Part A

Introduction

1. Place assignment2.pcap on the same folder and run analysis_pcap_tcp.py.
2. All answers are printed as the pic as follows at once.
3. The TCP unpack code is in Packet.py, which is also in the same folder.

1. Count the number of TCP flows initiated from the sender

Deserialize TCP packets through Packet.py, then we use (sender_port, receiver_port, receiver_ip) as key to distinct each flow.

```
======== Part A. Q1 ========
src_port: 43498, dst_port: 80, ip_dst: 128.208.2.198, count: 1
src_port: 43500, dst_port: 80, ip_dst: 128.208.2.198, count: 1
src_port: 43502, dst_port: 80, ip_dst: 128.208.2.198, count: 1
TCP flows sent from 130.245.145.12: 3
```

2. For each TCP flow

(a) For the first 2 transactions after the TCP connection is set up (from sender to receiver), get the values of the Sequence number, Ack number, and Receive Window size. Explain these values.

The results for each TCP flow are as follows:

```
======== Part A. Q2(a) ========
src_port: 43498, dst_port: 80, ip_dst: 128.208.2.198, win_scale: 14
SENT seq: 705669103, ack: 1921750144, win_size: 3, win_bytes: 49152, len: 24
SENT seq: 705669127, ack: 1921750144, win_size: 3, win_bytes: 49152, len: 1448
RECEIVE seq: 1921750144, ack: 705669127, win_size: 3, win_bytes: 49152, len: 0
RECEIVE seq: 1921750144, ack: 705670575, win_size: 3, win_bytes: 49152, len: 0

src_port: 43500, dst_port: 80, ip_dst: 128.208.2.198, win_scale: 14
SENT seq: 3636173852, ack: 2335809728, win_size: 3, win_bytes: 49152, len: 24
SENT seq: 3636173876, ack: 2335809728, win_size: 3, win_bytes: 49152, len: 1448
RECEIVE seq: 2335809728, ack: 3636173876, win_size: 3, win_bytes: 49152, len: 0
RECEIVE seq: 2335809728, ack: 3636175324, win_size: 3, win_bytes: 49152, len: 0

src_port: 43502, dst_port: 80, ip_dst: 128.208.2.198, win_scale: 14
SENT seq: 2558634630, ack: 3429921723, win_size: 3, win_bytes: 49152, len: 24
SENT seq: 2558634654, ack: 3429921723, win_size: 3, win_bytes: 49152, len: 1448
RECEIVE seq: 3429921723, ack: 2558634654, win_size: 3, win_bytes: 49152, len: 0
RECEIVE seq: 3429921723, ack: 2558636102, win_size: 3, win_bytes: 49152, len: 0
```

TCP Receive Window

1. It is used to give the sender an idea of how much free buffer space is available at the receiver
2. rwnd = RcvBuffer – [LastByteRcvd – LastByteRead]. Initially, the receiver sets rwnd = RcvBuffer. To ensure the receiver has LastByteRcvd – LastByteRead <= RcvBuffer, the sender makes sure LastByteSent – LastByteAcked <= rwnd.
3. win_scale is retrieved from [SYN] packet's option segment sent from the client.
4. win_bytes: win data in TCP header left shifted by win_scale bits: win_size << win_scale.

Take flow 1 for example (similar analysis applies to the other TCP flows):

1) SENT seq: 705669103, ack: 1921750144, win_size: 3, len: 24
   Client sent 24 bytes data to the server, thus its next seq would be 705669103+24 = 705669127. Ack 1921750144 means the expected seq of its next received packets.

2) SENT seq: 705669127, ack: 1921750144, win_size: 3, len: 1448
   Server hasn't replied to the last packets, while the client continues sending data with 1448 bytes.

3) RECEIVE seq: 1921750144, ack: 705669127, win_size: 3, len: 0
   Server packet returned with expected seq of 1921750144, and its ack 25 is equal to seq of packet 1 plus 24, which means it is a response to packet 1 and has received 24 bytes of data.

4) RECEIVE seq: 1921750144, ack: 705670575, win_size: 3, len: 0
   Server packet returned with expected seq of 1921750144, and its ack 705670575 is equal to seq of packet 2 plus 1448(705669127+1448), which means it is a response to packet 2 and has received 1448 bytes of data.

(b) Compute the throughput at the receiver. You can make assumptions on what you want to include as part of the throughput estimation.

We record the start and end time of each flow, thus we have the interval time. Then the throughput is calculated by the total bytes(header and payload, IP, TCP, and link layers are included) divided by the interval time.

```
======== Part A. Q2(b) ========
src_port: 43498, dst_port: 80, interval: 2.01 seconds
total: 86638336 bytes, throughput: 43.095 Mbps

src_port: 43500, dst_port: 80, interval: 8.32 seconds
total: 88076864 bytes, throughput: 10.586 Mbps

src_port: 43502, dst_port: 80, interval: 0.74 seconds
total: 9013552 bytes, throughput: 12.176 Mbps
```

(c) **Compute the loss rate for each flow. The loss rate is the number of packets not received divided by the number of packets sent. The loss rate is an application layer metric. So think about what makes sense when defining loss rate.**

Here the loss packets are considered both the sender side and receiver side. For a particular sequence number x, if it is duplicated(retransmission) then we increase loss by 1, otherwise, we see if the acknowledged number of x is duplicated(duplicate ACKs) which we will also consider a loss. Thus the loss rate is defined by the loss / transmitted, in which the transmitted is defined by the number of the distinct SEQ from the sender.

```
======== Part A. Q2(b) ========
loss: 3, transmitted: 6972, port: 43498, loss rate: 0.0004302925989672978
loss: 96, transmitted: 6972, port: 43500, loss rate: 0.013769363166953529
loss: 0, transmitted: 727, port: 43502, loss rate: 0.0
```

(d) Estimate the average RTT. Now compare your empirical throughput from (b) and the theoretical throughput (estimated using the formula derived in class). Explain your comparison.

For a sent packet of seq x, its expected ack y is either x+1(for SYN) or x+len(tcp.payload), thus the RTT is the delta time between the packet of seq x from sender and packet of ack y from the receiver.

The theoretical throughput is calculated by $\frac{1.22\ MSS}{RTT\ \sqrt{loss}}$

The theoretical results returned by the program:

```
======== Part A. Q2(c) ========
port: 43498, avg_rtt: 0.073518 s, MSS: 1460 bytes, theoretical throughput: 9.38021 Mbps
port: 43500, avg_rtt: 0.087968 s, MSS: 1460 bytes, theoretical throughput: 1.38582 Mbps
port: 43502, avg_rtt: 0.073273 s, MSS: 1460 bytes, theoretical throughput: inf Mbps
```

Compared to the results in (b), the empirical results are higher than the theoretical throughput. Apply this formula to the first flow, to achieve the same empirical throughput of 43 Mbps, the loss rate would be at around 2e-5, which is impossible, since nowadays the network environment has changed a lot, so this has become outdated.