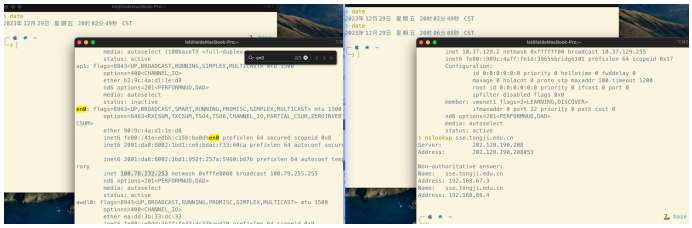
1 Q1

1.1 Q1.1

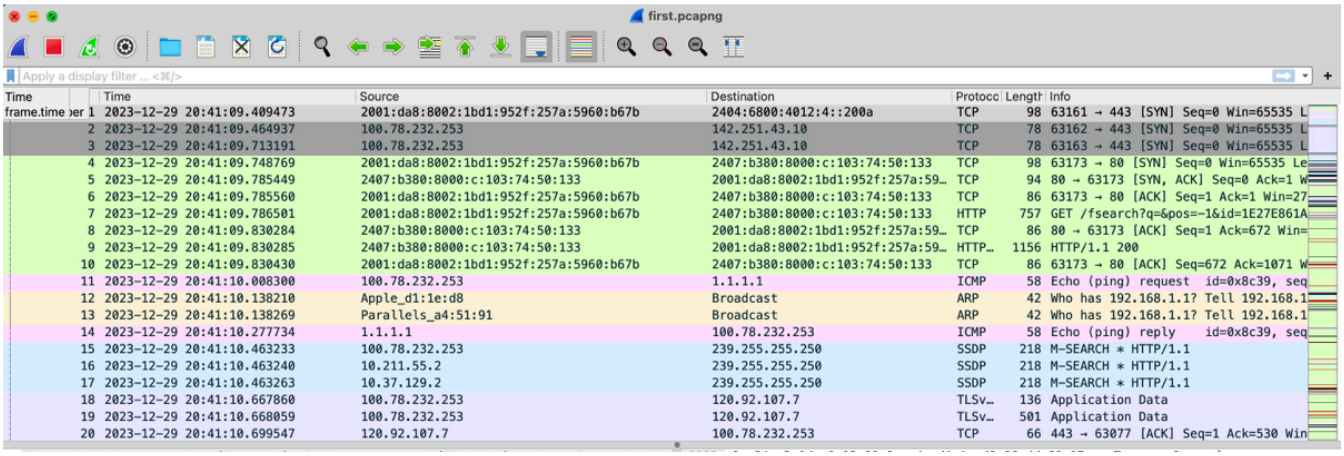
The IP address used by my own computer is `100.78.232.253`. For IPv6 address, my own computer is `2001:da8:8002:1bd1:952f:257a:5960:b67b`.

The IP address of sse.tongji.edu.cn is `192.168.67.3` or `192.168.66.4`.



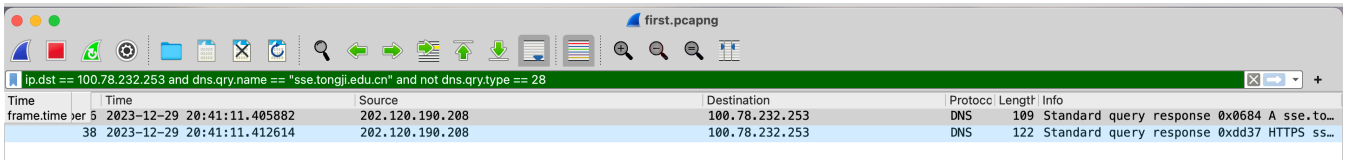
1.2 Q1.2

After launching additional applications like QQ and WeChat on my computer that can access the Internet, I started capturing network traffic again while keeping the previously accessed website open. After a few seconds, I stopped the capture.



From the packet trace, several connections can be observed. To focus on two specific connections, I applied filters in Wireshark as follows:

```
ip.dst == 100.78.232.253 and dns.qry.name ==
"sse.tongji.edu.cn" and not dns.qry.type == 28
```



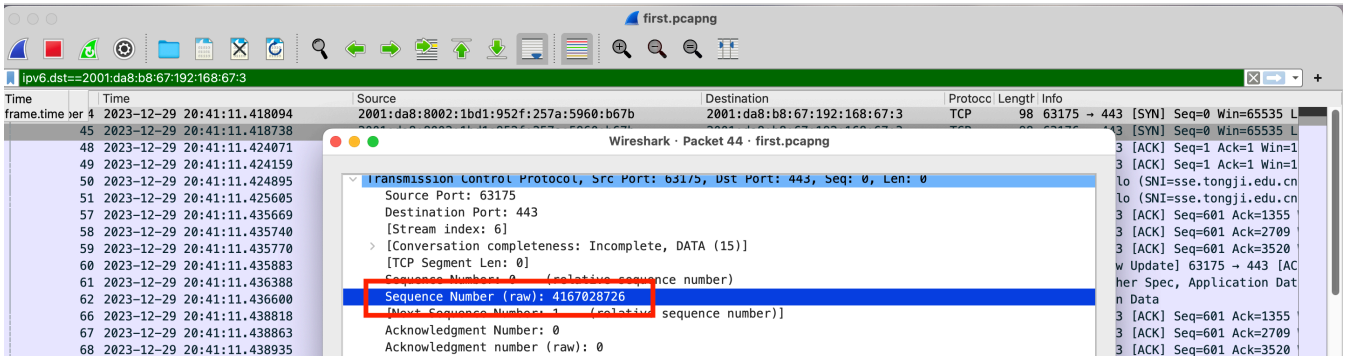
The Wireshark filter is set to: `ip.dst == 100.78.232.253` to filter packets destined for IP 100.78.232.253, `dns.qry.name == "sse.tongji.edu.cn"` for DNS queries to "sse.tongji.edu.cn", and `not dns.qry.type == 28` to exclude AAAA (IPv6) DNS queries.

2 Q2

2.1 Q2.1

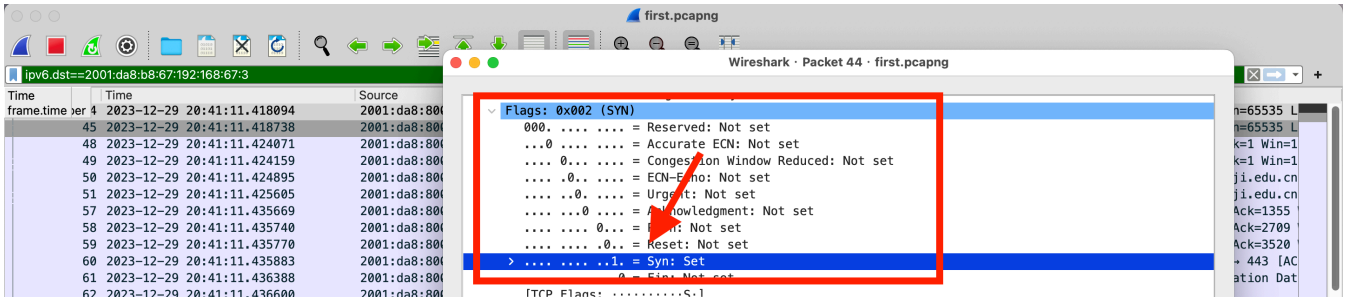
a. Use the following command to filte in Wireshark,:

```
ipv6.dst==2001:da8:b8:67:192:168:67:3
```



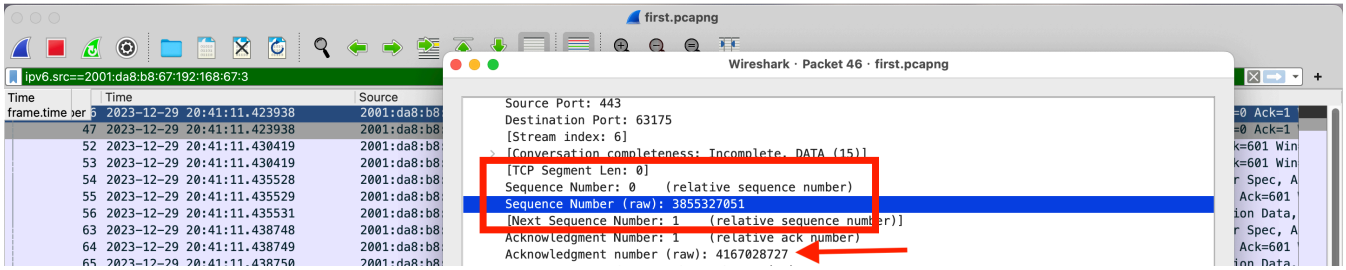
The sequence number of the TCP SYN segment that is used to initiate the TCP connection between the source and destination is **4167028726**.

b. A SYN segment is identified when the "flags" field in the TCP header is set to **0x002**, corresponding to the **SYN** flag bit being set to **1**.

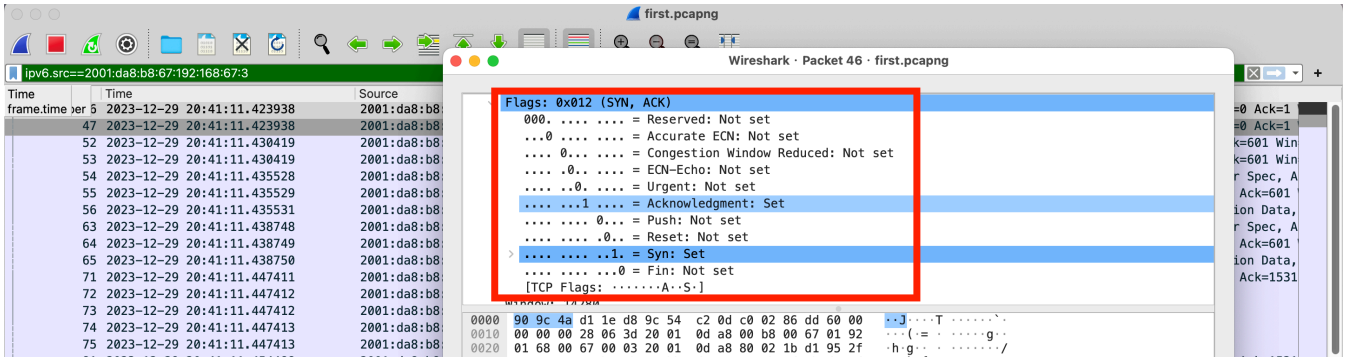


2.2 Q2.2

a. The sequence number of the SYNACK segment sent by destination to the source in reply to the SYN is **3855327051**.



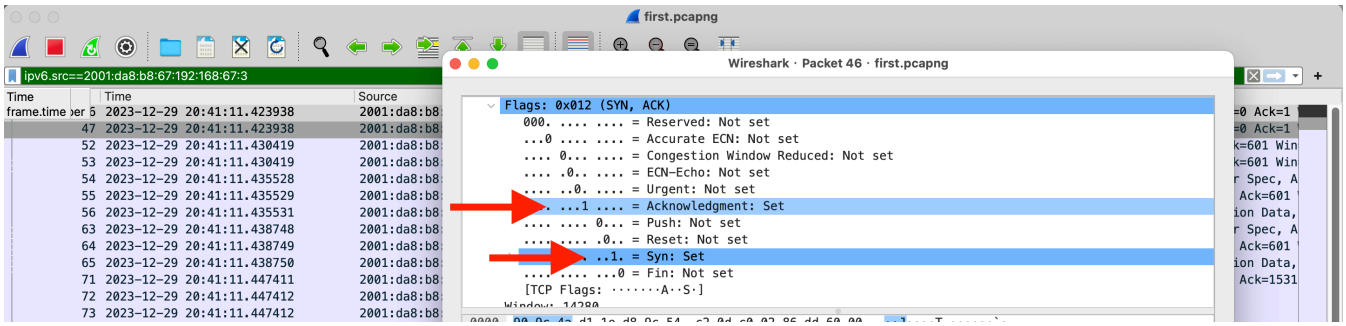
b. The value of the Acknowledgement filed in the SYNACK segment is below:



c. The destination determines the value in the acknowledgment field by adding one to the sequence number received in the TCP segment from the source.

$$Acknowledgement = sequence\ number + 1$$

d. A segment is identified as a SYNACK segment based on specific bits set in the TCP header's flags field. In this case, the flags value is set to **0x012**, which corresponds to both the **Acknowledgment** and **SYN** flags being set to **1**.



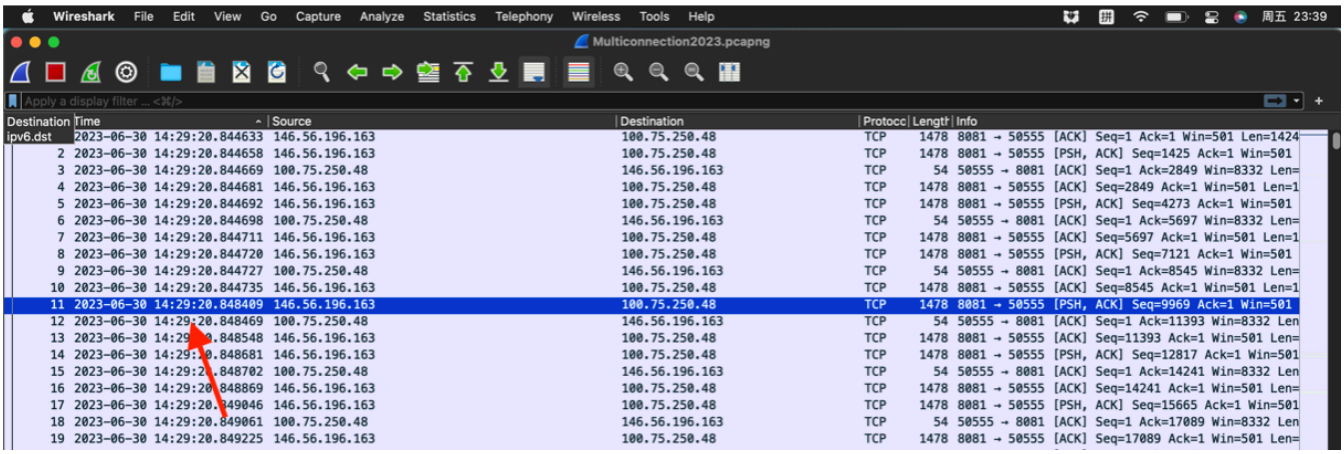
2.3 Q2.3

The TCP "three-way handshake" is an essential and non-reducible process for establishing a connection. It involves SYN, SYN-ACK, and ACK. This process is crucial for synchronizing sequence numbers for reliable data transmission, negotiating connection parameters, preventing issues from delayed packets of old connections, and ensuring security and reliability. Any reduction in these steps could compromise these key aspects.

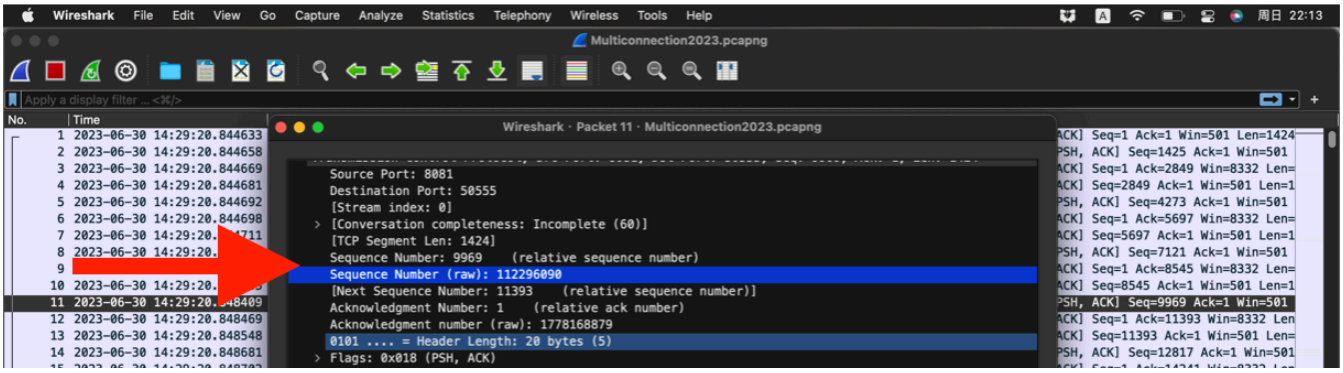
2.4 Q2.4

I pick segment No.11 to analysis.

a. Sent time: `2023-06-30 14:29:20.848409`



b. For segment No.11， the Sequence Number is 9969.

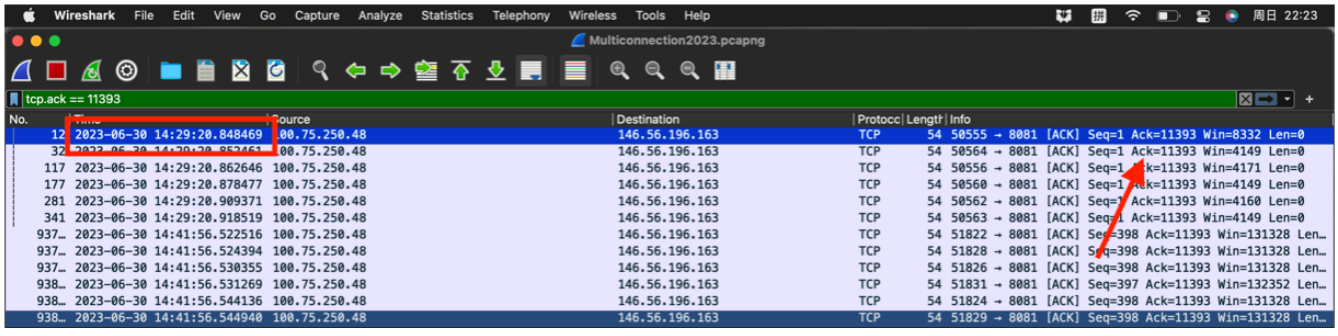


$$Ack = Sequence\ Number + Length\ of\ TCP\ Segment\ Data$$

Hence, Ack = 9969 + 1424 = 11393

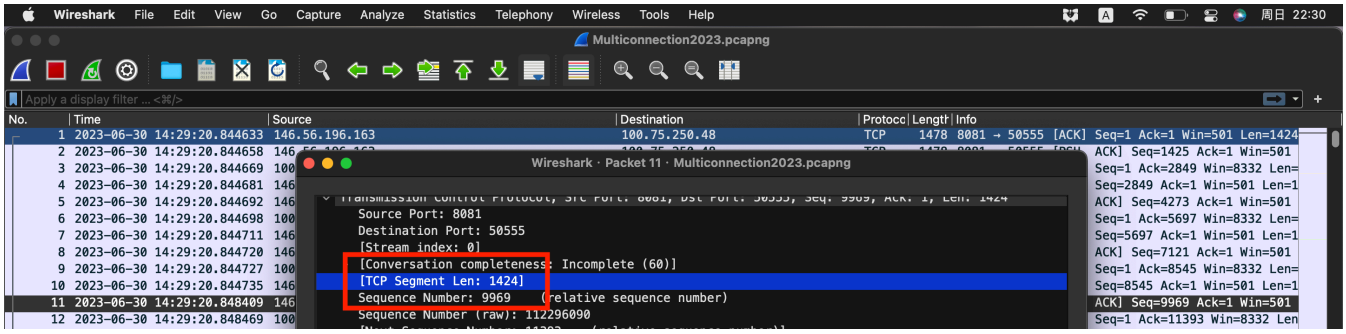
To filter this in Wireshark, use the following command:

```
tcp.ack == 11393
```



The ACK for this segment was received at `2023-06-30 14:29:20.848469`.

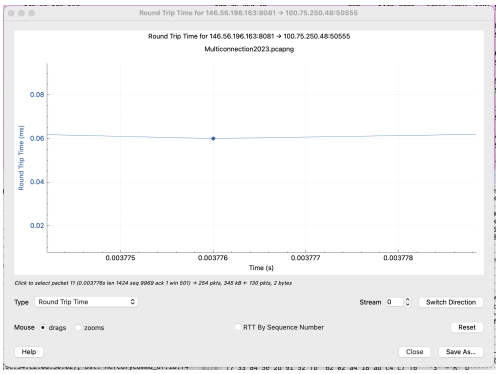
c. The length of the segment is **1424**.



d. RTT (Round-Trip Time) is the time difference between sending a TCP segment and receiving its ACK.

RTT = 2023-06-30 14:29:20.848469 - 2023-06-30 14:29:20.848409 = 0.000060s = 0.06ms

Using Wireshark, go to `Statistics → TCP Stream Graph → Round Trip Time Graph` and select segment 11, we can see the result is 0.06ms.



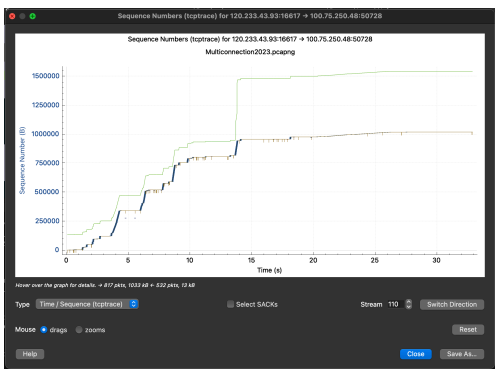
So the RTT value for this segment is 0.06ms.

2.5 Q2.5

a. First, apply the following filter in Wireshark

```
ip.addr==100.75.250.48 && ip.addr==120.233.43.93
```

Then, navigate to `Statistics -> TCP Stream Graphs -> Time Sequence (tcptrace)` to generate a tcptrace graph. Based on TA's suggestion, set the Stream to 110. The resulting graph provides valuable insights:



- **Green Line (Send Window Limit)**: This line represents the sender's perception of the receiver's window limit. When the blue line approaches the green line, it suggests that the sender has reached what it believes to be the limit of the receiver's window, and it must wait for a new ACK to send more data.
- **Blue Line (Send Sequence Number)**: This line grows over time, indicating that packets are being sent. A halt in the growth of the blue line signifies a temporary pause in data transmission by the sender.
- **Yellow Line (Acknowledgment Sequence Number)**: The growth of this line indicates that the receiver has received and acknowledged the packets. If the growth of the yellow line pauses, it may mean that the sender is waiting for more acknowledgments to continue sending data.

In Wireshark, clicking on `Switch Direction` allows us to switch between the two directions of the TCP connection, highlighting the full-duplex communication nature of TCP/IP.

The full-duplex communication: the channel allows simultaneous data transmission in both directions. In the TCP/IP model, this means the client can send data to the server while the server simultaneously sends data to the client, with each direction independently managed and controlled.

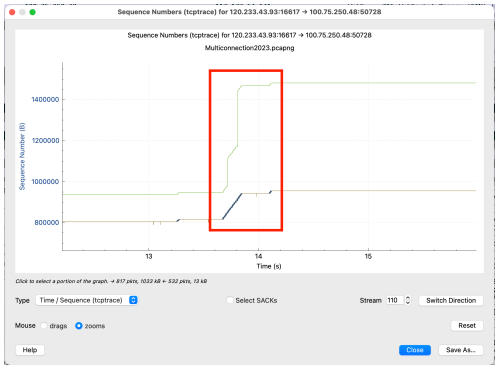
Example: Full-duplex communication is common in daily life, with telephone conversations being a classic example. When two people talk over the phone, full-duplex communication allows them to speak and listen at the same time without taking turns. The key here is that both parties can send and receive information simultaneously.

---

b. The lack of receiver buffer space throttles the sender. This is the flow control mechanism of TCP, which ensures that The sender will not send data that exceeds the receiver's buffer capacity.

Within the tcptrace graph, a diminishing gap between the blue line (sender's sequence numbers) and the yellow line (receiver's acknowledgments) indicates timely ACKs from the receiver, suggesting swift data handling without noticeable delays. A shrinking or stagnant gap between the green line (representative of the receiver's window limit) and the blue line could signify a full receiver buffer, necessitating the sender to halt transmissions until the receiver processes the incoming data and sends a new ACK to update its window size, thereby granting the sender additional space for data transmission.

For a nuanced analysis, I zoomed into a specific part of the graph.



Observe the horizontal stretches within the red box in the graph. These could signify that the sender has paused sending data due to not receiving timely acknowledgments, likely due to receiver buffer restrictions.

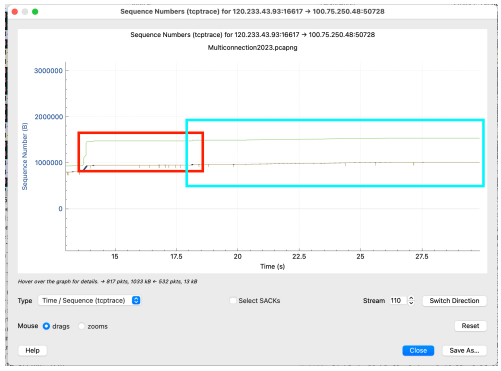
---

c. By analyzing the tcptrace graph, we can evaluate the "smooth" of a TCP connection.

In the tcptrace graph, the horizontal segments of the blue line indicate times when the sender has stopped sending new data, likely due to awaiting acknowledgments from the receiver, thus temporarily halting data transmission.

If these horizontal segments occur frequently and last for extended periods, they may indicate network congestion. The TCP protocol, upon detecting congestion, would reduce its sending rate, causing a stagnation in the sequence numbers.

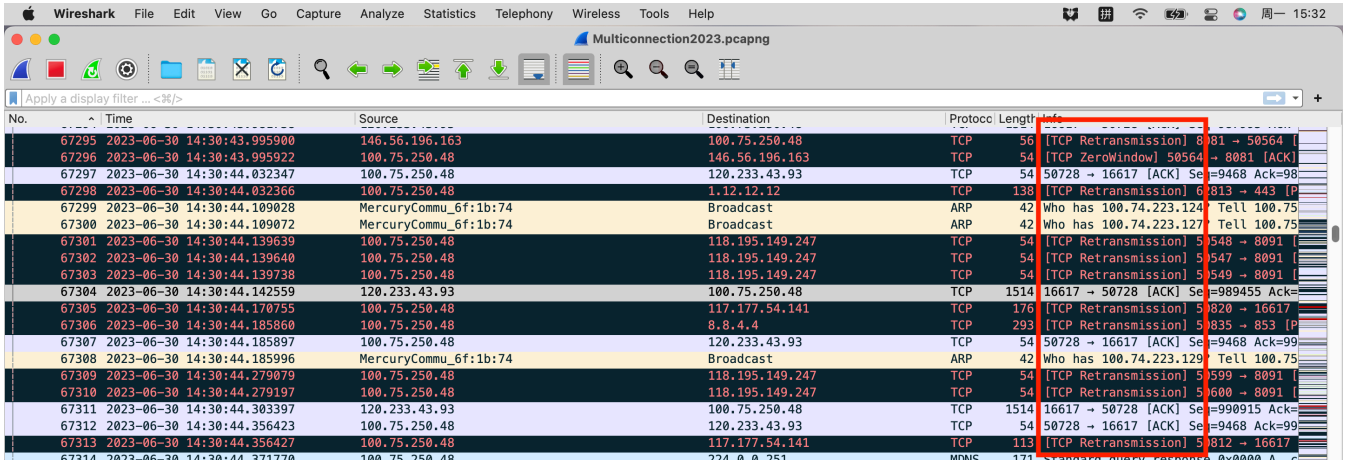
Brief upward movements between these horizontal segments are indicative of retransmissions by the sender, usually in response to delayed ACKs or perceived packet losses.



The graph exhibits multiple horizontal stretches, which could signal potential network congestion, punctuated by very brief increases in sequence numbers, suggesting retransmissions.

These patterns suggest that the TCP connection may have experienced a certain degree of network congestion during the observed period.

The packet list for the corresponding timeframe supports the same conclusion.



d. When a TCP connection requires retransmission, typical actions include packet loss detection, retransmitting lost segments, implementing congestion control measures, adjusting the retransmission timeout (RTO), and temporarily reducing the sending rate.

Common causes of transmission failure are network congestion, unreliable network links, improper timeout configuration, hardware issues, and high latency or jitter.

To mitigate these issues, it's advised to implement Quality of Service (QoS), enhance network reliability, optimize timeout settings, maintain hardware, and manage latency and jitter through effective routing and traffic shaping.

2.6 Q2.6

To calculate the throughput, follow these basic steps:

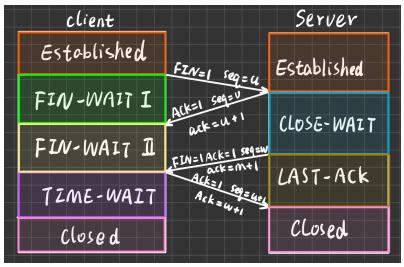
1. **Determine Data Volume and Time Frame**: Start by identifying the total amount of data transferred over the TCP connection (in bytes) and the time taken for this transfer.
2. **Collect Data**: Capture TCP packets with Wireshark, noting the amount of data transferred and the timestamps of each packet.
3. **Calculate Data Volume**: Analyze the captured packets to sum up the payload size (excluding the TCP header) of all TCP packets within the target time frame.
4. **Measure Time Interval**: Identify the start and end times of the period you're analyzing.
5. **Compute Throughput**: Divide the total amount of data from step 3 by the total time from step 4. $Throughput = \frac{TotalTransmissionBytes}{TotalTime}$

Additionally, Wireshark offers a direct way to visualize throughput with its built-in graph feature. You can access it through `Statistics -> TCP Stream Graph -> Throughput`, which generates a graph illustrating the rate of data transfer over time.

2.7 Q2.7

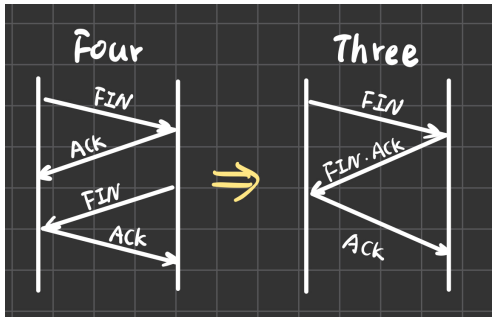
Four-way Handshake Process(Assuming the client initiates the closure):

1. **First Handshake (FIN from Client)**: The client sends a FIN (Finish) flag to indicate no more data is sent but can still receive.
2. **Second Handshake (ACK from Server)**: The server acknowledges with an ACK flag, indicating receipt of the client's termination request.
3. **Third Handshake (FIN from Server)**: After sending all data, the server sends a FIN flag, indicating it has no more data.
4. **Fourth Handshake (ACK from Client)**: The client sends an ACK flag in response to the server's FIN and enters TIME_WAIT state to ensure its ACK is received.



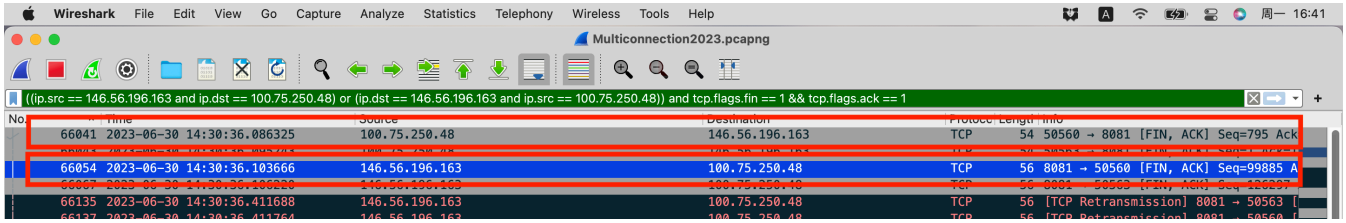
2.8 Q2.8

In certain scenarios, it is indeed possible for the TCP four-way handshake process to appear as a three-way handshake.



, Use the following filter in Wireshark to identify such occurrences:

```
((ip.src == 146.56.196.163 and ip.dst == 100.75.250.48) or
(ip.dst == 146.56.196.163 and ip.src == 100.75.250.48))
```



In this case, the second and third steps of the handshake are merged into one, making it appear as though a three-way handshake has occurred.

Such a three-way handshake scenario may arise when the passive closing party (e.g., the server in the illustration) has "no data to send" and "the TCP delayed acknowledgment mechanism is enabled" during the handshake process. The delay in sending ACKs, triggered after receiving the initial FIN, occurs because the conditions for immediate acknowledgment are not met. During this delay, if the application also decides to close the connection and has no more data to send, it triggers the sending of a FIN, which then gets combined with the previous ACK and sent together.

3 Feedback

Using Wireshark to capture and analyze network packets has not only deepened my understanding of how the TCP/IP protocol works but also enhanced my skills in troubleshooting network issues. Through hands-on experience, I was able to observe the transmission process of data packets and understand how TCP maintains reliable data transmission under various network conditions.

The experiment was detailed and complex, presenting many new challenges in terms of concepts and operations, which required me to do extensive research and learning. I realized that there is a significant gap between theoretical knowledge and practical application.

The difficulty of the experiments, especially parts 2.4 and 2.5, was quite high. My unfamiliarity with Wireshark led me to spend a lot of time searching for information online. In total, I spent over 16 hours on this project, with the majority of that time dedicated to familiarizing myself with the various functions of Wireshark and searching for relevant tutorials. During this process, I found that learning to use filters effectively and utilizing Wireshark to generate related charts were particularly challenging.

Based on my experience, I would suggest allocating some time in the classroom for demonstrating and explaining the basic use of Wireshark. This would not only help beginners to grasp the fundamental operations of this powerful tool more quickly but also improve the efficiency and depth of understanding of the experiments. Useful link: 1 2 3 4 5