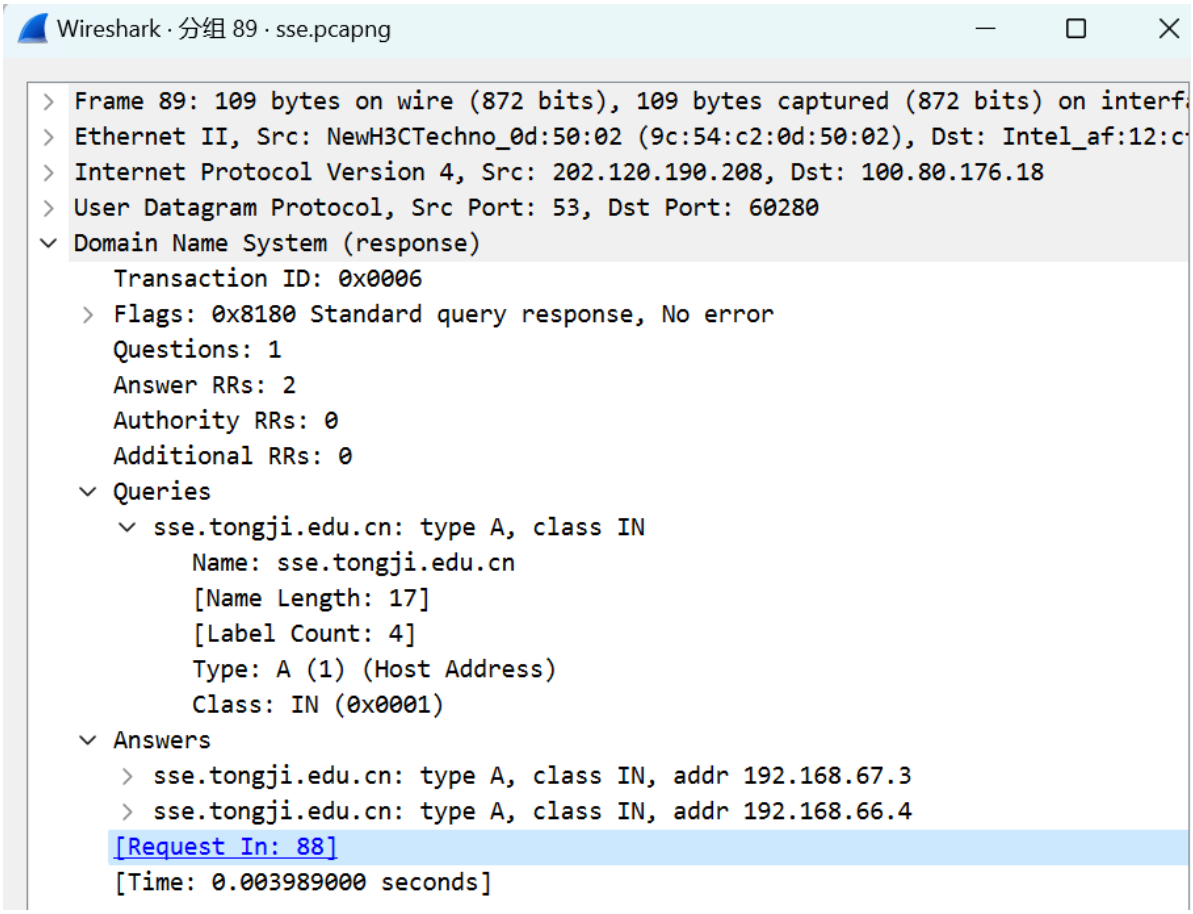
CN_miniProject

Q 1.1

My own IP address: 100.80.176.18

IP address of sse.tongji.edu.cn :192.168.67.3、192.168.66.4

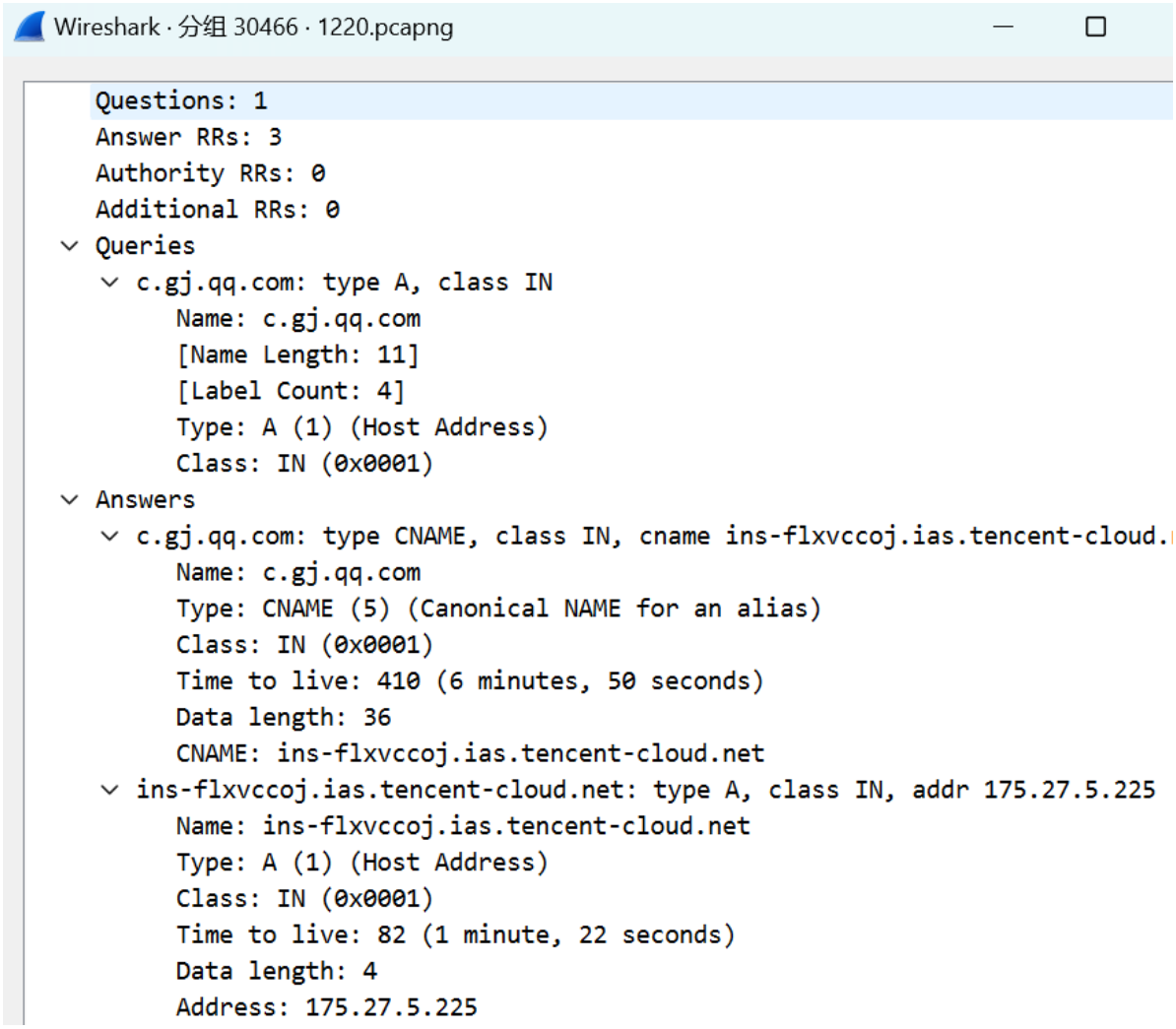


Q 1.2

I saw that in addition to the website I visited before, there are other three-way handshakes for network connection establishment.

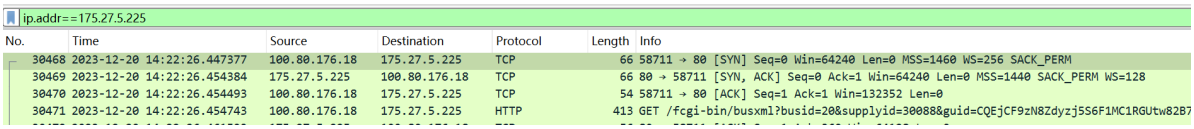
I opened QQ to log in, and then opened the Baidu web page. First, I need to know what the IP addresses of the two pages I open are connected to. So I filtered the data packets related to the domain name through the dns filtering conditions, and saw a lot of data packets with "qq.com" or "baidu.com" information, so I further refined the filtering conditions and used `dns.qry.name contains "qq.com"` filter, and the data packet sent during the domain name resolution process is found.

There are many data packets, and I only selected one of them to find the domain name. I clicked on the last response packet, found the domain name of one of the qq servers in the answer part of the application layer message.In the picture below, you can see that the IP related to qq obtained by domain name resolution is 175.27.5.225.



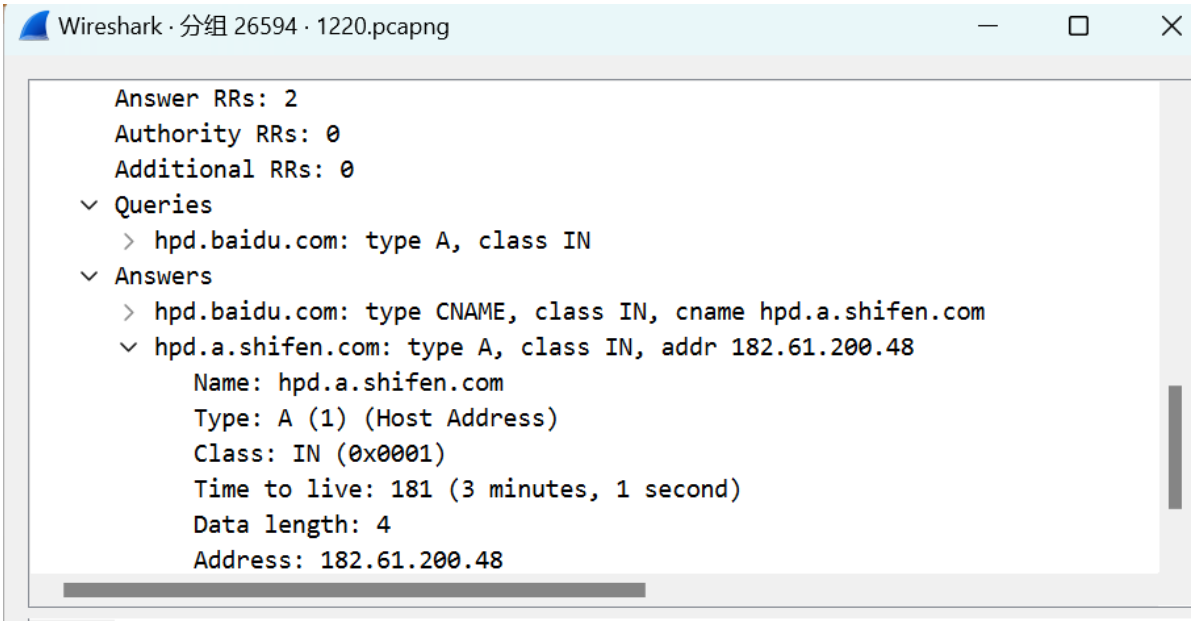
Then I used `ip.addr==175.27.5.225` again to filter the relevant packets. I can see my computer and this IP has a three-way handshake process to establish a connection, and the fourth message uses the HTTP protocol to send content.

connection to IP:175.27.5.225



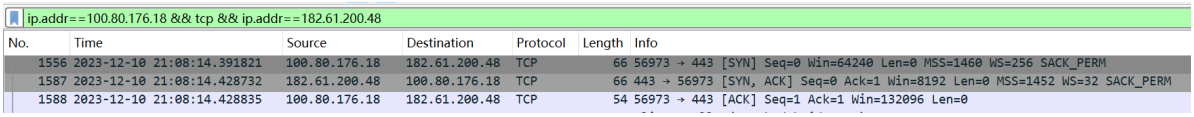
For the network connection search process after opening the Baidu web page, the process is the same as above. First, use dns.qry.name contains "baidu.com" to find the data packet related to domain name resolution, and then find the answer of the response from it, and obtain the relevant IP address, as follows As shown in the figure. Then use the filter statement of ip.addr to find the process of establishing a connection with this IP.

For the network connection search process after opening the Baidu web page, the process is the same as above. First, use `dns.qry.name contains "baidu.com"` to find the data packet related to domain name resolution, and then find the answer of the response from it, and obtain the relevant IP address, as shown in the figure.



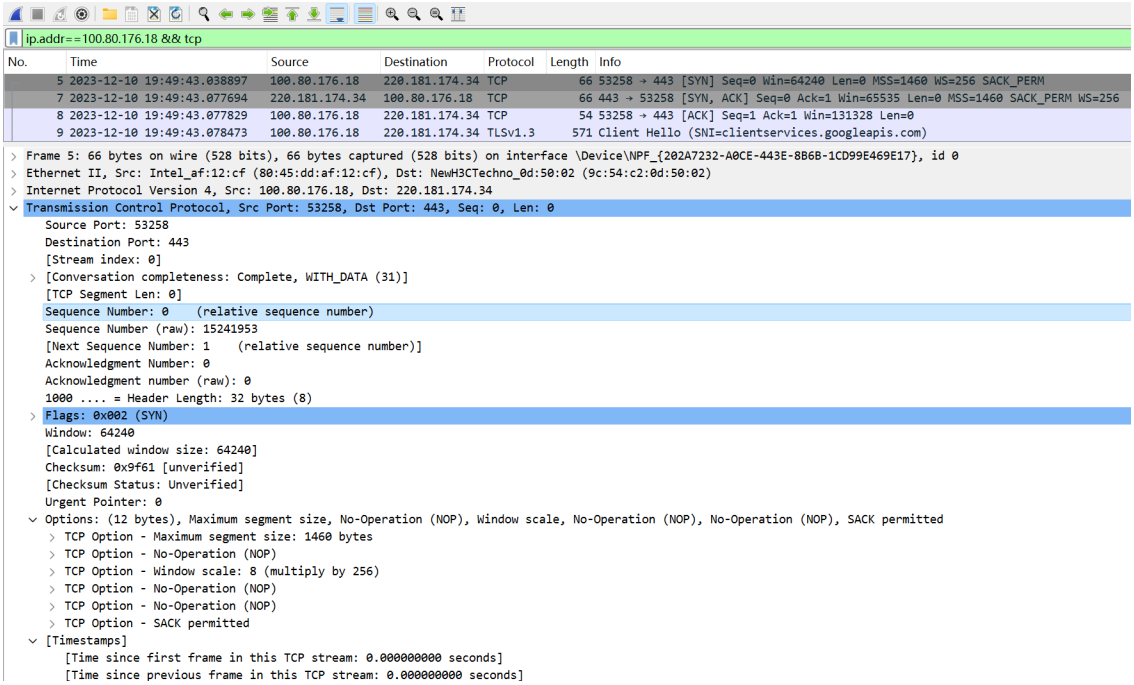
Then use the filter statement of `ip.addr==182.61.200.48` to find the process of establishing a connection with this IP.

connection to IP:182.61.200.48

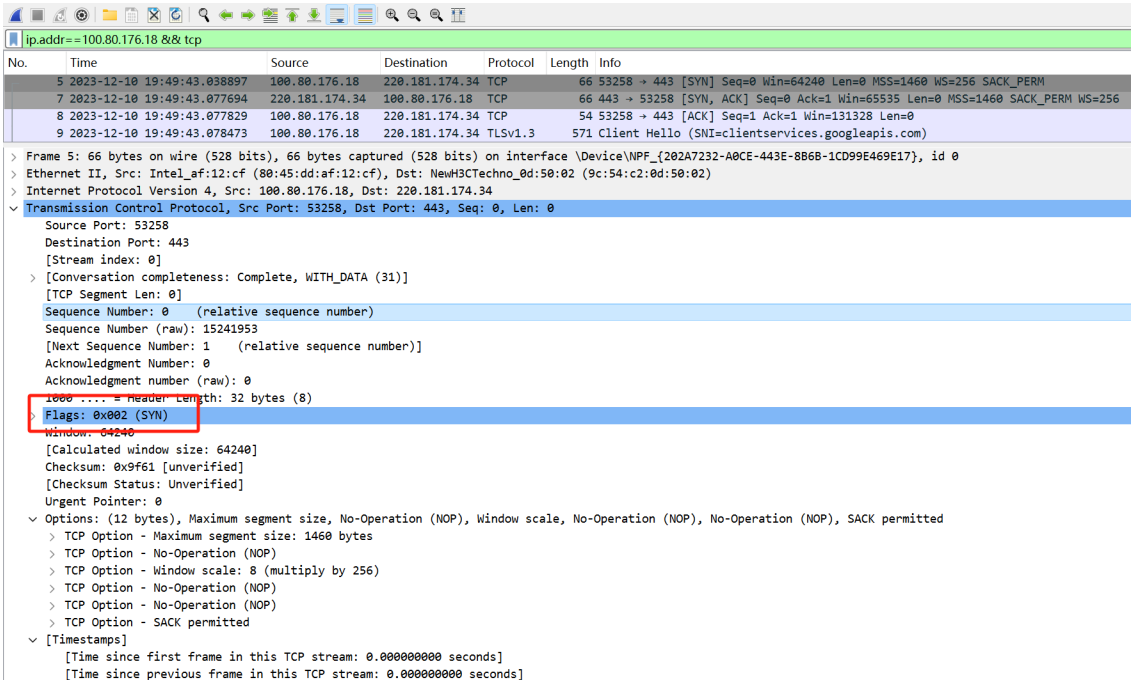


Q 2.1

1. The sequence number is 15241953,the relative sequence number is 0

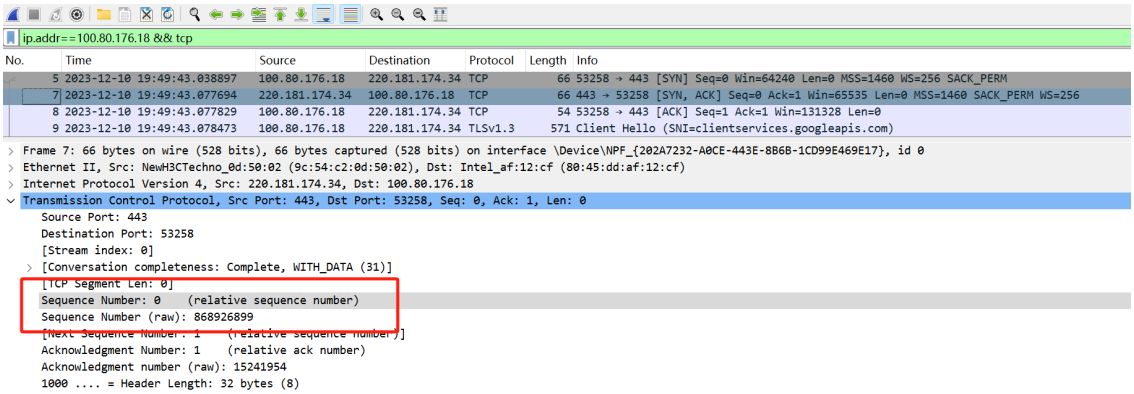


2. The Flags:0x002(SYN) identifies the segment as a SYN segment

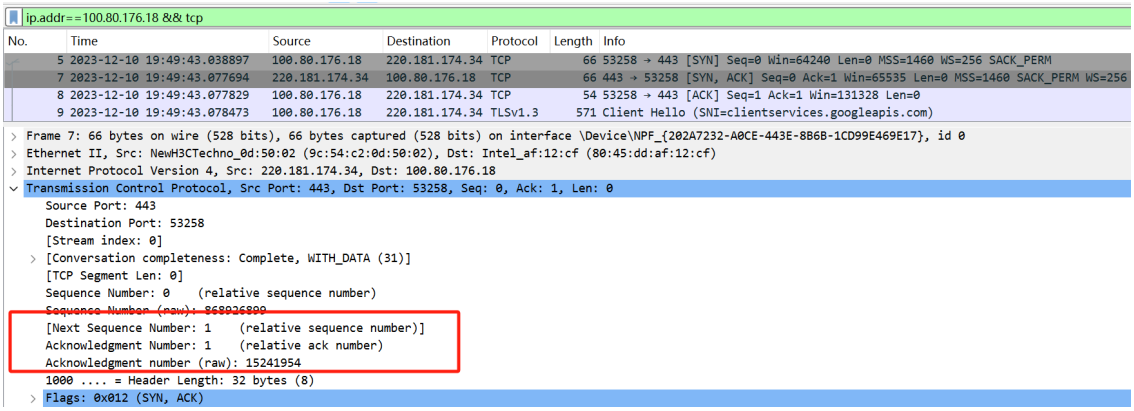


Q 2.2

1. The sequence number is 868926899,the relative sequence number is 0

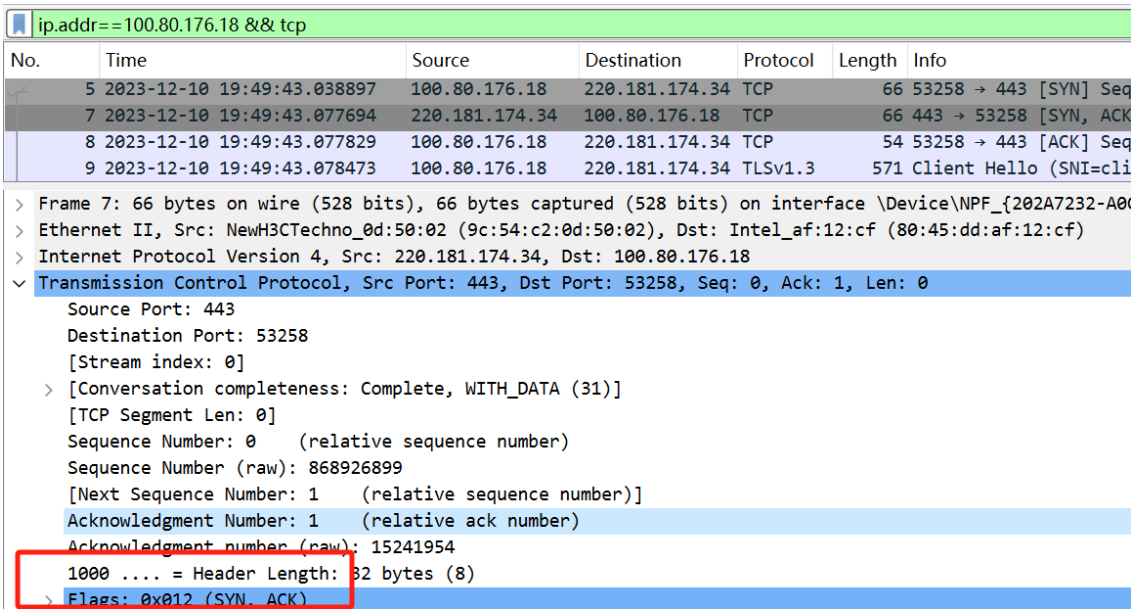


2. The value of relative ack number is 1,the raw acknowledgment number is 15241954



3. The value in the acknowledgment field is the sequence number in the received SYN packet + 1

4. Flags:0x012(SYN,ACK) identifies the segment as a SYNACK segment



Q 2.3

No,if a two-way handshake is used, at most only the starting sequence number of the connection initiator can be confirmed, and the sequence number selected by the other party cannot be confirmed.

For example: Suppose the Client sent a request to the Server, but it stayed in the network for a while and was not delivered in time. At this time, the Client will send a request to the Server again. The Server receives the request, sends a confirmation packet, completes the connection, and starts data transmission. After the data transfer is completed, disconnect.

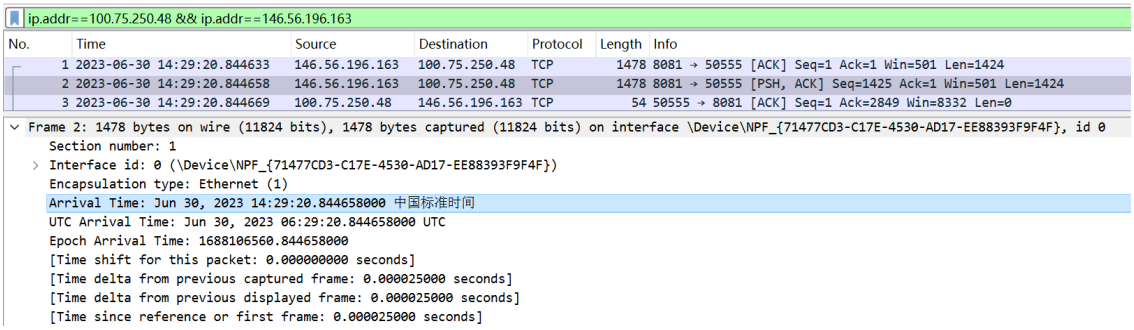
But after the connection is disconnected, the Client's first request finally finds the server again. At this time, the Server will send a confirmation packet and the connection is established. However, the Client has completed the data service and will not do anything, while the Server is still waiting for further steps. Data interaction - Server-side resources are wasted.

The connection request issued by the client cannot be confirmed by using a handshake, and it is also not feasible.

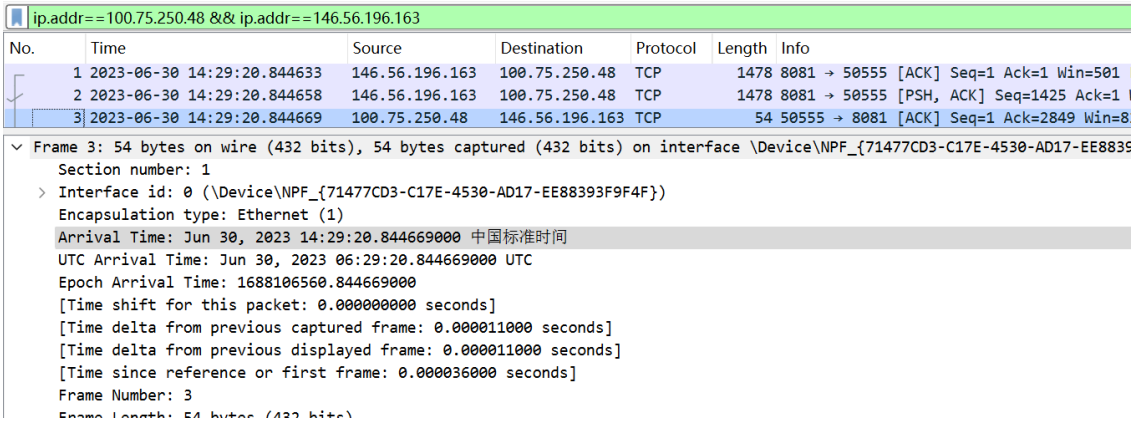
Q 2.4

I chose the No.2 segment.

1. 2023-06-30 14:29:20.844658



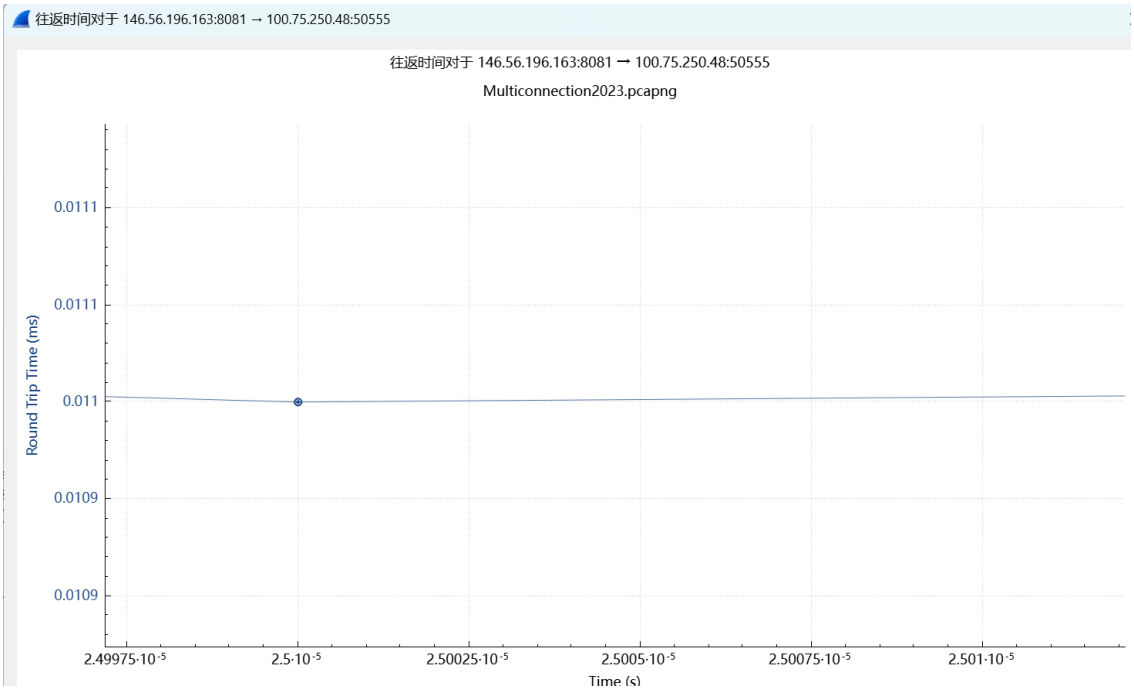
2. 2023-06-30 14:29:20.844669



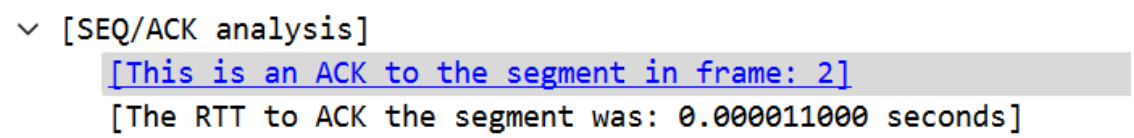
3. 1478
4. 0.000011000 seconds.

   The sending time of the selected data packet is 2023-06-30 14:29:20.844658, and the arrival time of the acknowledged data packet is 2023-06-30 14:29:20.844669. From this, the RTT can be calculated to be 0.000011000 seconds.

   In the TCP Stream Graph,we can see the RTT of the selected packet is 0.011ms



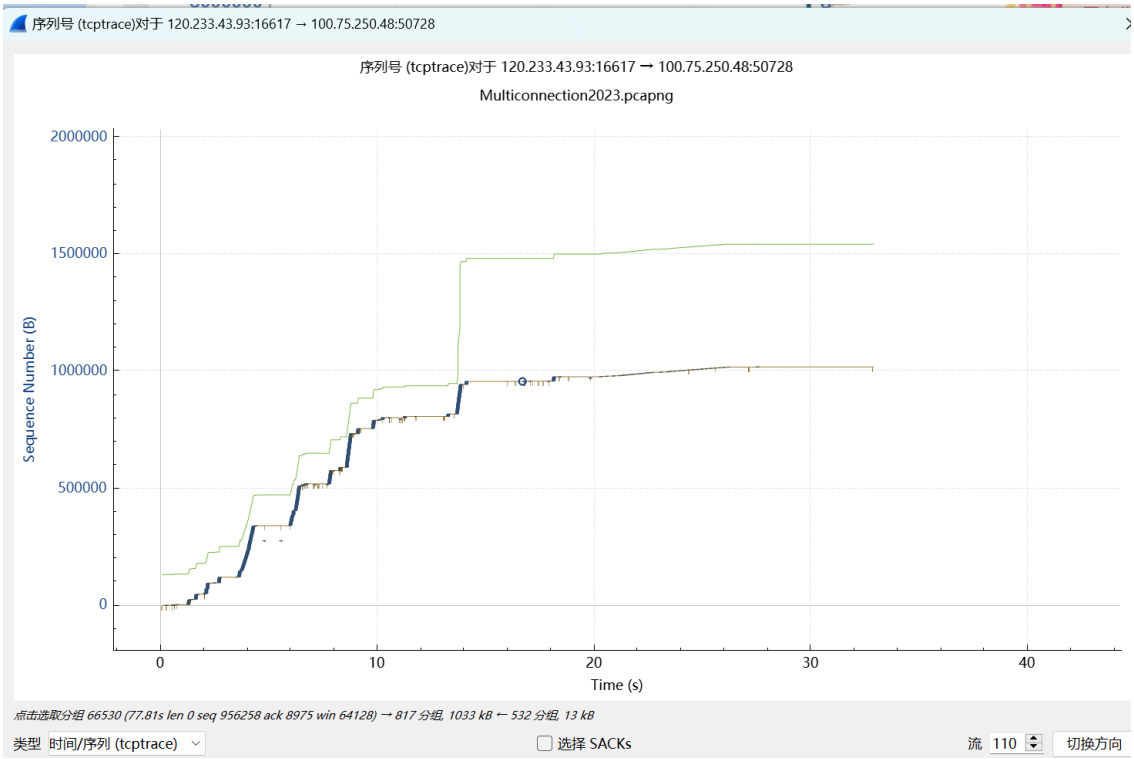
   Another method: Click to view the detailed information of the ACK datagram, which will display its RTT.



Q 2.5

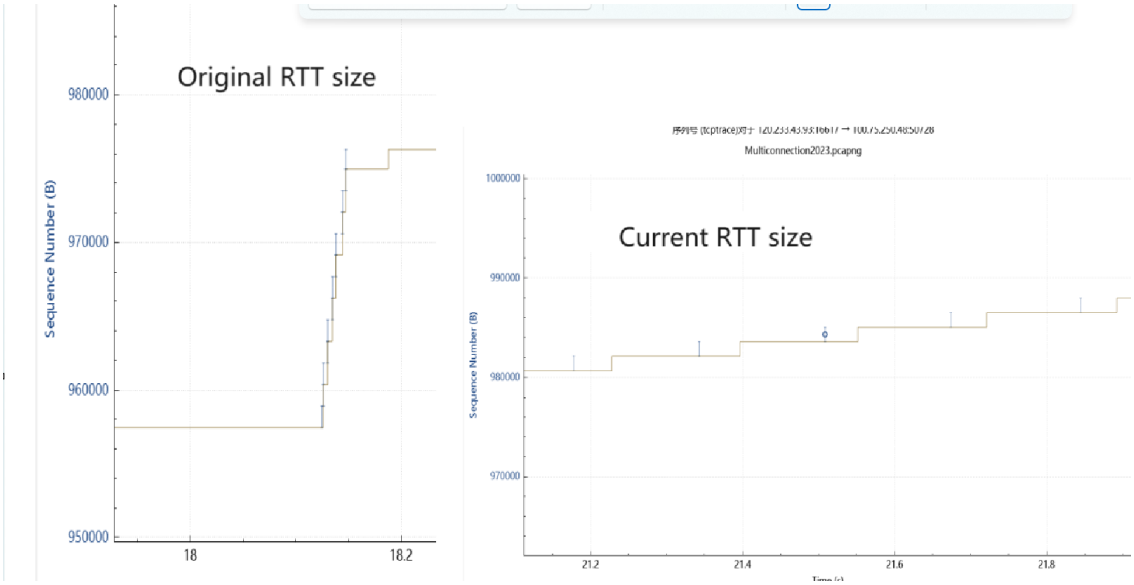
1. The concept is the full-duplex nature of TCP, which means that data can flow independently in both directions (from client to server and vice versa. That's why in full-duplex communication, tcptrace follows the direction of transmission. Changes as they change. For example, when you open a web page in a browser, the computer (client) sends a request to the server (this is one direction), and the server responds and sends the content of the web page back to your computer (this is another direction). one direction).

2. If the receiver's buffer space is insufficient, the sender's data transmission will be limited by adjusting the TCP window size, allowing the sender to send slower, and wait until the receiver has more buffer space before continuing to send. It can be seen from this picture that as soon as the yellow line rises (the receiving end confirms), the blue line rises (the sending end starts sending) until the green line is filled. After the blue line rises for a period of time, it is close to the green line (the receiving end starts sending). (Insufficient buffer space on the other side) will stagnate and rise for a period of time. During this period, no data is sent, indicating that the sending rate of the sender is limited. When the green line is farther away from the blue line, the blue line rises again, which indicates that the receiver has enough buffer space, so the sender starts sending data quickly again.



3. We can see three lines: a blue line, a yellow line and a green line. The blue line represents the sending sequence number, while the green line represents the upper limit of the sending window. The yellow line represents the sequence number confirmed by the receiving end. At about 20 seconds, the blue line suddenly dropped and remained stable. After zooming in on the picture, you can see in the figure below that the parallel distance between the blue line and the yellow line has also become larger, indicating that the RTT time has become larger, which may mean that Data packet loss occurs, causing the TCP sending end to reduce the sending window, thereby reducing the data sending speed and avoiding network congestion. Therefore, from this tcptrace graph, we can deduce that network congestion may have occurred at about 20 seconds.

   I think this congestion is not serious, because there are not a large number of red lines in the picture, only a few scattered ones, which means that there are just some repeated ACKs and the RTT time is getting longer, which means that the receiver's confirmation information can still arrive. On the sender side, there is just some network congestion. It is not serious enough that the receiver's information cannot be sent back.



4. When a TCP connection needs to retransmit data, it follows certain steps:

    1. Retransmission Trigger: If an acknowledgment (ACK) for a segment is not received within the retransmission timeout (RTO), the sender retransmits the segment. This is often triggered by the expiration of the time-out timer or the receipt of three duplicate ACKs at the sender.
    2. Sequence Number: The sequence number of the retransmitted segment remains the same as the original segment. This allows the receiver to correctly order the received segments.
    3. Window Size Adjustment: TCP uses congestion control mechanisms to adjust the window size. When packet loss is detected, TCP assumes it's due to network congestion and reduces the congestion window size. This slows down the rate of data transmission to alleviate the congestion.

    Transmission failures can be caused by various factors:

    1. Network Congestion: This can lead to packet loss as more packets are sent than the network can handle.
    2. Hardware Issues: Problems with network hardware, such as routers or cables, can cause packets to be lost or corrupted.
    3. Software Bugs: Errors in network software can lead to packet loss.

    To prevent transmission failures, several strategies can be employed:

    1. Congestion Control Algorithms: Implement advanced congestion control algorithms to better handle network congestion and prevent packet loss.
    2. Hardware and Software Maintenance: Regularly check and maintain network hardware and update network software to prevent failures.
    3. Proper Network Design: Design the network to avoid congestion, ensure sufficient capacity, and provide redundancy to handle failures

Q 2.6

Because TCP is byte-stream oriented, you only need to count the sequence number of the first segment sent and the last segment acknowledged within a certain period of time. The difference between the two is the number of bytes sent. The throughput is obtained by dividing this number of bytes by this period of time.

Q 2.7

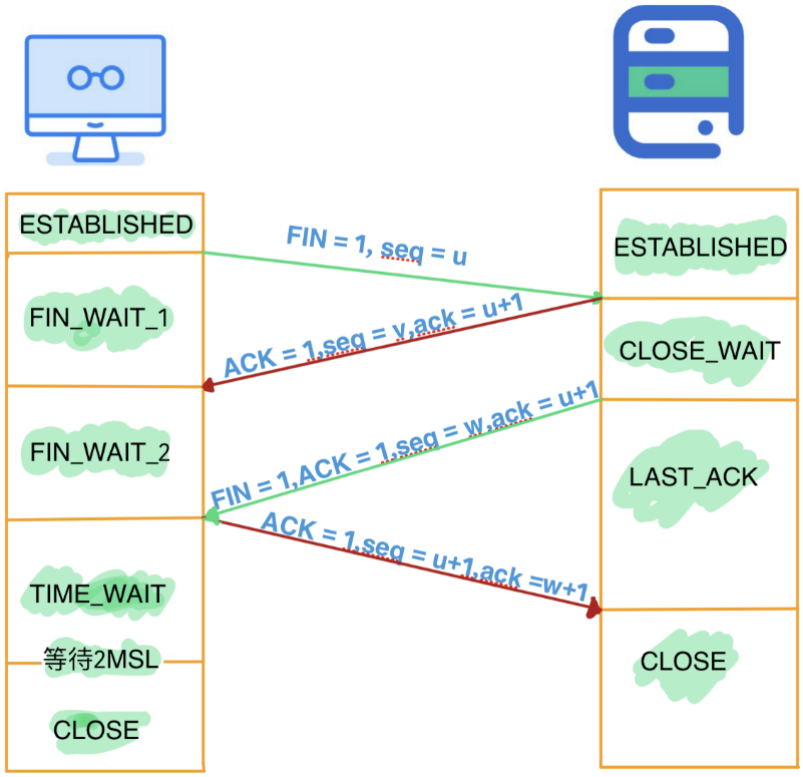
One set of connections between the two IPs 100.75.250.48 and 120.233.43.93 specified in Q2.5: 100.75.250.48:50384 and 120.233.43.93:16617 was terminated at 2023-06-30 14:29:52.786767.

ip.addr==100.75.250.48 && ip.addr==120.233.43.93 &&tcp

No.	Time	Source	Destination	Protocol	Length	Info
7016	2023-06-30 14:29:22.816259	100.75.250.48	120.233.43.93	TCP	55	50384 → 16617 [ACK] Seq=1 Ack=1 Win=513 Len=1
7213	2023-06-30 14:29:22.868537	120.233.43.93	100.75.250.48	TCP	66	16617 → 50384 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
53475	2023-06-30 14:29:52.531091	120.233.43.93	100.75.250.48	TCP	56	16617 → 50384 [FIN, ACK] Seq=1 Ack=2 Win=501 Len=0
53476	2023-06-30 14:29:52.531132	100.75.250.48	120.233.43.93	TCP	54	50384 → 16617 [ACK] Seq=2 Ack=2 Win=513 Len=0
53477	2023-06-30 14:29:52.531210	100.75.250.48	120.233.43.93	TCP	62	50384 → 16617 [PSH, ACK] Seq=2 Ack=2 Win=513 Len=8
53478	2023-06-30 14:29:52.531247	100.75.250.48	120.233.43.93	TCP	54	50384 → 16617 [FIN, ACK] Seq=10 Ack=2 Win=513 Len=0
53489	2023-06-30 14:29:52.786767	120.233.43.93	100.75.250.48	TCP	56	16617 → 50384 [RST] Seq=2 Win=0 Len=0

Usually, if you want to find when two connections are disconnected, you can look for TCP packets with the FIN flag bit, and then look for whether there are four wave-disconnection processes in the nearby time period, so that you can find the time when the connection is terminated.

In the four-way handshake:



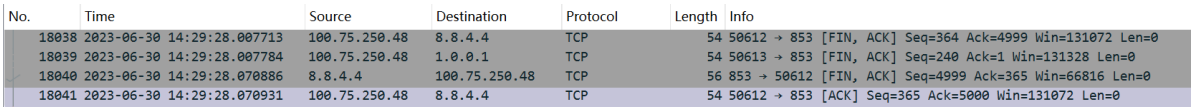
1. The first device begins the termination process by sending a FIN (Finish) packet to the second device.The sequence number `u` of the datagram depends on the context.
2. The second device responds with an ACK (Acknowledgment) packet, acknowledging the receipt of the FIN packet.
3. Subsequently, the second device sends its own FIN packet to the first device, indicating that it also wants to end the connection.
4. The first device completes the handshake by sending an ACK packet in response to the second device's FIN packet.

This process ensures an orderly closure of the connection where both devices agree to terminate the communication. Each step includes the necessary confirmation before proceeding to the next, allowing both sides to close their respective end of the connection gracefully.

Q 2.8

TCP connection can be terminated by waving three times. As long as the end that passively receives the disconnection request message combines the acknowledgment packet for the incoming FIN datagram and its own data packet with the FIN flag and sends them together, it can achieve TCP disconnection by waving three times.

For example, the example in the figure below achieves disconnection by sending data packets three times.



No.	Time	Source	Destination	Protocol	Length	Info
18038	2023-06-30 14:29:28.007713	100.75.250.48	8.8.4.4	TCP	54	50612 → 853 [FIN, ACK] Seq=364 Ack=4999 Win=131072 Len=0
18039	2023-06-30 14:29:28.007784	100.75.250.48	1.0.0.1	TCP	54	50613 → 853 [FIN, ACK] Seq=240 Ack=1 Win=131328 Len=0
18040	2023-06-30 14:29:28.070886	8.8.4.4	100.75.250.48	TCP	56	853 → 50612 [FIN, ACK] Seq=4999 Ack=365 Win=66816 Len=0
18041	2023-06-30 14:29:28.070931	100.75.250.48	8.8.4.4	TCP	54	50612 → 853 [ACK] Seq=365 Ack=5000 Win=131072 Len=0

But I think this is essentially still four waves, but two of them are sent in the same piece of data.

Feedback

I think this assignment is relatively new. This way of analyzing real cases gave me a concrete understanding of the theoretical knowledge I learned. And there are many complex situations in reality that are not as ideal as what the textbooks say.

It took me about 8 hours to completely complete this task.

I really like the progressive difficulty of this assignment. The first few questions allowed me to quickly get started with the software, which is good.

One problem is that the description of the question occasionally has some ambiguities, such as the first question in 2.4 and the third question in 2.4. But I think this may be a problem with English expression. Maybe it can be described more clearly in Chinese.