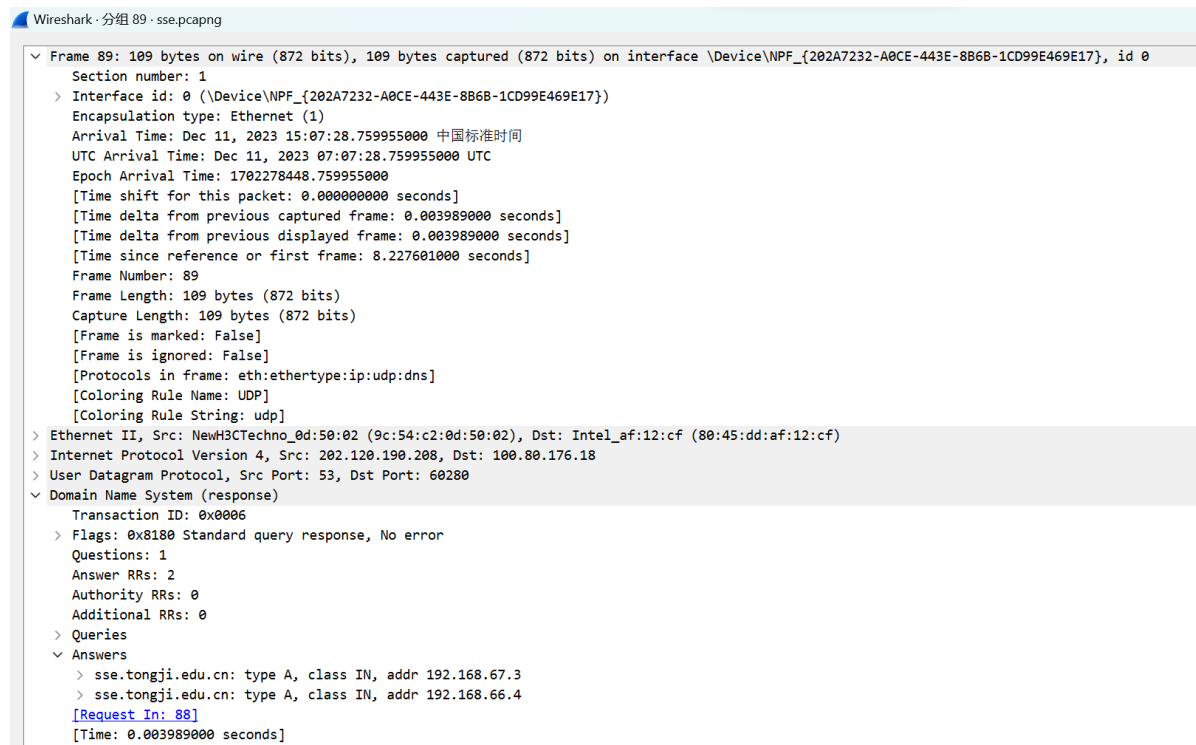


CN_miniProject

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



Wireshark · 分组 89 · sse.pcapng

```

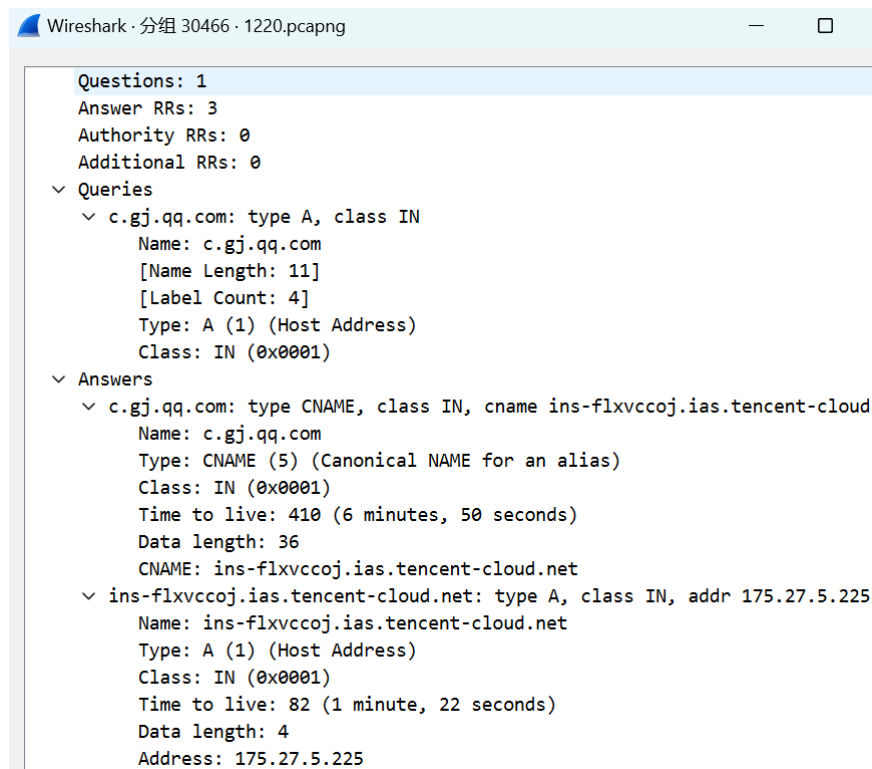
  ▾ Frame 89: 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface \Device\NPF_{202A7232-A0CE-443E-8B6B-1CD99E469E17}, id 0
    Section number: 1
    ▸ Interface id: 0 (\Device\NPF_{202A7232-A0CE-443E-8B6B-1CD99E469E17})
      Encapsulation type: Ethernet (1)
      Arrival Time: Dec 11, 2023 15:07:28.759955000 中国标准时间
      UTC Arrival Time: Dec 11, 2023 07:07:28.759955000 UTC
      Epoch Arrival Time: 1702278448.759955000
      [Time shift for this packet: 0.000000000 seconds]
      [Time delta from previous captured frame: 0.003989000 seconds]
      [Time delta from previous displayed frame: 0.003989000 seconds]
      [Time since reference or first frame: 8.227601000 seconds]
      Frame Number: 89
      Frame Length: 109 bytes (872 bits)
      Capture Length: 109 bytes (872 bits)
      [Frame is marked: False]
      [Frame is ignored: False]
      [Protocols in frame: eth:ethertype:ip:udp:dns]
      [Coloring Rule Name: UDP]
      [Coloring Rule String: udp]
    ▸ Ethernet II, Src: NewH3CTechno_0d:50:02 (9c:54:c2:0d:50:02), Dst: Intel_af:12:cf (80:45:dd:af:12:cf)
    ▸ Internet Protocol Version 4, Src: 202.120.190.208, Dst: 100.80.176.18
    ▸ User Datagram Protocol, Src Port: 53, Dst Port: 60280
    ▾ Domain Name System (response)
      Transaction ID: 0x0006
      ▸ Flags: 0x8180 Standard query response, No error
      Questions: 1
      Answer RRs: 2
      Authority RRs: 0
      Additional RRs: 0
      ▸ Queries
      ▾ Answers
        ▸ sse.tongji.edu.cn: type A, class IN, addr 192.168.67.3
        ▸ sse.tongji.edu.cn: type A, class IN, addr 192.168.66.4
      [Request In: 88]
      [Time: 0.003989000 seconds]
```

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.

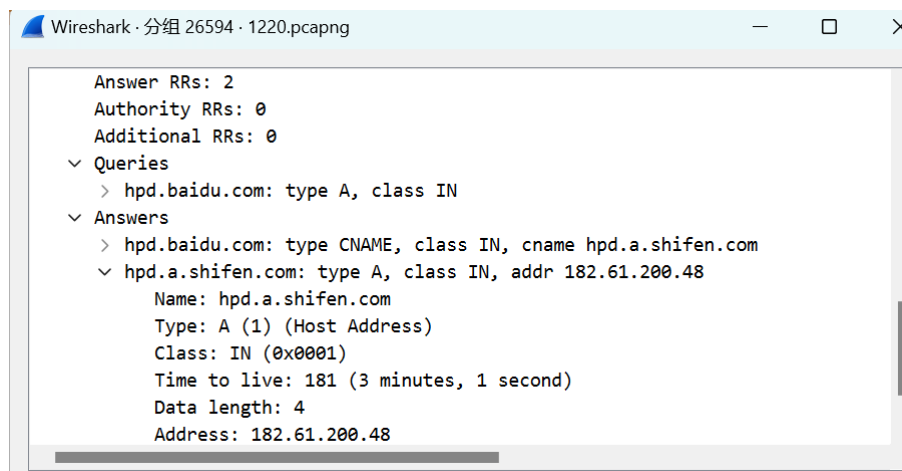


Then I used `ip.addr==175.27.5.225` again to filter the relevant packets. I can see my computer and this IP has a three-way handshake process to establish a connection, and the fourth message uses the HTTP protocol to send content.

connection to IP:175.27.5.225

No.	Time	Source	Destination	Protocol	Length	Info
30466	2023-12-20 14:22:26.447377	100.80.176.18	175.27.5.225	TCP	66	58711 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
30469	2023-12-20 14:22:26.454384	175.27.5.225	100.80.176.18	TCP	66	80 → 58711 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1440 SACK_PERM WS=128
30470	2023-12-20 14:22:26.454493	100.80.176.18	175.27.5.225	TCP	54	58711 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
30471	2023-12-20 14:22:26.454743	100.80.176.18	175.27.5.225	HTTP	413	GET /fcgi-bin/busxml?busid=20&supplyid=30088&guid=CQEjCF9zN8Zdyzj5S6F1MC1RGUtw82B7

To search for a network connection after opening the Baidu webpage, use `dns.qry.name contains "baidu.com"` to locate the packet related to domain name resolution. Extract the response's answer to get the IP address. Then, use the `ip.addr` filter to find the process of establishing a connection with this IP. This process is identical to the one described above. The details are 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

No.	Time	Source	Destination	Protocol	Length	Info
1556	2023-12-10 21:08:14.391821	100.80.176.18	182.61.200.48	TCP	66	56973 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1587	2023-12-10 21:08:14.428732	182.61.200.48	100.80.176.18	TCP	66	443 → 56973 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=32 SACK_PERM
1588	2023-12-10 21:08:14.428835	100.80.176.18	182.61.200.48	TCP	54	56973 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0

1. The sequence number is 4249678530, 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 3069923765, the relative sequence number is 0

2. The value of relative ack number is 1, the raw acknowledgment number is 3112092635

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

337	2023-12-29 14:50:47.149251	100.80.176.18	192.168.66.4	TCP	66	64004 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
338	2023-12-29 14:50:47.149251	100.80.176.18	192.168.66.4	TCP	66	64003 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
340	2023-12-29 14:50:47.154163	192.168.66.4	100.80.176.18	TCP	62	443 → 64003 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 WS=512
341	2023-12-29 14:50:47.154163	192.168.66.4	100.80.176.18	TCP	62	443 → 64004 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 WS=512

Wireshark · 分组 340 · WLAN					
> [Conversation completeness: Incomplete, DATA (15)]					
[TCP Segment Len: 0]					
Sequence Number: 0 (relative sequence number)					
Sequence Number (raw): 3069923765					
[Next Sequence Number: 1 (relative sequence number)]					
Acknowledgment Number: 1 (relative ack number)					
Acknowledgment number (raw): 3112092635					
0111 = Header Length: 28 bytes (7)					
▼ Flags: 0x012 (SYN, ACK)					
000. = Reserved: Not set					

54	64004 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
54	64004 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
686	Client Hello (SNI=sse.tongji.edu.cn)
654	Client Hello (SNI=sse.tongji.edu.cn)
56	443 → 64003 [ACK] Seq=1 Ack=601 Win=15872 Len=0
56	443 → 64004 [ACK] Seq=1 Ack=633 Win=15872 Len=0
314	Server Hello, Change Cipher Spec, Application Data, Application Data
134	Change Cipher Spec, Application Data
314	Server Hello, Change Cipher Spec, Application Data, Application Data
134	Change Cipher Spec, Application Data
133	Application Data
133	Application Data

Q 2.3

In a two-way handshake, only the initiator's starting sequence number can be confirmed, not the other party's.

Consider this: The Client sends a request to the Server, which gets delayed in the network. The Client resends the request. The Server receives it, sends a confirmation, establishes the connection, and begins data transfer. Once data transfer is complete, the connection is disconnected.

However, the Client's first request reaches the Server after disconnection. The Server sends a confirmation, establishing a connection. But the Client, having completed its data service, does nothing, leaving the Server waiting for data interaction and wasting resources.

Thus, a handshake can't confirm a client's connection request, making it unfeasible.

Q 2.4

I chose the No.2 segment.

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

ip.addr==100.75.250.48 && ip.addr==146.56.196.163						
No.	Time	Source	Destination	Protocol	Length	Info
1	2023-06-30 14:29:20.844633	146.56.196.163	100.75.250.48	TCP	1478	8081 → 50555 [ACK] Seq=1 Ack=1 Win=501 Len=1424
2	2023-06-30 14:29:20.844658	146.56.196.163	100.75.250.48	TCP	1478	8081 → 50555 [PSH, ACK] Seq=1425 Ack=1 Win=501 Len=1424
3	2023-06-30 14:29:20.844669	100.75.250.48	146.56.196.163	TCP	54	50555 → 8081 [ACK] Seq=1 Ack=2849 Win=8332 Len=0

▼ Frame 2: 1478 bytes on wire (11824 bits), 1478 bytes captured (11824 bits) on interface \Device\NPF_{71477CD3-C17E-4530-AD17-EE88393F9F4F}, id 0

Section number: 1

> Interface id: 0 (\Device\NPF_{71477CD3-C17E-4530-AD17-EE88393F9F4F})

Encapsulation type: Ethernet (1)

Arrival Time: Jun 30, 2023 14:29:20.844658000 中国标准时间

UTC Arrival Time: Jun 30, 2023 06:29:20.844658000 UTC

Epoch Arrival Time: 1688106560.844658000

[Time shift for this packet: 0.000000000 seconds]

[Time delta from previous captured frame: 0.000025000 seconds]

[Time delta from previous displayed frame: 0.000025000 seconds]

[Time since reference or first frame: 0.000025000 seconds]

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

ip.addr==100.75.250.48 && ip.addr==146.56.196.163						
No.	Time	Source	Destination	Protocol	Length	Info
1	2023-06-30 14:29:20.844633	146.56.196.163	100.75.250.48	TCP	1478	8081 → 50555 [ACK] Seq=1 Ack=1 Win=501
2	2023-06-30 14:29:20.844658	146.56.196.163	100.75.250.48	TCP	1478	8081 → 50555 [PSH, ACK] Seq=1425 Ack=1
3	2023-06-30 14:29:20.844669	100.75.250.48	146.56.196.163	TCP	54	50555 → 8081 [ACK] Seq=1 Ack=2849 Win=8

▼ Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{71477CD3-C17E-4530-AD17-EE88393F9F4F}, id 0

Section number: 1

> Interface id: 0 (\Device\NPF_{71477CD3-C17E-4530-AD17-EE88393F9F4F})

Encapsulation type: Ethernet (1)

Arrival Time: Jun 30, 2023 14:29:20.844669000 中国标准时间

UTC Arrival Time: Jun 30, 2023 06:29:20.844669000 UTC

Epoch Arrival Time: 1688106560.844669000

[Time shift for this packet: 0.000000000 seconds]

[Time delta from previous captured frame: 0.000011000 seconds]

[Time delta from previous displayed frame: 0.000011000 seconds]

[Time since reference or first frame: 0.000036000 seconds]

3. 1424

```

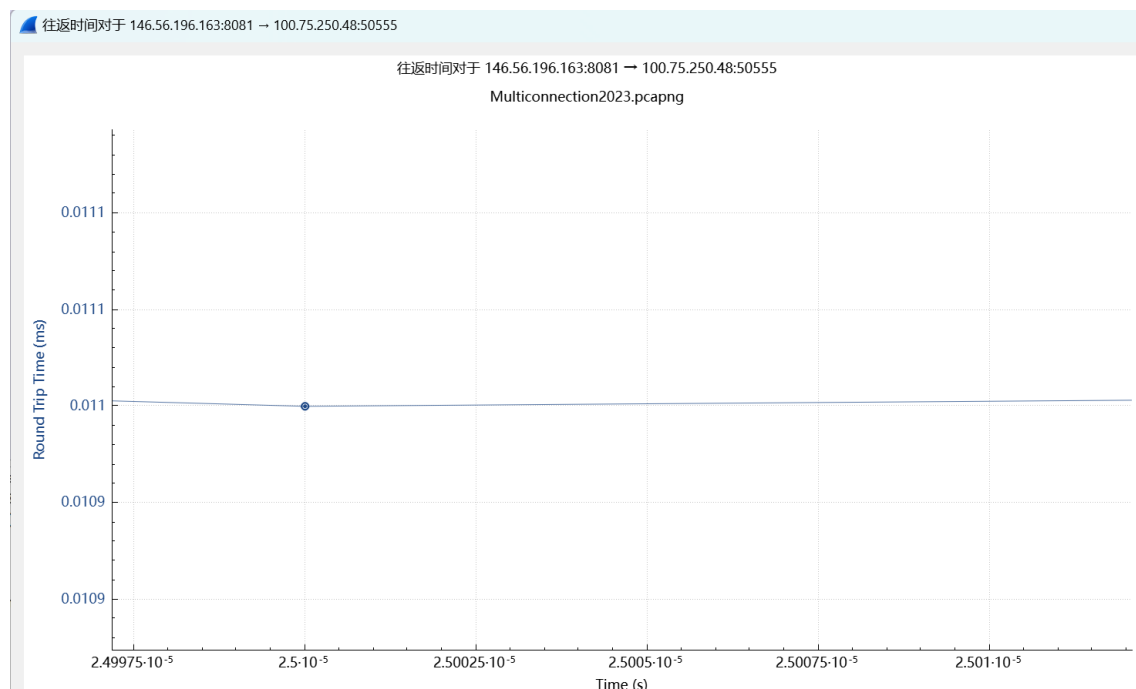
Transmission Control Protocol, Src Port: 8081, Dst Port: 50555, Seq: 1425, Ac
Source Port: 8081
Destination Port: 50555
[Stream index: 0]
> [Conversation completeness: Incomplete (60)]
[TCP Segment Len: 1424]
Sequence Number: 1425 (relative sequence number)
Sequence Number (raw): 112287546
[Next Sequence Number: 2849 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 1778168879
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
Window: 501
[Calculated window size: 501]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x0f9f [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [Timestamps]
> [SEQ/ACK analysis]
TCP payload (1424 bytes)
Data (1424 bytes)
Data [truncated]: de8f1a01313bbb2a2360728ffb51c360728faeca3018f2f656d4d730
[Length: 1424]

```

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



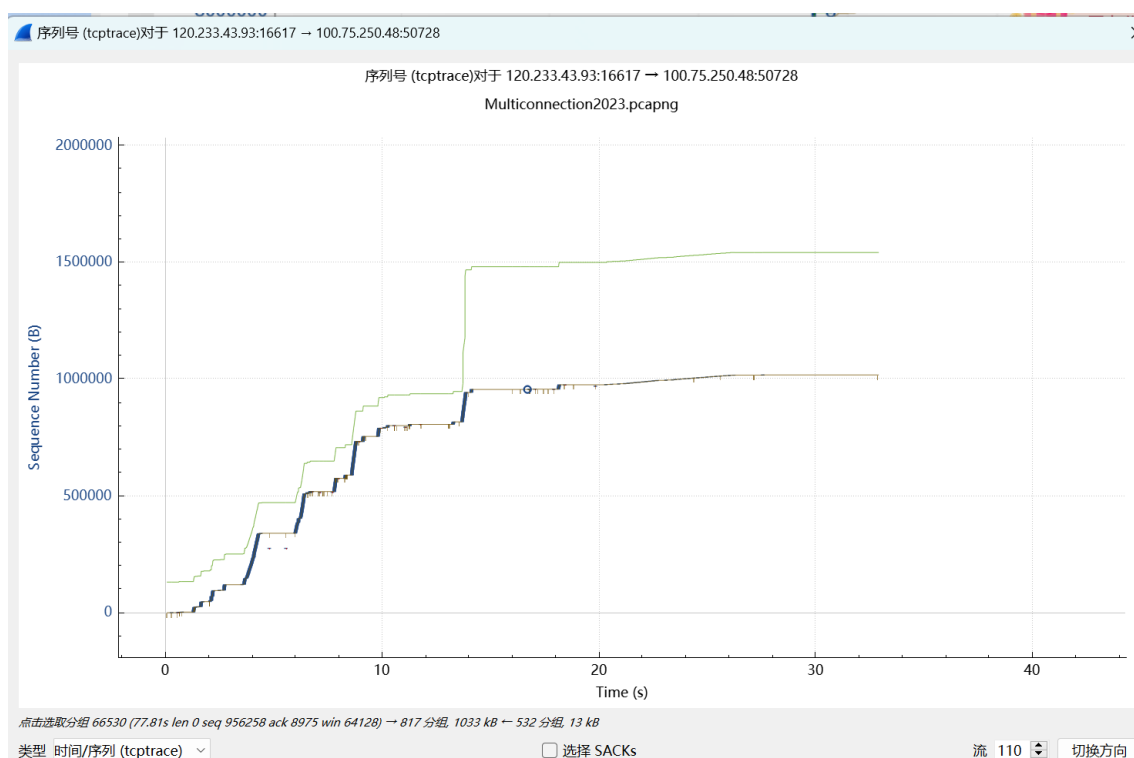
Another method: Click to view the detailed information of the ACK datagram, which will display its RTT.

▼ [SEQ/ACK analysis]

[\[This is an ACK to the segment in frame: 2\]](#)

[The RTT to ACK the segment was: 0.000011000 seconds]

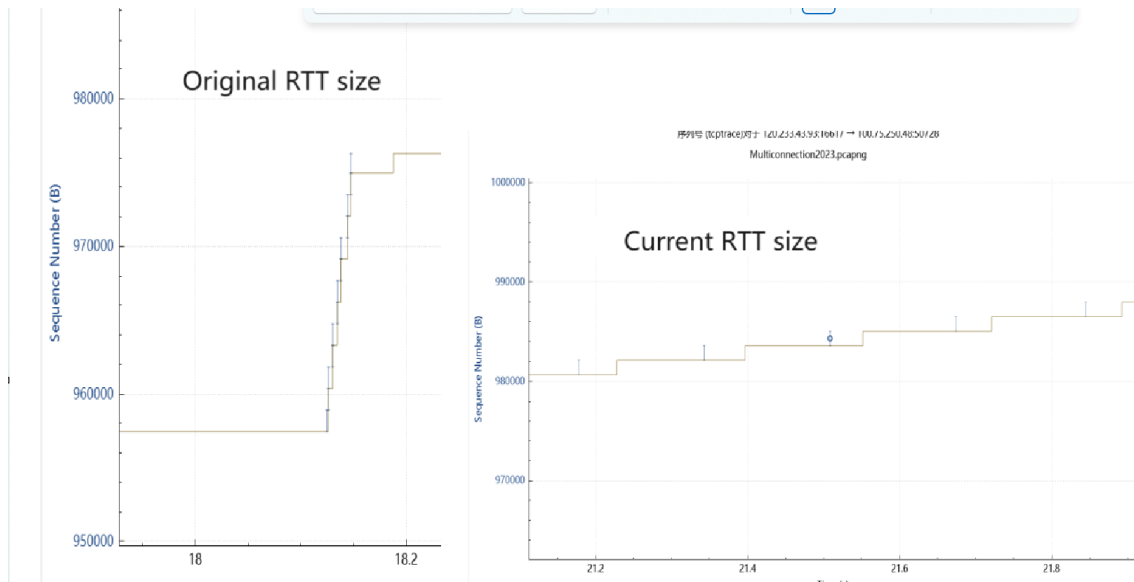
1. The concept is the full-duplex nature of TCP, which means that data can flow independently in both directions. That's why in full-duplex communication, tcptrace follows the direction of transmission. Changes as they change. For example, when you open a web page in a browser, the computer (client) 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 due to network congestion, hardware issues, or software bugs. These can cause packet loss as the network becomes overloaded, hardware malfunctions, or software errors occur.

Preventing transmission failures involves implementing advanced congestion control algorithms, maintaining hardware and software, and designing the network to avoid congestion, ensure capacity, and provide redundancy.

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.

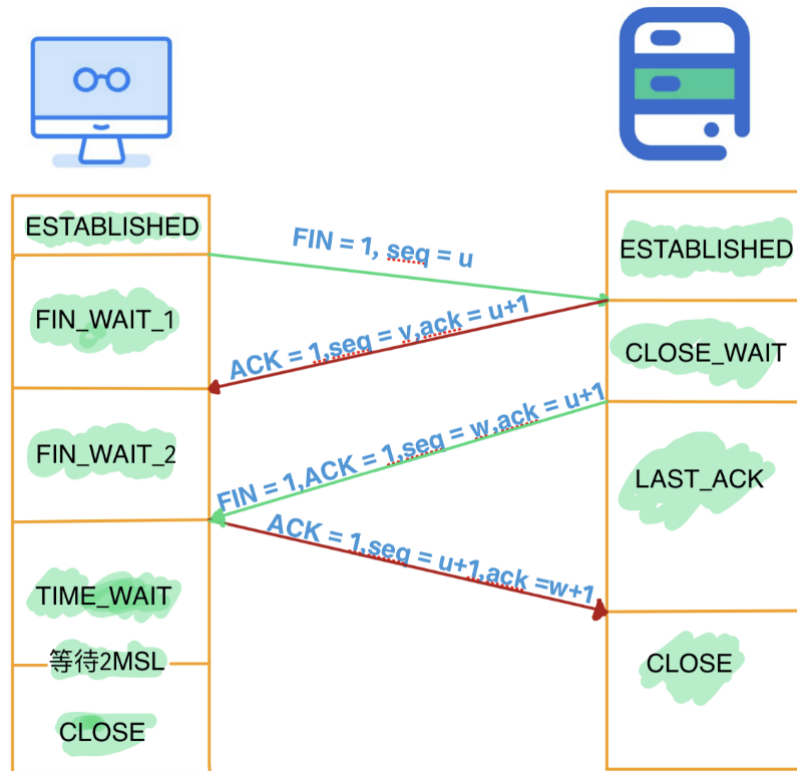
Q 2.7

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.

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.