source and to the same destination.



1. Describe the pattern you see in the values in the Identification field and TTL field of the IP datagram in the series of ICMP TTL exceeded replies sent to the computer by the nearest (first hop) router.

**Answer :**

The value of identification field of IP datagram increases by 1 and then changes based on TTL. Identification field increases proportionally to the change in TTL field. The value in identification field changes per ICMP TTL exceed reply as 0xffe6, 0xffe7, 0xffe8, 0xffed, 0xffef, 0xfff4, 0xfff5 and 0xfff6 ; but the TTL field remains constant at 128.

The TTL field always stays the same, because TTL for the first hop router is always same. The identification field does not remain constant because it is identifying unique requests.

