CS305-2022 Fall Lab8 Report

Name：Ji Chenqing

Student ID：11911303

Practice8-1: TCP stream

Firstly, we should know the ipv4 address of **gaia.cs.umass.edu**, so we open the command line and do DNS query:

```
nslookup gaia.cs.umass.edu
```

The query result is shown below:

```
C:\Users\Administrator>nslookup gaia.cs.umass.edu
服务器:  ns1.sustech.edu.cn
Address:  172.18.1.92

非权威应答:
名称:    gaia.cs.umass.edu
Address:  128.119.245.12
```

So, then we open the Wireshark, and start capturing the packets, using the **display filters**:

```
ip.addr == 128.119.245.12 && tcp.stream
```

Then, open the command line, and invoke a HTTP request:

```
curl http://gaia.cs.umass.edu/wiresharklabs/alice.txt
```

And we can get multiple packets:

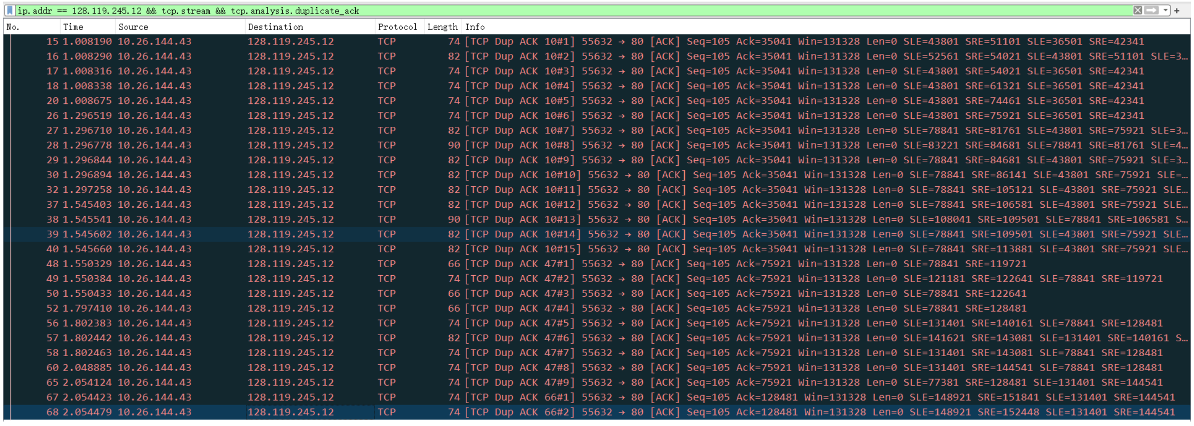
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.26.144.43	128.119.245.12	TCP	66	55632 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.253159	128.119.245.12	10.26.144.43	TCP	66	80 → 55632 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM WS=128
3	0.253274	10.26.144.43	128.119.245.12	TCP	54	55632 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
4	0.253462	10.26.144.43	128.119.245.12	HTTP	158	GET /wiresharklabs/alice.txt HTTP/1.1
5	0.500723	128.119.245.12	10.26.144.43	TCP	56	80 → 55632 [ACK] Seq=1 Ack=105 Win=29312 Len=0
6	0.500723	128.119.245.12	10.26.144.43	TCP	14654	80 → 55632 [ACK] Seq=1 Ack=105 Win=29312 Len=14600 [TCP segment of a reassembled PDU]
7	0.500897	10.26.144.43	128.119.245.12	TCP	54	55632 → 80 [ACK] Seq=105 Ack=14601 Win=131328 Len=0
8	0.755790	128.119.245.12	10.26.144.43	TCP	20494	80 → 55632 [ACK] Seq=14601 Ack=105 Win=29312 Len=20440 [TCP segment of a reassembled PDU]
9	0.755790	128.119.245.12	10.26.144.43	TCP	5894	[TCP Previous segment not captured] 80 → 55632 [ACK] Seq=36501 Ack=105 Win=29312 Len=5840 [TCP segment of a reas
10	0.756012	10.26.144.43	128.119.245.12	TCP	66	55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=36501 SRE=42341
11	1.008123	128.119.245.12	10.26.144.43	TCP	7354	[TCP Previous segment not captured] 80 → 55632 [ACK] Seq=43801 Ack=105 Win=29312 Len=7300 [TCP segment of a reas
12	1.008123	128.119.245.12	10.26.144.43	TCP	1514	[TCP Previous segment not captured] 80 → 55632 [ACK] Seq=52561 Ack=105 Win=29312 Len=1460 [TCP segment of a reas
13	1.008123	128.119.245.12	10.26.144.43	TCP	1514	[TCP Out-Of-Order] 80 → 55632 [ACK] Seq=51101 Ack=105 Win=29312 Len=1460 [TCP segment of a reassembled PDU]
14	1.008123	128.119.245.12	10.26.144.43	TCP	7354	80 → 55632 [ACK] Seq=54021 Ack=105 Win=29312 Len=7300 [TCP segment of a reassembled PDU]
15	1.008190	10.26.144.43	128.119.245.12	TCP	74	[TCP Dup ACK 10#1] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=43801 SRE=51101 SLE=36501 SRE=42341
16	1.008290	10.26.144.43	128.119.245.12	TCP	82	[TCP Dup ACK 10#2] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=52561 SRE=54021 SLE=43801 SRE=51101 S
17	1.008316	10.26.144.43	128.119.245.12	TCP	74	[TCP Dup ACK 10#3] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=43801 SRE=54021 SLE=36501 SRE=42341
18	1.008338	10.26.144.43	128.119.245.12	TCP	74	[TCP Dup ACK 10#4] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=43801 SRE=61321 SLE=36501 SRE=42341
19	1.008617	128.119.245.12	10.26.144.43	TCP	13194	80 → 55632 [ACK] Seq=61321 Ack=105 Win=29312 Len=13140 [TCP segment of a reassembled PDU]
20	1.008675	10.26.144.43	128.119.245.12	TCP	74	[TCP Dup ACK 10#5] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=43801 SRE=74461 SLE=36501 SRE=42341
21	1.296390	128.119.245.12	10.26.144.43	TCP	1514	80 → 55632 [ACK] Seq=74461 Ack=105 Win=29312 Len=1460 [TCP segment of a reassembled PDU]
22	1.296390	128.119.245.12	10.26.144.43	TCP	2974	[TCP Previous segment not captured] 80 → 55632 [ACK] Seq=78841 Ack=105 Win=29312 Len=2920 [TCP segment of a reas
23	1.296390	128.119.245.12	10.26.144.43	TCP	1514	[TCP Previous segment not captured] 80 → 55632 [ACK] Seq=83221 Ack=105 Win=29312 Len=1460 [TCP segment of a reas
24	1.296390	128.119.245.12	10.26.144.43	TCP	1514	[TCP Out-Of-Order] 80 → 55632 [ACK] Seq=81761 Ack=105 Win=29312 Len=1460 [TCP segment of a reassembled PDU]
25	1.296390	128.119.245.12	10.26.144.43	TCP	1514	80 → 55632 [ACK] Seq=84681 Ack=105 Win=29312 Len=1460 [TCP segment of a reassembled PDU]
26	1.296519	10.26.144.43	128.119.245.12	TCP	74	[TCP Dup ACK 10#6] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=43801 SRE=75921 SLE=36501 SRE=42341
27	1.296710	10.26.144.43	128.119.245.12	TCP	82	[TCP Dup ACK 10#7] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=78841 SRE=81761 SLE=43801 SRE=75921 S
28	1.296778	10.26.144.43	128.119.245.12	TCP	90	[TCP Dup ACK 10#8] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=83221 SRE=84681 SLE=78841 SRE=81761 S
29	1.296844	10.26.144.43	128.119.245.12	TCP	82	[TCP Dup ACK 10#9] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=78841 SRE=84681 SLE=43801 SRE=75921 S
30	1.296894	10.26.144.43	128.119.245.12	TCP	82	[TCP Dup ACK 10#10] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=78841 SRE=86141 SLE=43801 SRE=75921
31	1.297128	128.119.245.12	10.26.144.43	TCP	19034	80 → 55632 [ACK] Seq=86141 Ack=105 Win=29312 Len=18980 [TCP segment of a reassembled PDU]
32	1.297258	10.26.144.43	128.119.245.12	TCP	82	[TCP Dup ACK 10#11] 55632 → 80 [ACK] Seq=105 Ack=35041 Win=131328 Len=0 SLE=78841 SRE=105121 SLE=43801 SRE=75921

Q1: Duplicate ACKs

Change filters into this:

```
ip.addr == 128.119.245.12 && tcp.stream && tcp.analysis.duplicate_ack
```

So, we get many Duplicate ACKs packets.



The possible cause is that some packet segments are lost during sending. The receiving end receives the sequence after these packet segments (out-of-order arrival). Therefore, the receiving end sends **Duplicate ACKs packets** to the sending end.

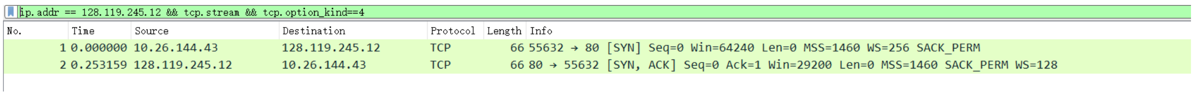
Q2: SACK

1. sack permit option

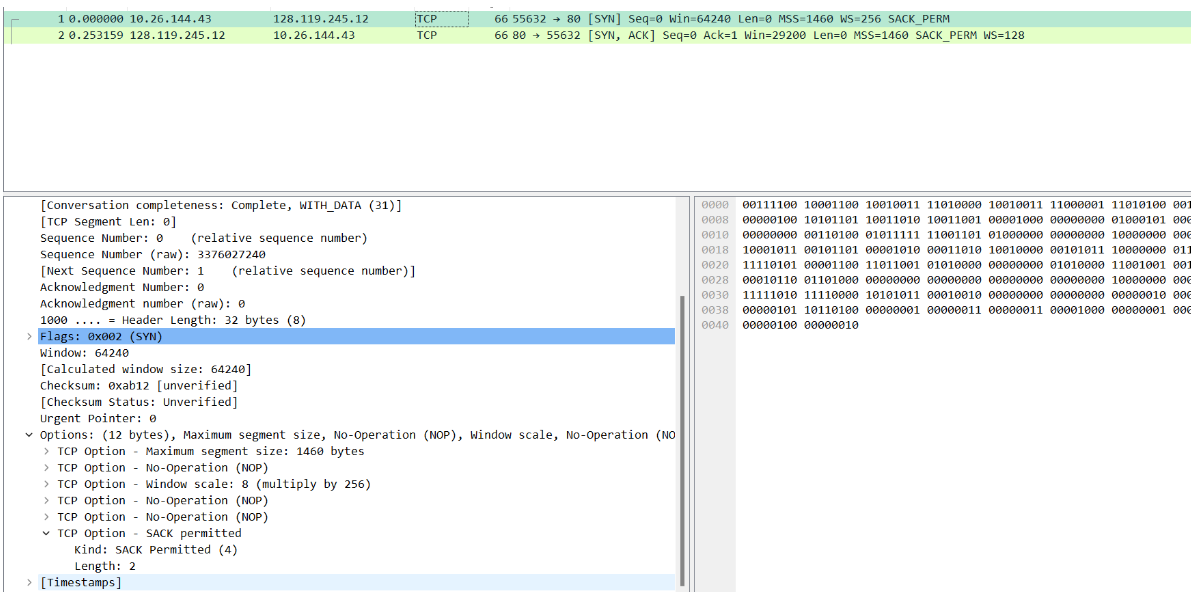
Change filters into this:

```
ip.addr == 128.119.245.12 && tcp.stream && tcp.option_kind==4
```

So we can get two TCP packets:



Expand the packet one we can get sack permit option:

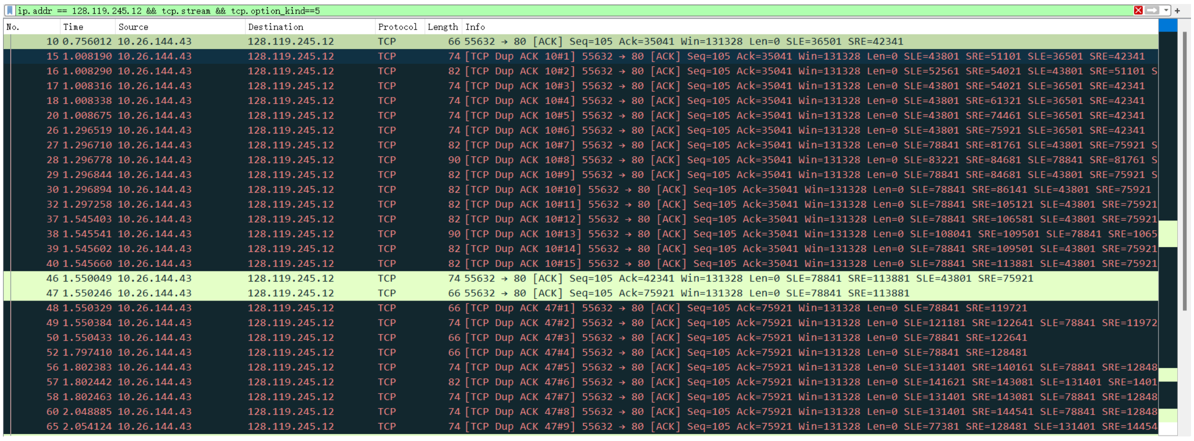


2. sack option

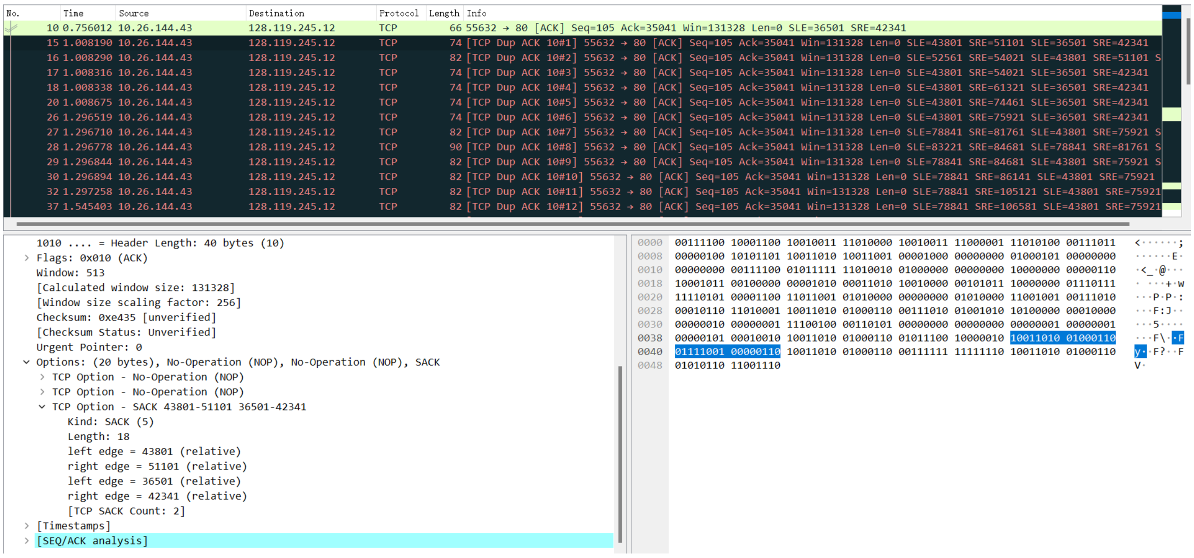
Change filters into this:

```
ip.addr == 128.119.245.12 && tcp.stream && tcp.option_kind==5
```

So we can get many TCP packets:



Expand the packet NO.15 we can get sack option:



So, in this sack option, the segment ranges 43801 - 51101 and 36501 - 42301 is acked.

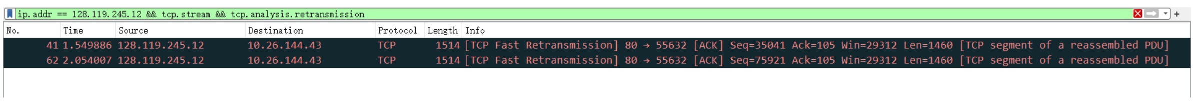
(Note: sack option is the option information contained in the duplicate ack reply packet)

Q3: Retransmission

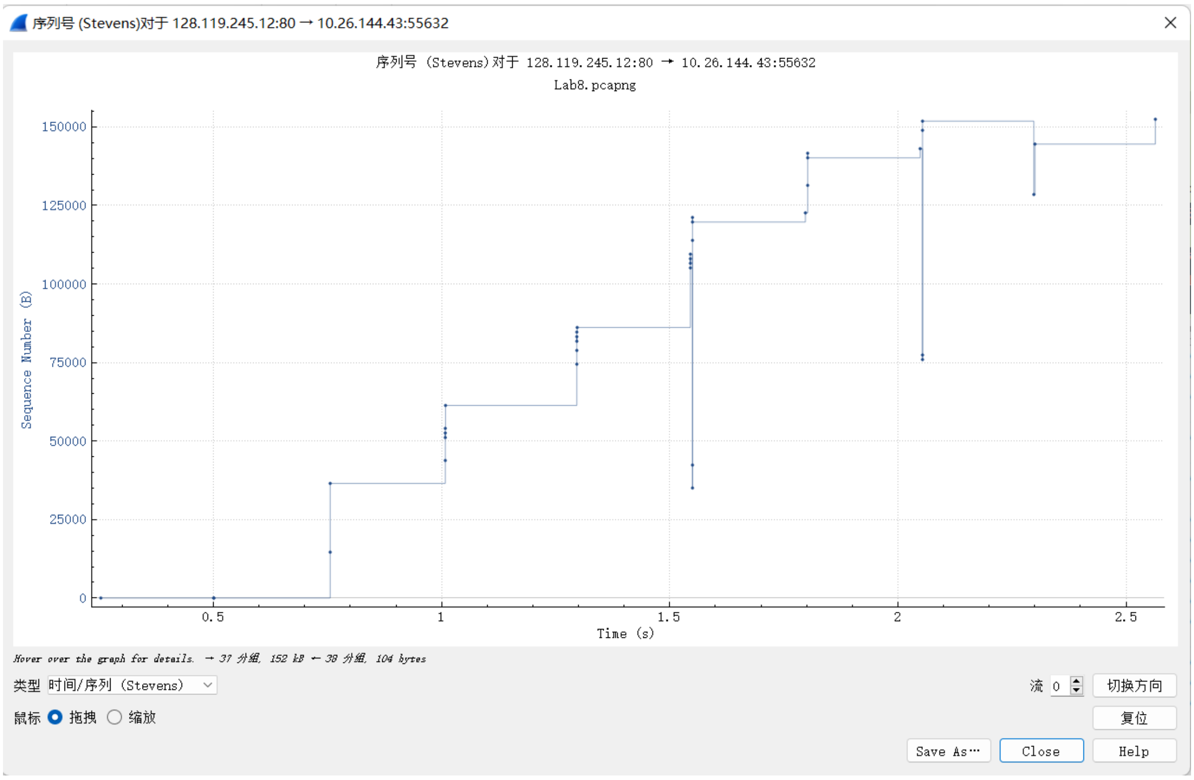
Change filters into this:

```
ip.addr == 128.119.245.12 && tcp.stream && tcp.analysis.retransmission
```

So, we can get two packets which use the fast retransmission



Also, we can consider the sequence number-time graph:



We can see that the sequence number is decreasing at some time, thus there exists retransmission(fast retransmission).

Q4& Q5: TCP Windows size

1. Zero window size

Change filters into this:

```
ip.addr == 128.119.245.12 && tcp.stream && tcp.window_size == 0
```

Then, we cannot see any packets. So, there is no one whose windows size is 0.

If there exists some packet which window sizes is zeros, it means the window size of the receiver (The host send this window size message) is 0. Then, the sender will stop to send the packet, it will send "[TCP Keep-Alive]" to keep the TCP connection, waiting for the changing of receiver's window. Wireshark收到的报文显示的win=***代表该报文发送方的接收窗口的大小

2. Full window size

Change filters into this:

```
ip.addr == 128.119.245.12 && tcp.stream && tcp.analysis.window_full
```

Then, we cannot see any packets. So, there is no one whose windows size is full.

If there exists some packet which window sizes is full, it means the sender (The host send this window size message) will not send any segment at this moment (The size of its usable window turn to be 0, because the number of bytes in transmit (Seq + Len-Ack [the lastest Ack of the other host]) is equal to the receiving window of the other host). At this time, wireshark mark the segment with "[Tcp Window Full]".

Note: TCP implements congestion control and traffic control by dynamically adjusting the size of the sending window.