

# Southern University of Science and Technology

## Computer Networking Lab Report

Name: 吴培霖 Student Number: 11711717

Major: None

Time: 2018/11/10

### Introduction:

#### Assignment 6.1

Select one UDP packet from your trace.

1. From this packet, determine how many fields there are in the UDP header. Name these fields.
2. 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.
3. The value in the Length field is the length of what? Verify your claim with your captured UDP packet.
4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)
5. What is the largest possible source port number? (Hint: see the hint in 4.)
6. What is the protocol number for UDP?( Give your answer in both hexadecimal and decimal notation. )

•

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.

1. Describe the relationship between the port numbers in the two packets.

#### Assignment 6.2 TCP

Finish the question 3~10 question of Wireshark\_TCP\_v7.0.pdf

## **Procedure:**

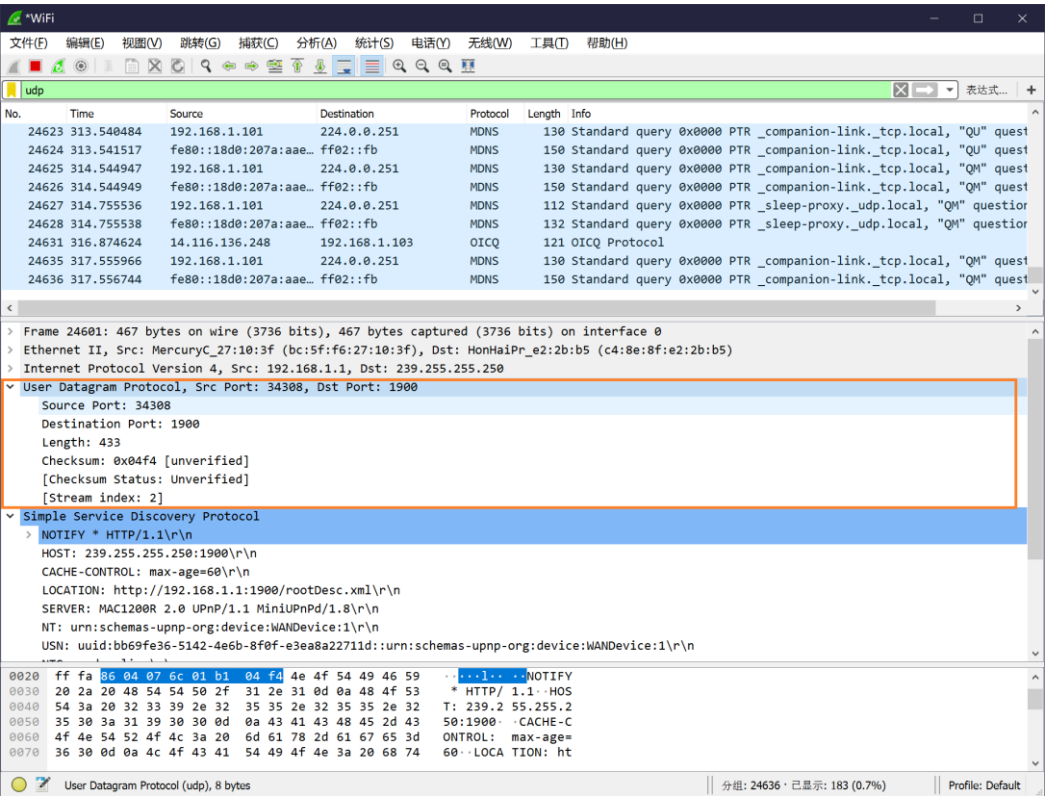
### **Assignment 6.1**

1. Open the Wireshark, select WiFi to monitor, then type 'udp' to filter the message.
2. Then we can see the result in the fig.1 showed in result. The message in the User Datagram Protocol is the head of UDP header.
3. After search in the Internet showed in the fig.2, we know the information of each header fields.
4. The hexadecimal data is showed in the fig.3.
5. We can find the protocol number of UDP in the IP field in the fig.4.

### **Assignment 6.2 TCP**

1. First we download the alice.txt from <http://gaia.cs.umass.edu/wiresharklabs/alice.txt>.
2. Next go to <http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>.. Attach alice.txt to the browser(fig.5). Open the Wireshark to start the package capture. Click upload (fig.6), then stop Wireshark capture.
3. Then filter TCP result, we can find many packages show in the Wireshark. (fig7)

Result:



(fig.1 The header field of this UDP package)

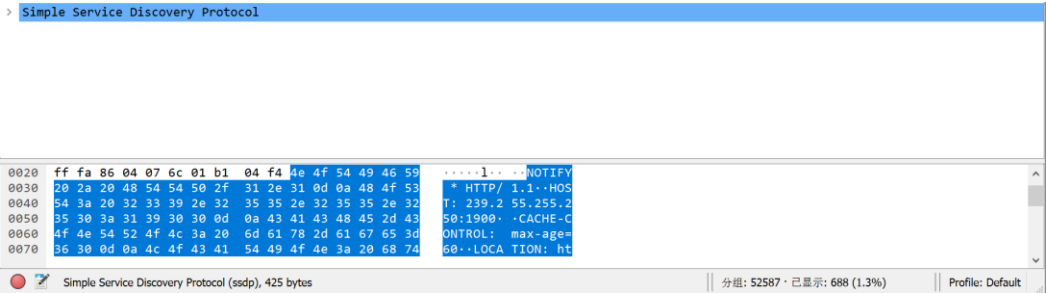
**Source port number**  
This field identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.<sup>[4]</sup>

**Destination port number**  
This field identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.<sup>[4]</sup>

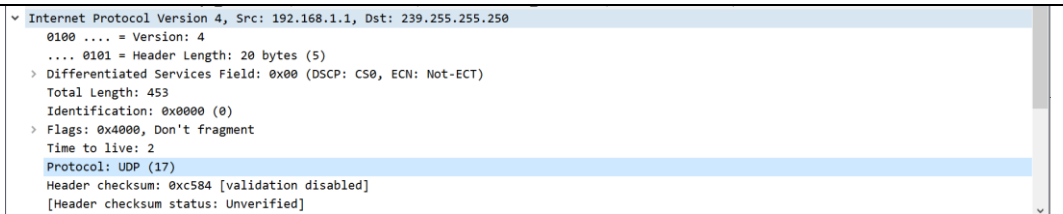
**Length**  
A field that specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes because that is the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. However the actual limit for the data length, which is imposed by the underlying IPv4 protocol, is 65,507 bytes (65,535 - 8 byte UDP header - 20 byte IP header).<sup>[4]</sup>  
In IPv6 jumbograms it is possible to have UDP packets of size greater than 65,535 bytes.<sup>[5]</sup> RFC 2675<sup>[6]</sup> specifies that the length field is set to zero if the length of the UDP header plus UDP data is greater than 65,535.

**Checksum**  
The checksum field may be used for error-checking of the header and data. This field is optional in IPv4, and mandatory in IPv6.<sup>[6]</sup> The field carries all-zeros if unused.<sup>[7]</sup>

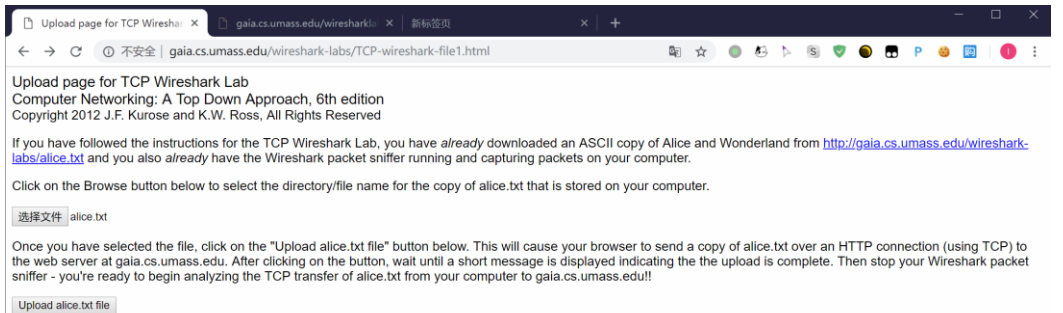
(fig.2 The information found in the Wiki)



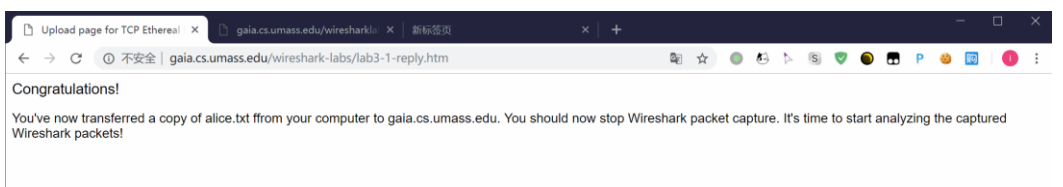
(fig.3 The hexadecimal data of the content of this UDP package \*not all data are showed above\*)



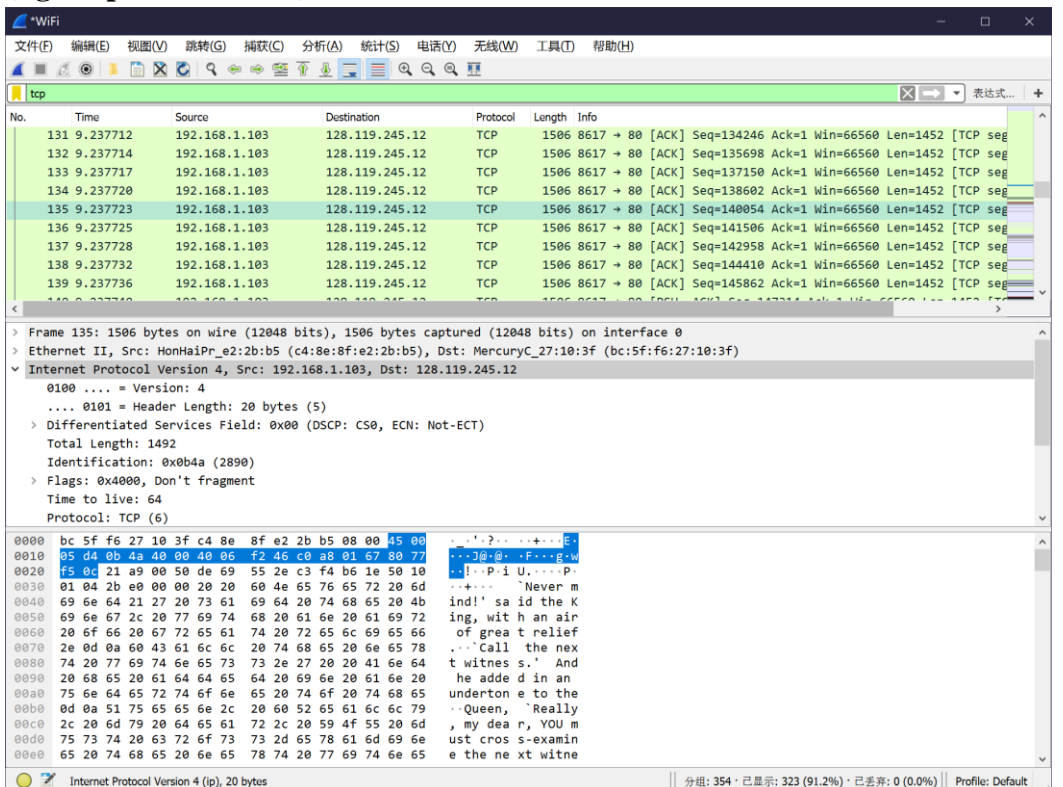
(fig.4 The protocol number of UDP)



(fig.5 Attach alice.txt to the browser)



(fig.6 Upload alice.txt)



(fig.7 Packages captured by Wireshark)

```

Source: 192.168.1.103
Destination: 128.119.245.12
Transmission Control Protocol, Src Port: 8617, Dst Port: 80, Seq: 140054, Ack: 1, Len: 1452
Source Port: 8617
Destination Port: 80

```

(fig.8 Source IP and source port)

```

Transmission Control Protocol, Src Port: 8616, Dst Port: 80, Seq: 0, Len: 0
Source Port: 8616
Destination Port: 80
[Stream index: 3]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
[Next sequence number: 0 (relative sequence number)]
Acknowledgment number: 0
1000 .... = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)

```

(fig.9 Sequence number of SYN segment)

```

Flags: 0x002 (SYN)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): 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

```

(fig.10 Flag field identify the segment as a SYN segment)

Seq	Source	Destination	Protocol	Source Port	Destination Port	Length	Flags	Window	Seq	Ack	Win	Len	MSS	SACK_PERM	WS
16	8.532508	128.119.245.12	192.168.1.103	TCP	66	80	→ 8617	[SYN, ACK]	Seq=0	Ack=1	Win=29200	Len=0	MSS=1452	SACK_PERM=1	WS=128
17	8.532635	192.168.1.103	128.119.245.12	TCP	54	8617	→ 80	[ACK]	Seq=1	Ack=1	Win=66560	Len=0			
18	8.533700	192.168.1.103	128.119.245.12	TCP	715	8617	→ 80	[PSH, ACK]	Seq=1	Ack=1	Win=66560	Len=661			[TCP segment of a reasse...
19	8.534080	192.168.1.103	128.119.245.12	TCP	1506	8617	→ 80	[ACK]	Seq=662	Ack=1	Win=66560	Len=1452			[TCP segment of a reasse...

```

Transmission Control Protocol, Src Port: 80, Dst Port: 8617, Seq: 0, Ack: 1, Len: 0
Source Port: 80
Destination Port: 8617
[Stream index: 4]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
[Next sequence number: 0 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK)

```

(fig.11 The sequence of SYNACK)

```

Destination: 192.168.1.103
Transmission Control Protocol, Src Port: 80, Dst Port: 8617, Seq: 0, Ack: 1, Len: 0
Source Port: 80
Destination Port: 8617
[Stream index: 4]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
[Next sequence number: 0 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
1000 .... = Header Length: 32 bytes (8)

```

(fig.12 The acknowledgement field of SYNACK)

```

1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... 0... = ECN-Echo: Not set
.... 0... = Urgent: Not set
...1 .... = Acknowledgment: Set
.... 0... = Push: Not set
.... 0... = Reset: Not set

```

(fig.13 Flag field)

Seq	Source	Destination	Protocol	Source Port	Destination Port	Length	Flags	Window	Seq	Ack	Win	Len	
18	8.533700	192.168.1.103	128.119.245.12	TCP	715	8617	→ 80	[PSH, ACK]	Seq=1	Ack=1	Win=66560	Len=661	[TCP segment of a reasse...
19	8.534080	192.168.1.103	128.119.245.12	TCP	1506	8617	→ 80	[ACK]	Seq=662	Ack=1	Win=66560	Len=1452	[TCP segment of a reasse...
20	8.534087	192.168.1.103	128.119.245.12	TCP	1506	8617	→ 80	[ACK]	Seq=2114	Ack=1	Win=66560	Len=1452	[TCP segment of a reasse...
21	8.534092	192.168.1.103	128.119.245.12	TCP	1506	8617	→ 80	[ACK]	Seq=3566	Ack=1	Win=66560	Len=1452	[TCP segment of a reasse...
22	8.534095	192.168.1.103	128.119.245.12	TCP	1506	8617	→ 80	[ACK]	Seq=5018	Ack=1	Win=66560	Len=1452	[TCP segment of a reasse...

```

Transmission Control Protocol, Src Port: 8617, Dst Port: 80, Seq: 1, Ack: 1, Len: 661
Source Port: 8617
Destination Port: 80
[Stream index: 4]
[TCP Segment Len: 661]
Sequence number: 1 (relative sequence number)
[Next sequence number: 662 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
0101 .... = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
000. .... = Reserved: Not set
0020 f5 0c 21 a0 00 50 da 67 32 19 c3 f4 b6 1a 50 18 ...!..P.g.2....P
0030 01 04 8a 81 00 00 50 4f 53 54 20 2f 69 72 65 .....PO ST /wine

```

(fig.14 The sequence number of the TCP segment contain HTTP POST)

0030	01 04 8a 81 00 00 50 4f 53 54 20 2f 77 69 72 65	.....PO ST /win
0040	73 68 61 72 6b 2d 6c 61 62 73 2f 6c 61 62 33 2d	shark.la bs/lab3.
0050	31 2d 72 65 70 6c 79 2e 68 74 6d 20 48 54 54 50	i-reply. htm HTTP
0060	2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 67 61 69 61	/1.1..Ho st: gaia
0070	2e 63 73 2e 75 6d 61 73 73 2e 65 64 75 0d 0a 43	.cs.umass s.edu..C
0080	6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d	connectio n: keep-
0090	61 6c 69 76 65 0d 0a 43 6f 6e 74 65 6e 74 2d 4c	alive..C ontent-L
00a0	55 6e 67 74 68 3a 20 31 35 32 33 31 39 0d 0a 43	length: 1 52319..C
00b0	61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d 61	ache-Con trol: ma
00c0	78 2d 61 67 65 3d 30 0d 0a 4f 72 69 67 69 6e 3a	x-age=0. -Origin:
00d0	20 68 74 74 70 3a 2f 2f 67 61 69 61 2e 63 73 2e	http:// gaia.cs.
00e0	75 6d 61 73 73 2e 65 64 75 0d 0a 55 70 67 72 61	umass.ed u..Upgra

(fig.15 POST is in the line 0030)

18	8.533700	192.168.1.103	128.119.245.12	TCP	715 8617 → 80 [PSH, ACK] Seq=1 Ack=1 Win=66560 Len=661 [TCP segment of a reasemb...
19	8.534080	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=662 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
20	8.534087	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=2114 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
21	8.534092	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=3566 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
22	8.534095	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=5018 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
23	8.534098	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=6470 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
24	8.534101	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=7922 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...

(fig.16 Sequence number of the first six segments: No.35~40)

46	0.277148	192.168.1.103	128.119.245.12	TCP	54 10454 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
47	0.516876	128.119.245.12	192.168.1.103	TCP	54 80 → 10453 [ACK] Seq=1 Ack=652 Win=30592 Len=0
48	0.516877	128.119.245.12	192.168.1.103	TCP	54 80 → 10453 [ACK] Seq=1 Ack=2104 Win=33536 Len=0
49	0.516878	128.119.245.12	192.168.1.103	TCP	54 80 → 10453 [ACK] Seq=1 Ack=9364 Win=48000 Len=0
50	0.516878	128.119.245.12	192.168.1.103	TCP	54 80 → 10453 [ACK] Seq=1 Ack=13720 Win=56704 Len=0

(fig.17 The time when ACK for each segment received)

27	8.534109	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=12278 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
28	8.535885	128.119.245.12	192.168.1.103	TCP	66 80 → 8616 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1452 SACK_PERM=1 WS=128
29	8.535991	192.168.1.103	128.119.245.12	TCP	54 8616 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0
30	8.765298	128.119.245.12	192.168.1.103	TCP	54 80 → 8617 [ACK] Seq=1 Ack=662 Win=30592 Len=0
31	0.766955	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=13790 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
32	8.767774	128.119.245.12	192.168.1.103	TCP	54 80 → 8617 [ACK] Seq=1 Ack=7922 Win=45056 Len=0
33	8.767774	128.119.245.12	192.168.1.103	TCP	54 80 → 8617 [ACK] Seq=1 Ack=10826 Win=50944 Len=0
34	8.767775	128.119.245.12	192.168.1.103	TCP	66 [TCP Window Update] 80 → 8617 [ACK] Seq=1 Ack=10826 Win=53888 Len=0 SLE=122...

(fig.18 The length of first six segment)

50	0.517035	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=20700 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
57	0.517039	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=22432 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
58	0.517042	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=23884 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
59	0.517046	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=25336 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
60	0.517049	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=26788 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
61	0.517053	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=28240 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
62	0.517056	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=29692 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
63	0.517059	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=31144 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
64	0.517064	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=32596 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
65	0.517067	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=34048 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
66	0.517071	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=35500 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
67	0.517080	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=36952 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...
68	0.517083	192.168.1.103	128.119.245.12	TCP	1506 10453 → 80 [ACK] Seq=38404 Ack=1 Win=262144 Len=1452 [TCP segment of a reasemb...

```

.... ..0 = Fin: Not set
[TCP Flags: .....A....]
Window size value: 1024
[Calculated window size: 262144]
[Window size scaling factor: 256]
Checksum: 0xed51 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0

```

(fig.19 The calculated window size in entire connection)

Time	Source	Destination	Protocol	Length	Info
19	8.534080	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=662 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
20	8.534087	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=2114 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
21	8.534092	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=3566 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
22	8.534095	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=5018 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
23	8.534098	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=6470 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
24	8.534101	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=7922 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
25	8.534103	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=9374 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
26	8.534106	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=10826 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
27	8.534109	192.168.1.103	128.119.245.12	TCP	1506 8617 → 80 [ACK] Seq=12278 Ack=1 Win=66560 Len=1452 [TCP segment of a reasemb...
28	8.535885	128.119.245.12	192.168.1.103	TCP	66 80 → 8616 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1452 SACK_PERM=1 WS=128

(fig.20 No package is retransmitted)

## Analysis(including answer of question):

### Assignment 6.1

1.From the fig.1 we can see there are four fields in the UDP header.

They are: **Source port, Destination Port, Length and Checksum.**

2.The fig.2 shows that each function of the header fields in the UDP header.

The length of each of the UDP header fields is **2 bytes.**

3.From the fig.1, we can see the length field in the length is **433.**

We can verify by counting the byte number of the content showed in the fig.3.

4.From question 2, we learn that the length of length field is 2 bytes. 2 bytes contain  $2 \times 8 = 16$  bits, so that the maximum number of the length is  $2^{16} = 65536$ . But IP field take up 20bytes, UDP header take up 8 bytes. So the answer is  **$65536 - 28 = 65507$  bytes.**

5.The length of source port number is 2bytes. The largest possible source port number is  **$2^{16} = 65536$ .**

6. See the fig.4, we can learn the protocol number is  **$(17)_{10}$  or  $(11)_{16}$ .**

### Assignment 6.2 TCP

3.Q: What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

A: From fig.8, we can know the IP address and TCP port number by you client are **192.168.1.103 and 8617.**

4.Q: 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?

A: (1) The sequence number of the TCP SYN segment is **0(fig.9).**

(2) The **flag segment** identify this segment as a SYN segment.

5.Q: 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?

A: (1) The sequence of SYNACK is **0(fig.11).**

(2) The value of the acknowledgement field is **1(fig.12).**

(3) Server add 1 to the sequence number of SNY segment.



(4) The flag field identify the segment as a SYNACK segment **by set Acknowledgement as value 1(fig.13).**

6. Q: 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.

A: (1) The sequence number showed in the fig.14 is **1.**

(2) The 'POST' is shown in the fig.15.

7. Q: Consider the TCP segment containing the HTTP POST as the first segment in the

TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what

time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3, page 242 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 242 for all subsequent segments.

Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select: Statistics->TCP Stream Graph->Round Trip Time Graph.

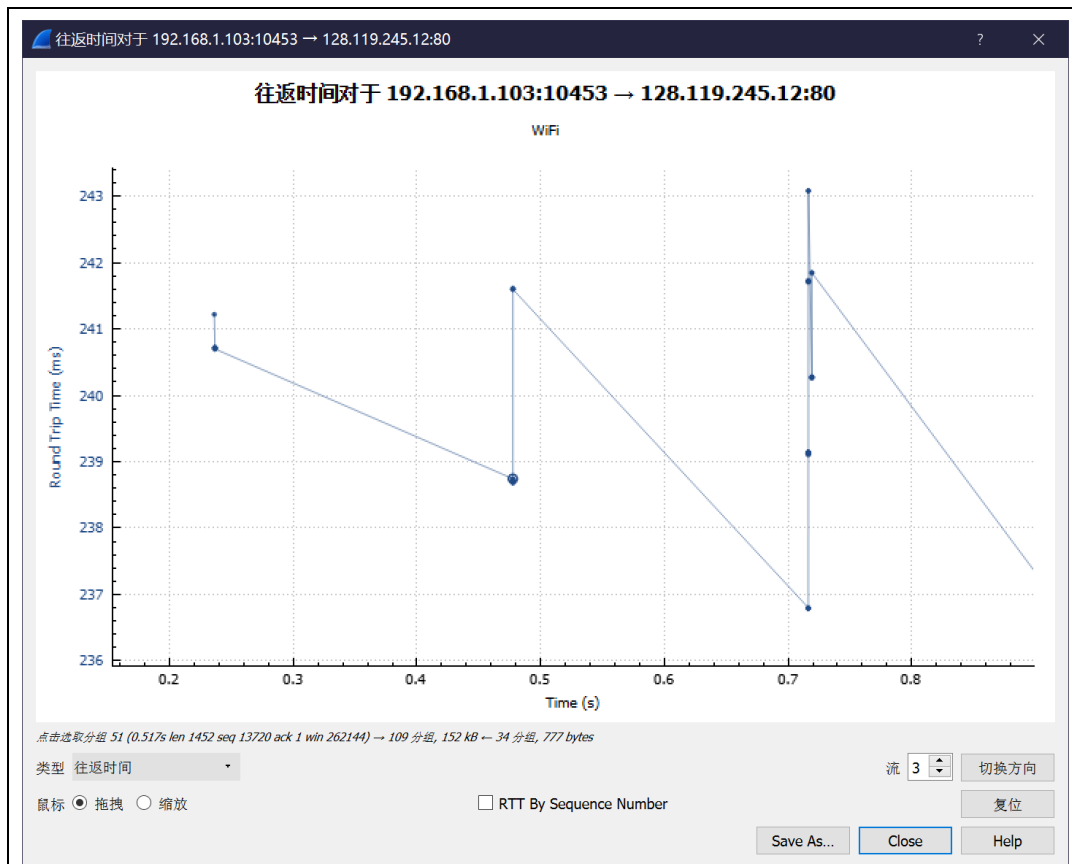
A: (1) **1, 662, 2114, 3566, 5008, 6460 (fig.16)**

(2) **0.516876, 0.516877, 0.516878, 0.516878, 0.516878, 0.516878(fig.17)**

(3) **0.241217, 0.240723, 0.240717, 0.240616, 0.240619, 0.240622(fig.16,17)**

(4) EstimatedRTT value=**0.240947**





8. Q: What is the length of each of the first six TCP segments?

A: The length of them are **66, 54, 54, 54, 54, 66(fig.18).**

9. Q: What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

A: (1) The minimum amount of buffer space is the calculated window size value **262144(fig.19).**

(2) No, the buffer space of the receiver is large enough so that it does not throttle the sender.

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

A: No, there is no segment is retransmitted. We can check if the sequence number of packages sent by client are duplicated(**fig.20**).

**Conclusion and Experience:**

From this assignment, we learn how to use Wireshark to capture TCP package and how to analysis it. Also, we learn the principle of UDP protocol, TCP protocol and how it works.

Tips: