NAME: S.PRIYADHARSHINI

ROLL NO.: CB.EN.P2CYS22010

DATE: 27.10.2022

INTERNET PROTOCOL LAB – 4

ANALYSING TCP AND UDP USING WIRESHARK

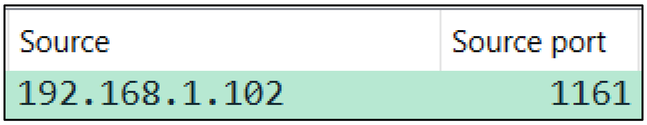
AIM:

To analyze TCP and UDP using Wireshark.

PROCEDURE:

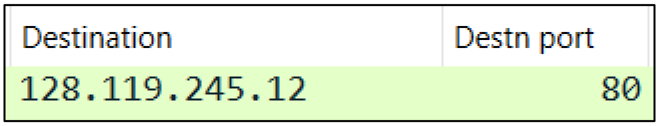
1.Open the pcap file "tcp" in Wireshark to answer the following questions.

a. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?



Source	Source port
192.168.1.102	1161

b. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

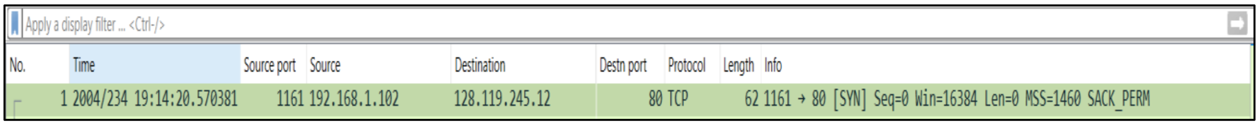


Destination	Destn port
128.119.245.12	80

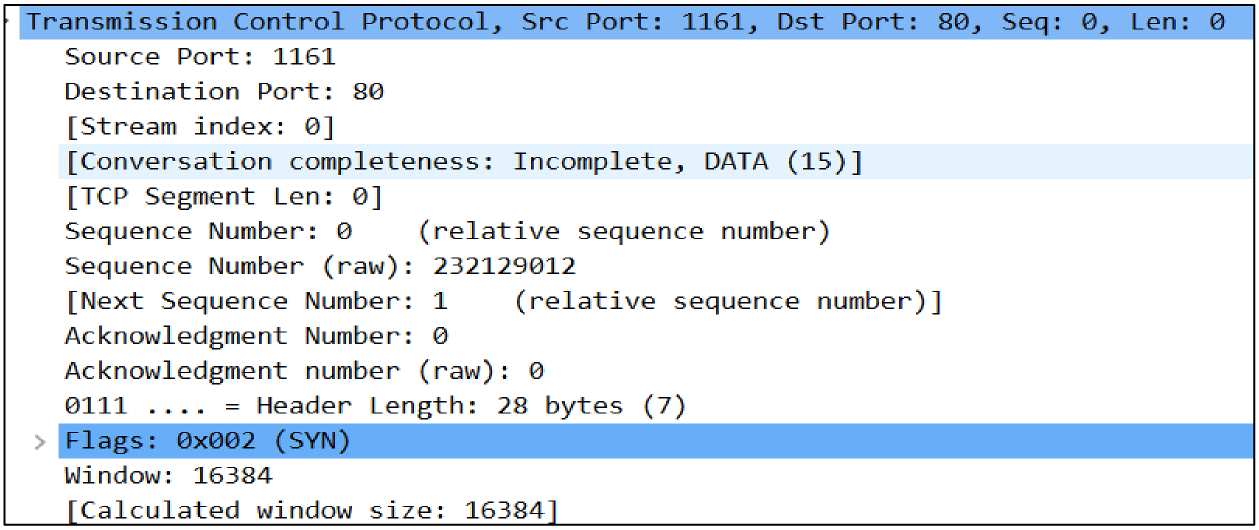
Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages rather than about the HTTP messages. To have Wireshark do this, select Analyze->Enabled Protocols. Then uncheck the HTTP box and select OK.

c. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

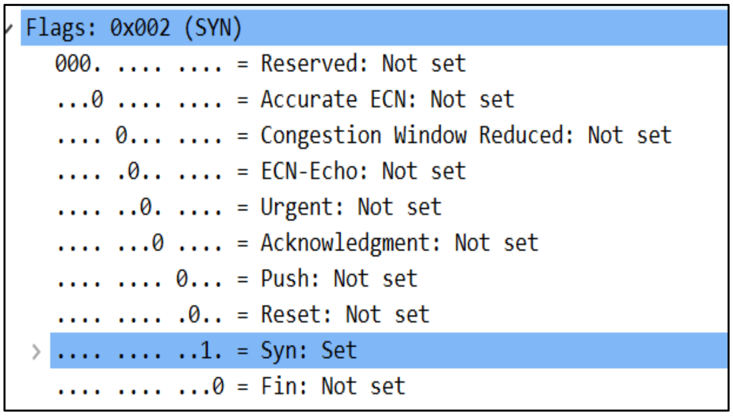
The first packet here initiates the TCP connection with Seq=0. This packet has 1 in SYN flag field.



No.	Time	Source port	Source	Destination	Destn port	Protocol	Length	Info
1	2004/234 19:14:20.570381	1161	192.168.1.102	128.119.245.12	80	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM



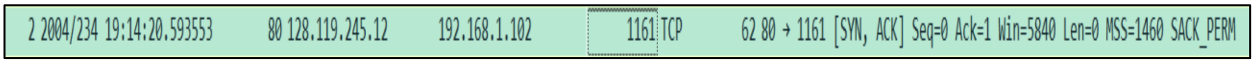
```
Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 1161
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 232129012
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    0111 .... = Header Length: 28 bytes (7)
  > Flags: 0x002 (SYN)
    Window: 16384
    [Calculated window size: 16384]
```



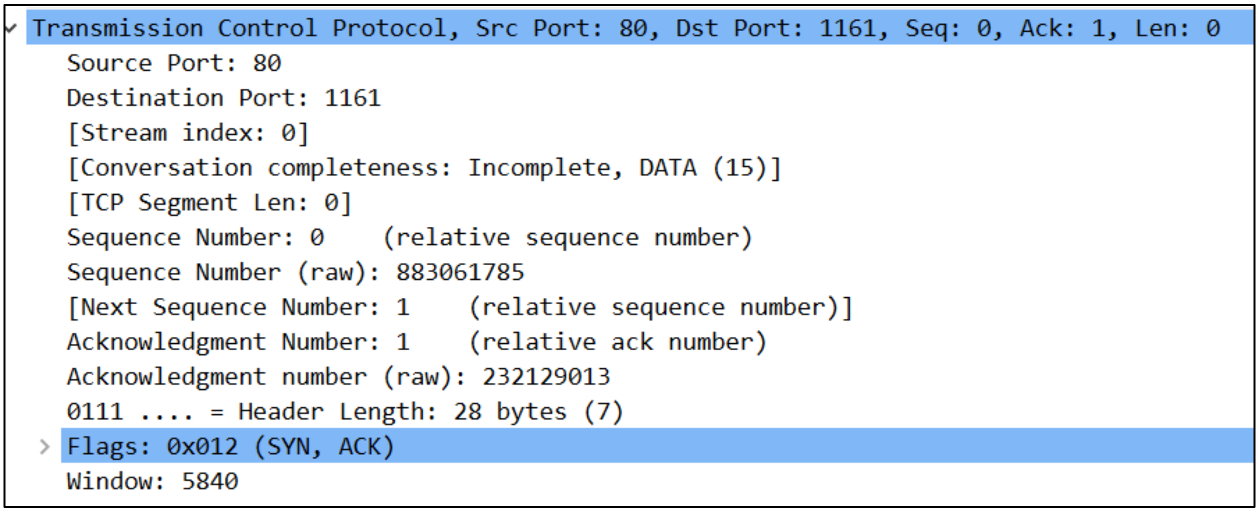
```
Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  > .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
```

d. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Seq = 0 , Ack = 1

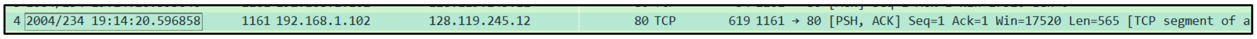


```
2 2004/234 19:14:20.593553    80 128.119.245.12    192.168.1.102    1161 TCP    62 80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
```



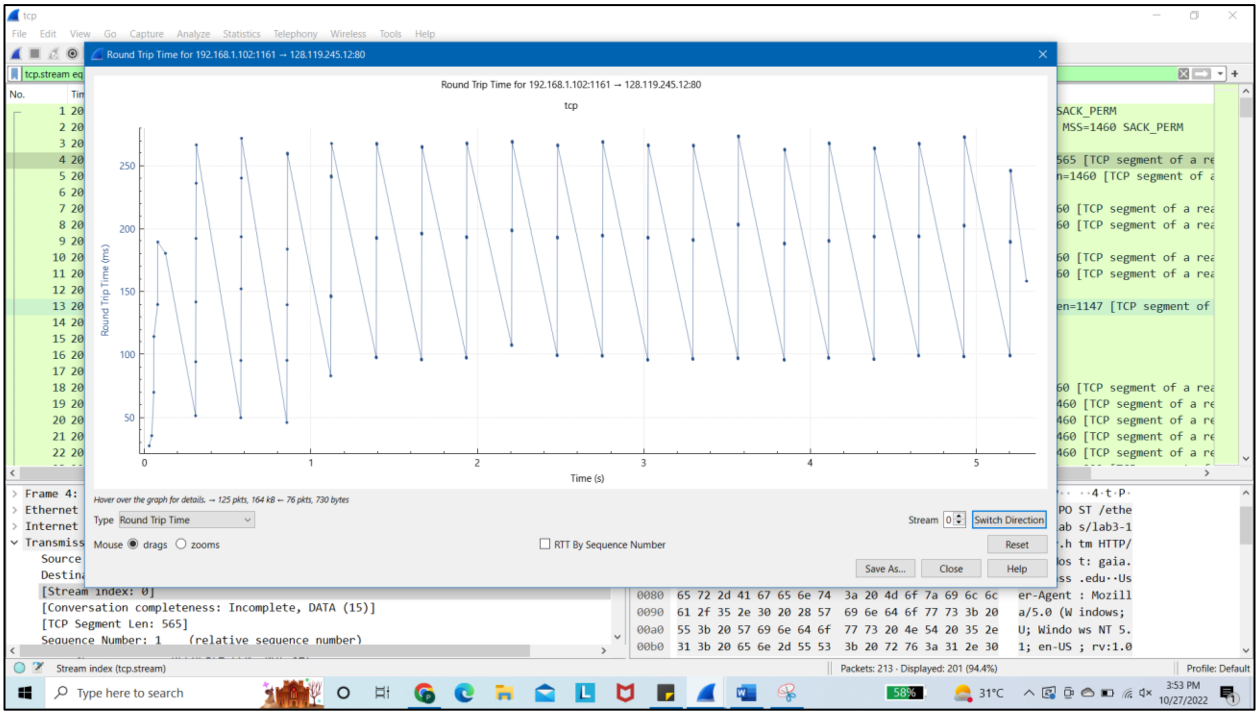
```
Transmission Control Protocol, Src Port: 80, Dst Port: 1161, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 1161
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 883061785
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 232129013
    0111 .... = Header Length: 28 bytes (7)
  > Flags: 0x012 (SYN, ACK)
    Window: 5840
```

e. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.



```
4 2004/234 19:14:20.596858    1161 192.168.1.102    128.119.245.12    80 TCP    619 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment of a
```

f. Plot the RTT graph using Wireshark.



Round Trip Time for 192.168.1.102:1161 → 128.119.245.12:80

Hover over the graph for details. → 125 pkts, 164 kB ← 76 pkts, 730 bytes

Type Round Trip Time

Mouse ● drags ○ zooms

□ RTT By Sequence Number

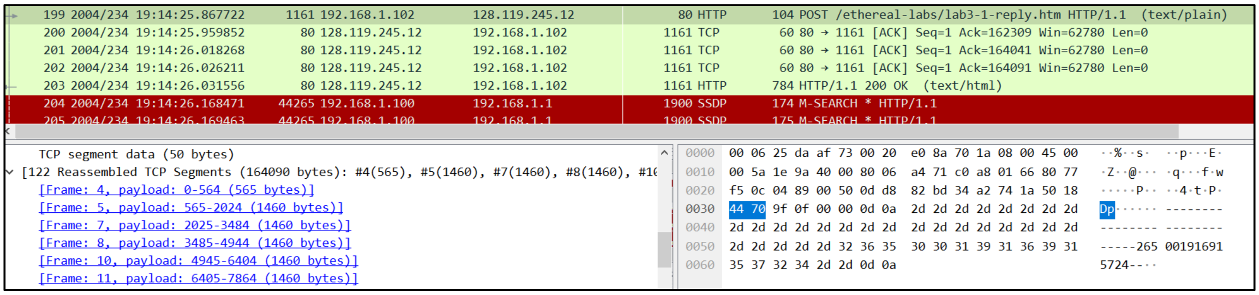
Stream 0

Packets: 213 · Displayed: 201 (94.4%)

In Wireshark go to statistics -> TCP stream graphs -> Round trip time.

(Select switch direction)

g. What is the length of each of the first six TCP segments (HTTP POST)?



```
199 2004/234 19:14:25.867722    1161 192.168.1.102    128.119.245.12      80 HTTP     104 POST /ethereal-labs/lab3-1-reply.htm HTTP/1.1  (text/plain)
200 2004/234 19:14:25.959852      80 128.119.245.12   192.168.1.102     1161 TCP       60 80 → 1161 [ACK] Seq=1 Ack=162309 Win=62780 Len=0
201 2004/234 19:14:26.018268      80 128.119.245.12   192.168.1.102     1161 TCP       60 80 → 1161 [ACK] Seq=1 Ack=164041 Win=62780 Len=0
202 2004/234 19:14:26.026211      80 128.119.245.12   192.168.1.102     1161 TCP       60 80 → 1161 [ACK] Seq=1 Ack=164091 Win=62780 Len=0
203 2004/234 19:14:26.031556      80 128.119.245.12   192.168.1.102     1161 HTTP     784 HTTP/1.1 200 OK  (text/html)
204 2004/234 19:14:26.168471   44265 192.168.1.100    192.168.1.1       1900 SSDP     174 M-SEARCH * HTTP/1.1
205 2004/234 19:14:26.169463   44265 192.168.1.100    192.168.1.1       1900 SSDP     175 M-SEARCH * HTTP/1.1
```

```
  TCP segment data (50 bytes)
⌄ [122 Reassembled TCP Segments (164090 bytes): #4(565), #5(1460), #7(1460), #8(1460), #1(
    [Frame: 4, payload: 0-564 (565 bytes)]
    [Frame: 5, payload: 565-2024 (1460 bytes)]
    [Frame: 7, payload: 2025-3484 (1460 bytes)]
    [Frame: 8, payload: 3485-4944 (1460 bytes)]
    [Frame: 10, payload: 4945-6404 (1460 bytes)]
    [Frame: 11, payload: 6405-7864 (1460 bytes)]
```

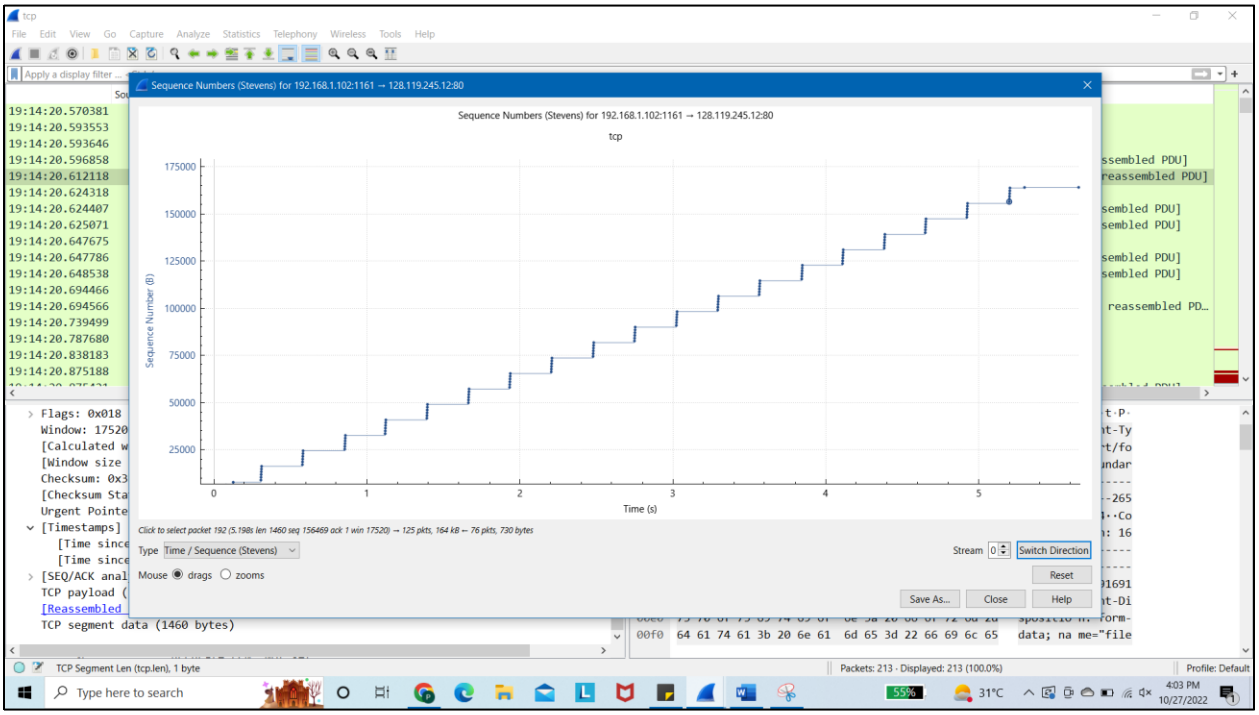
```
0000  00 06 25 da af 73 00 20  e0 8a 70 1a 08 00 45 00   ··%··s·  ··p···E·
0010  00 5a 1e 9a 40 00 80 06  a4 71 c0 a8 01 66 80 77   ·Z··@···  ·q···f·w
0020  f5 0c 04 89 00 50 0d d8  82 bd 34 a2 74 1a 50 18   ·····P··  ··4·t·P·
0030  44 70 9f 0f 00 00 0d 0a  2d 2d 2d 2d 2d 2d 2d 2d   Dp······  --------
0040  2d 2d 2d 2d 2d 2d 2d 2d  2d 2d 2d 2d 2d 2d 2d 2d   --------  --------
0050  2d 2d 2d 2d 2d 32 36 35  30 30 31 39 31 36 39 31   -----265 00191691
0060  35 37 32 34 2d 2d 0d 0a                            5724--··
```

For the length of first six TCP segments, we analyze the HTTP POST packet. There we can find the reassembled TCP segments and their lengths. From that the information of the first six TCP segments can be taken.

h. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

Statistics-> TCP stream graphs -> Time Sequence(Stevens)

In the graph if there is any decrease or drop it will represent the retransmitted packets. For retransmission packet with same sequence number will be transmitted so, there will be a drop. Here there is no drop. So in this capture no segment was retransmitted.


tcp
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ...
Sequence Numbers (Stevens) for 192.168.1.102:1161 → 128.119.245.12:80
Sequence Numbers (Stevens) for 192.168.1.102:1161 → 128.119.245.12:80
tcp
Sequence Number (B)
175000
150000
125000
100000
75000
50000
25000
0
1
2
3
4
5
Time (s)
Save As...
Close
Help
Reset
Switch Direction
Type here to search
55%
31°C
4:03 PM
10/27/2022

```
19:14:20.570381
19:14:20.593553
19:14:20.593646
19:14:20.596858
19:14:20.612118
19:14:20.624318
19:14:20.624407
19:14:20.625071
19:14:20.647675
19:14:20.647786
19:14:20.648538
19:14:20.694466
19:14:20.694566
19:14:20.739499
19:14:20.787680
19:14:20.838183
19:14:20.875188
```

```
ssembled PDU]
reassembled PDU]
sembled PDU]
sembled PDU]
sembled PDU]
sembled PDU]
 reassembled PD…
```

```
> Flags: 0x018
  Window: 17520
  [Calculated w
  [Window size
  Checksum: 0x3
  [Checksum Sta
  Urgent Pointe
⌄ [Timestamps]
    [Time since
    [Time since
> [SEQ/ACK anal
  TCP payload (
  [Reassembled
  TCP segment data (1460 bytes)
```

```
00f0  64 61 74 61 3b 20 6e 61  6d 65 3d 22 66 69 6c 65   data; na me="file
```

Click to select packet 192 (5.198s len 1460 seq 156469 ack 1 win 17520) → 125 pkts, 164 kB ← 76 pkts, 730 bytes

Type: Time / Sequence (Stevens)

Stream: 0

Mouse ● drags ○ zooms

TCP Segment Len (tcp.len), 1 byte

Packets: 213 · Displayed: 213 (100.0%)

Profile: Default

i.What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

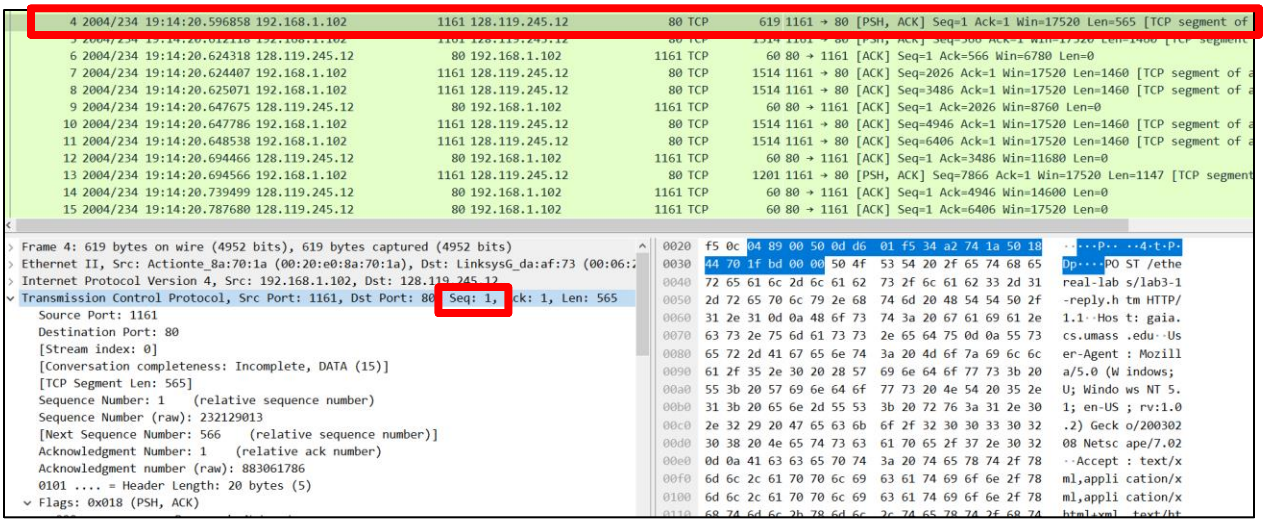
Throughput = total data transferred / total time taken.

Packet 4 and 202 is considered.

Total data transmitted = Ack of last tcp packet of this transfer – Seq of first tcp packet of this transfer.

= 164091 – 1

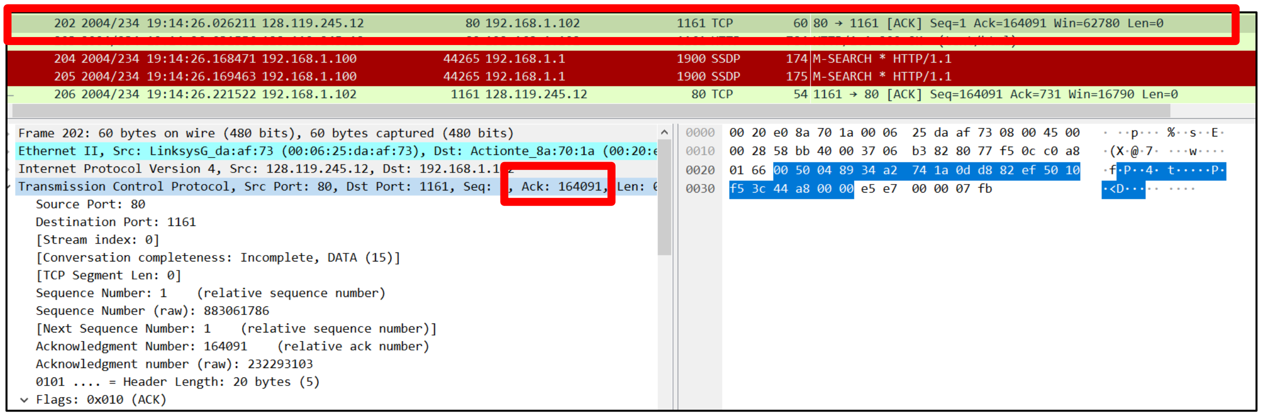
= 164090



```
  4 2004/234 19:14:20.596858 192.168.1.102    1161 128.119.245.12    80 TCP    619 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment of
  6 2004/234 19:14:20.624318 128.119.245.12     80 192.168.1.102   1161 TCP     60 80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
  7 2004/234 19:14:20.624407 192.168.1.102    1161 128.119.245.12    80 TCP   1514 1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP segment of a
  8 2004/234 19:14:20.625071 192.168.1.102    1161 128.119.245.12    80 TCP   1514 1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP segment of a
  9 2004/234 19:14:20.647675 128.119.245.12     80 192.168.1.102   1161 TCP     60 80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
 10 2004/234 19:14:20.647786 192.168.1.102    1161 128.119.245.12    80 TCP   1514 1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP segment of a
 11 2004/234 19:14:20.648538 192.168.1.102    1161 128.119.245.12    80 TCP   1514 1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP segment of a
 12 2004/234 19:14:20.694466 128.119.245.12     80 192.168.1.102   1161 TCP     60 80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
 13 2004/234 19:14:20.694566 192.168.1.102    1161 128.119.245.12    80 TCP   1201 1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 [TCP segment
 14 2004/234 19:14:20.739499 128.119.245.12     80 192.168.1.102   1161 TCP     60 80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
 15 2004/234 19:14:20.787680 128.119.245.12     80 192.168.1.102   1161 TCP     60 80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
```

```
> Frame 4: 619 bytes on wire (4952 bits), 619 bytes captured (4952 bits)
> Ethernet II, Src: Actionte_8a:70:1a (00:20:e0:8a:70:1a), Dst: LinksysG_da:af:73 (00:06:2
> Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
v Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 1, Ack: 1, Len: 565
    Source Port: 1161
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 565]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 232129013
    [Next Sequence Number: 566    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 883061786
    0101 .... = Header Length: 20 bytes (5)
  v Flags: 0x018 (PSH, ACK)
```

```
0020  f5 0c 04 89 00 50 0d d6  01 f5 34 a2 74 1a 50 18   ·····P·· ··4·t·P·
0030  44 70 1f bd 00 00 50 4f  53 54 20 2f 65 74 68 65   Dp····PO ST /ethe
0040  72 65 61 6c 2d 6c 61 62  73 2f 6c 61 62 33 2d 31   real-lab s/lab3-1
0050  2d 72 65 70 6c 79 2e 68  74 6d 20 48 54 54 50 2f   -reply.h tm HTTP/
0060  31 2e 31 0d 0a 48 6f 73  74 3a 20 67 61 69 61 2e   1.1··Hos t: gaia.
0070  63 73 2e 75 6d 61 73 73  2e 65 64 75 0d 0a 55 73   cs.umass .edu··Us
0080  65 72 2d 41 67 65 6e 74  3a 20 4d 6f 7a 69 6c 6c   er-Agent : Mozill
0090  61 2f 35 2e 30 20 28 57  69 6e 64 6f 77 73 3b 20   a/5.0 (W indows;
00a0  55 3b 20 57 69 6e 64 6f  77 73 20 4e 54 20 35 2e   U; Windo ws NT 5.
00b0  31 3b 20 65 6e 2d 55 53  3b 20 72 76 3a 31 2e 30   1; en-US ; rv:1.0
00c0  2e 32 29 20 47 65 63 6b  6f 2f 32 30 30 33 30 32   .2) Geck o/200302
00d0  30 38 20 4e 65 74 73 63  61 70 65 2f 37 2e 30 32   08 Netsc ape/7.02
00e0  0d 0a 41 63 63 65 70 74  3a 20 74 65 78 74 2f 78   ··Accept : text/x
00f0  6d 6c 2c 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78   ml,appli cation/x
0100  6d 6c 2c 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78   ml,appli cation/x
```



```
202 2004/234 19:14:26.026211 128.119.245.12     80 192.168.1.102   1161 TCP     60 80 → 1161 [ACK] Seq=1 Ack=164091 Win=62780 Len=0
204 2004/234 19:14:26.168471 192.168.1.100   44265 192.168.1.1     1900 SSDP   174 M-SEARCH * HTTP/1.1
205 2004/234 19:14:26.169463 192.168.1.100   44265 192.168.1.1     1900 SSDP   175 M-SEARCH * HTTP/1.1
206 2004/234 19:14:26.221522 192.168.1.102    1161 128.119.245.12    80 TCP     54 1161 → 80 [ACK] Seq=164091 Ack=731 Win=16790 Len=0
```

```
Frame 202: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: LinksysG_da:af:73 (00:06:25:da:af:73), Dst: Actionte_8a:70:1a (00:20:e
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.1
Transmission Control Protocol, Src Port: 80, Dst Port: 1161, Seq: , Ack: 164091, Len: 0
   Source Port: 80
   Destination Port: 1161
   [Stream index: 0]
   [Conversation completeness: Incomplete, DATA (15)]
   [TCP Segment Len: 0]
   Sequence Number: 1    (relative sequence number)
   Sequence Number (raw): 883061786
   [Next Sequence Number: 1    (relative sequence number)]
   Acknowledgment Number: 164091    (relative ack number)
   Acknowledgment number (raw): 232293103
   0101 .... = Header Length: 20 bytes (5)
 v Flags: 0x010 (ACK)
```

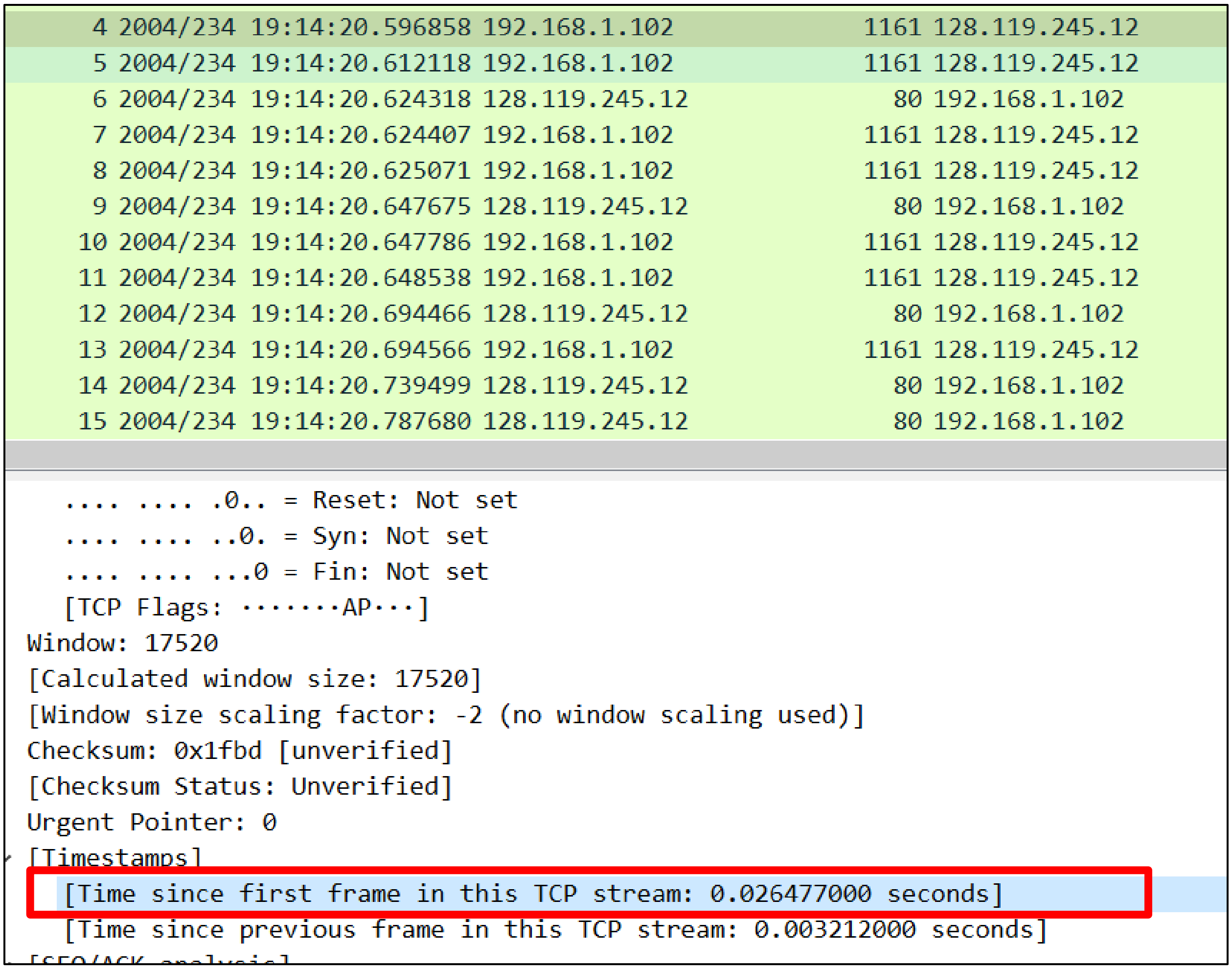
```
0000  00 20 e0 8a 70 1a 00 06  25 da af 73 08 00 45 00   · ··p··· %··s··E·
0010  00 28 58 bb 40 00 37 06  b3 82 80 77 f5 0c c0 a8   ·(X·@·7· ···w····
0020  01 66 00 50 04 89 34 a2  74 1a 0d d8 82 ef 50 10   ·f·P··4· t·····P·
0030  f5 3c 44 a8 00 00 e5 e7  00 00 07 fb               ·<D····· ····
```

Total time taken = 5.455830000 – 0.026477000

= 5.429353

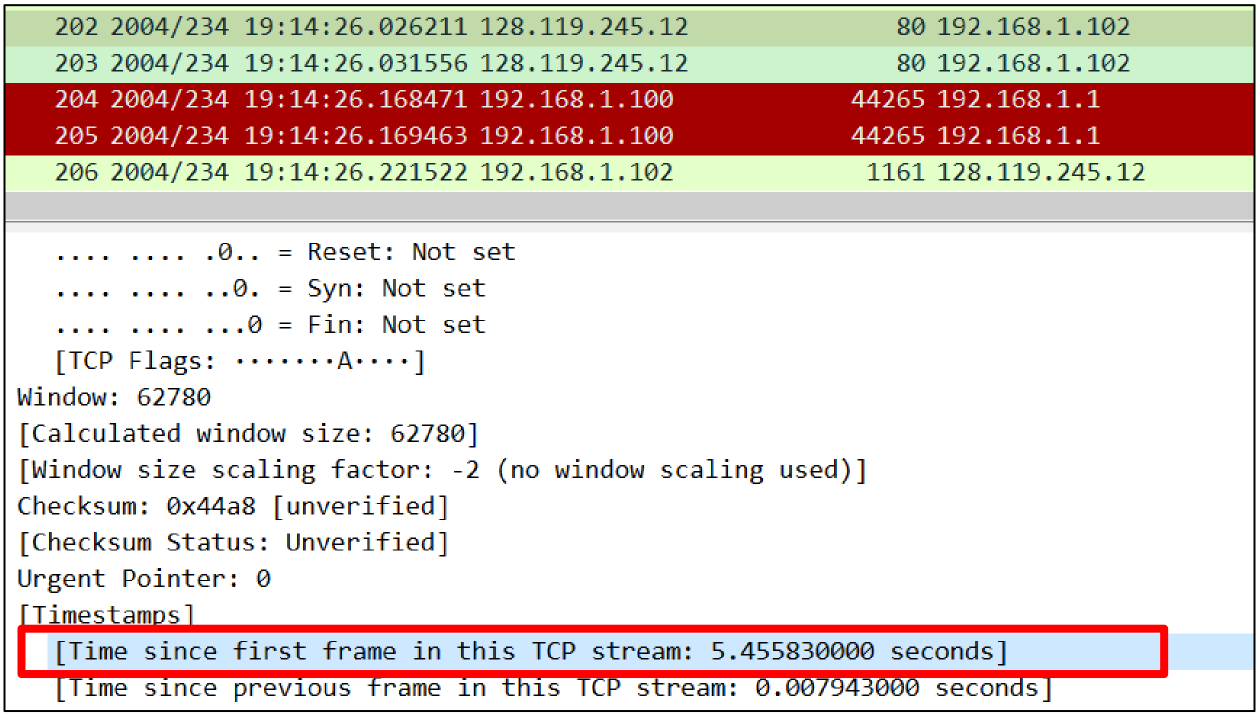
Throughput = 164090 / 5.429353

= 30.22 KB



```
      4 2004/234 19:14:20.596858 192.168.1.102              1161 128.119.245.12
      5 2004/234 19:14:20.612118 192.168.1.102              1161 128.119.245.12
      6 2004/234 19:14:20.624318 128.119.245.12               80 192.168.1.102
      7 2004/234 19:14:20.624407 192.168.1.102              1161 128.119.245.12
      8 2004/234 19:14:20.625071 192.168.1.102              1161 128.119.245.12
      9 2004/234 19:14:20.647675 128.119.245.12               80 192.168.1.102
     10 2004/234 19:14:20.647786 192.168.1.102              1161 128.119.245.12
     11 2004/234 19:14:20.648538 192.168.1.102              1161 128.119.245.12
     12 2004/234 19:14:20.694466 128.119.245.12               80 192.168.1.102
     13 2004/234 19:14:20.694566 192.168.1.102              1161 128.119.245.12
     14 2004/234 19:14:20.739499 128.119.245.12               80 192.168.1.102
     15 2004/234 19:14:20.787680 128.119.245.12               80 192.168.1.102
```

```
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·······AP···]
  Window: 17520
  [Calculated window size: 17520]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x1fbd [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
    [Time since first frame in this TCP stream: 0.026477000 seconds]
    [Time since previous frame in this TCP stream: 0.003212000 seconds]
  [SEQ/ACK analysis]
```



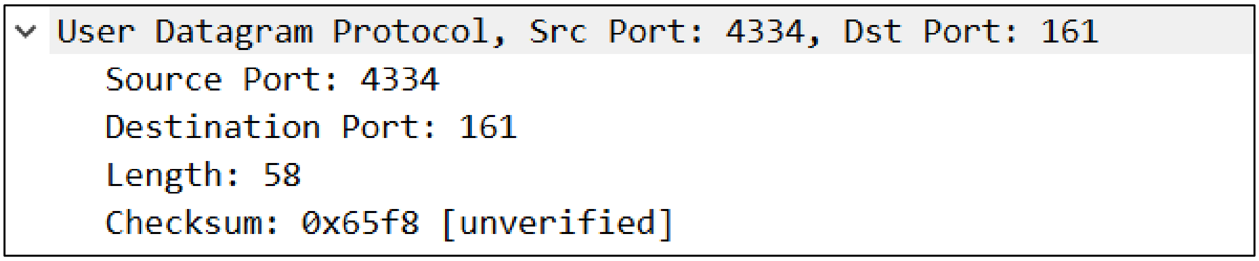
```
    202 2004/234 19:14:26.026211 128.119.245.12             80 192.168.1.102
    203 2004/234 19:14:26.031556 128.119.245.12             80 192.168.1.102
    204 2004/234 19:14:26.168471 192.168.1.100           44265 192.168.1.1
    205 2004/234 19:14:26.169463 192.168.1.100           44265 192.168.1.1
    206 2004/234 19:14:26.221522 192.168.1.102            1161 128.119.245.12
```

```
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·······A····]
  Window: 62780
  [Calculated window size: 62780]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x44a8 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
    [Time since first frame in this TCP stream: 5.455830000 seconds]
    [Time since previous frame in this TCP stream: 0.007943000 seconds]
```

2. Open the pcap file “udp” in Wireshark to answer the following questions

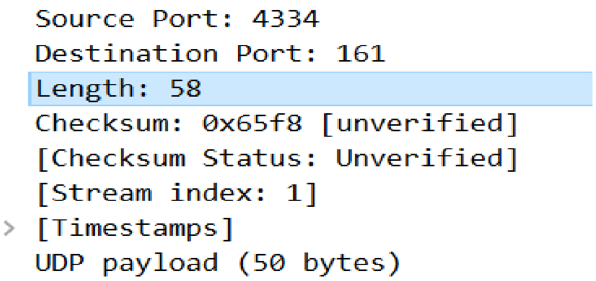
j. Select one UDP packet from your trace. From this packet, determine how many fields are there in the UDP header. Name these fields.

There are 4 fields: source port, destination port, length, checksum.



```
v User Datagram Protocol, Src Port: 4334, Dst Port: 161
      Source Port: 4334
      Destination Port: 161
      Length: 58
      Checksum: 0x65f8 [unverified]
```

k. By consulting the displayed information in Wireshark’s packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.

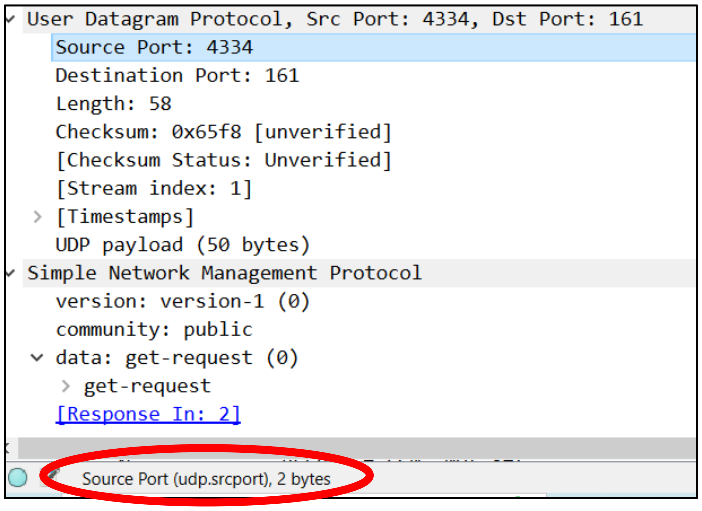


```
  Source Port: 4334
  Destination Port: 161
  Length: 58
  Checksum: 0x65f8 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 1]
> [Timestamps]
  UDP payload (50 bytes)
```

Length = 58

UDP payload = 50 bytes

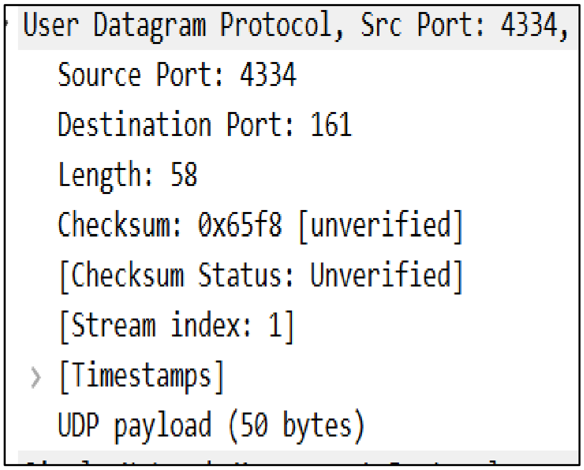
58 – 50 = 8 bytes. Therefore, all 4 fields are of 2 bytes each.



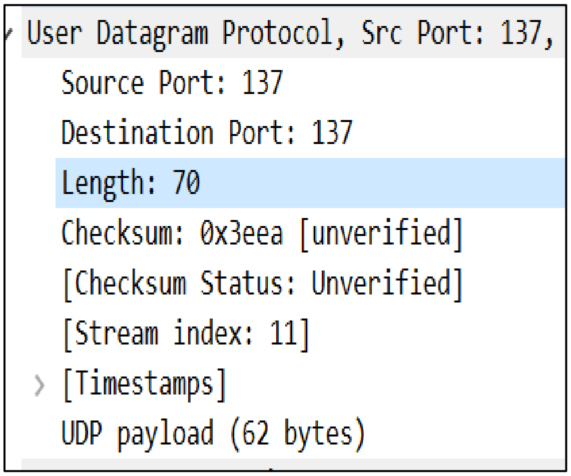
```
v User Datagram Protocol, Src Port: 4334, Dst Port: 161
    Source Port: 4334
    Destination Port: 161
    Length: 58
    Checksum: 0x65f8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
  > [Timestamps]
    UDP payload (50 bytes)
v Simple Network Management Protocol
    version: version-1 (0)
    community: public
  v data: get-request (0)
      > get-request
    [Response In: 2]

Source Port (udp.srcport), 2 bytes
```

We can also refer from the above picture that it displays the length in the bottom of the page when the particular field is selected.

l. The value in the Length field is the length of what? Verify your claim with your captured UDP packet.



```
User Datagram Protocol, Src Port: 4334,
    Source Port: 4334
    Destination Port: 161
    Length: 58
    Checksum: 0x65f8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
  > [Timestamps]
    UDP payload (50 bytes)
```



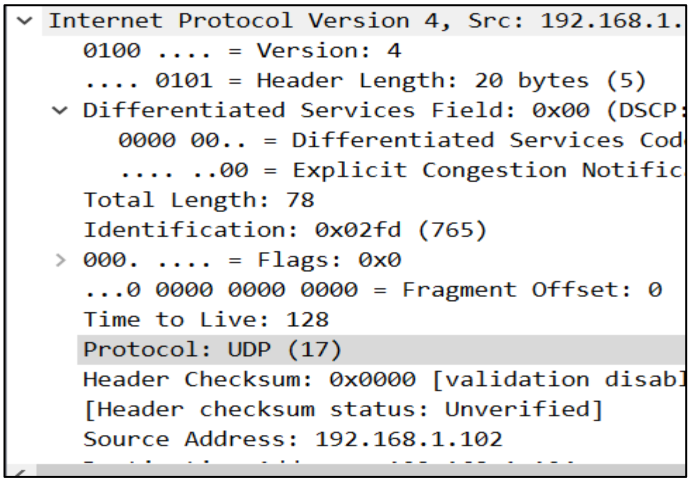
```
User Datagram Protocol, Src Port: 137,
    Source Port: 137
    Destination Port: 137
    Length: 70
    Checksum: 0x3eea [unverified]
    [Checksum Status: Unverified]
    [Stream index: 11]
  > [Timestamps]
    UDP payload (62 bytes)
```

Length field represents the total length of the UDP header. UDP payload represents the data transmitted. Therefore, Length – UDP payload gives the total bytes of the fields in UDP header.

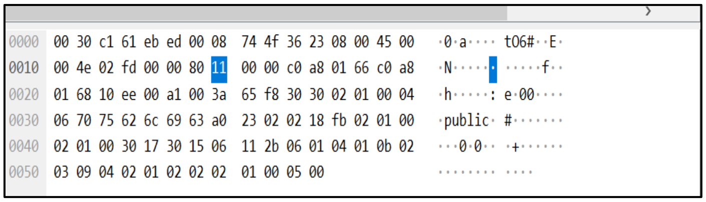
58 – 50 = 8 and 70 – 62 = 8. Therefore, all 4 fields in udp header are of 2 bytes each.

m. What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation.

Protocol number = 17, hexadecimal notation = 0x11

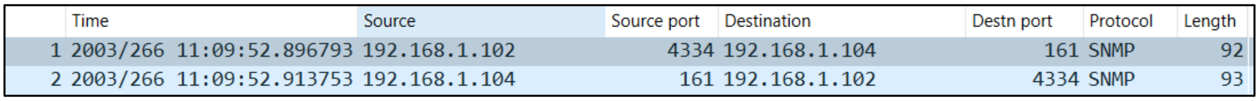


```
v Internet Protocol Version 4, Src: 192.168.1.
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    v Differentiated Services Field: 0x00 (DSCP:
          0000 00.. = Differentiated Services Cod
          .... ..00 = Explicit Congestion Notific
      Total Length: 78
      Identification: 0x02fd (765)
    > 000. .... = Flags: 0x0
      ...0 0000 0000 0000 = Fragment Offset: 0
      Time to Live: 128
      Protocol: UDP (17)
      Header Checksum: 0x0000 [validation disabl
      [Header checksum status: Unverified]
      Source Address: 192.168.1.102
```



```
0000  00 30 c1 61 eb ed 00 08  74 4f 36 23 08 00 45 00   ·0·a····  tO6#··E·
0010  00 4e 02 fd 00 00 80 11  00 00 c0 a8 01 66 c0 a8   ·N······  ·····f··
0020  01 68 10 ee 00 a1 00 3a  65 f8 30 30 02 01 00 04   ·h·····:  e·00····
0030  06 70 75 62 6c 69 63 a0  23 02 02 18 fb 02 01 00   ·public·  #·······
0040  02 01 00 30 17 30 15 06  11 2b 06 01 04 01 0b 02   ···0·0··  ·+······
0050  03 09 04 02 01 02 02 02  01 00 05 00               ········  ····
```

n. Examine a pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). Describe the relationship between the port numbers in the two packets.



	Time	Source	Source port	Destination	Destn port	Protocol	Length
1	2003/266 11:09:52.896793	192.168.1.102	4334	192.168.1.104	161	SNMP	92
2	2003/266 11:09:52.913753	192.168.1.104	161	192.168.1.102	4334	SNMP	93

Since second packet is a reply to the first UDP packet, we can see that the source port number of the request will be the destination port number of the reply. The destination port number of the request will be the source port number of the reply.

RESULT:

Thus, TCP and UDP protocols have been analyzed successfully using Wireshark.