LAB 3\_Jinbo Li\_Taoran Liu

We have read and understood the course academic integrity policy.

Q1:

What is the IP address and TCP port number used by the client computer (source) that is transferring the alice.txt file to gaia.cs.umass.edu? To answer this question, it’s probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the “details of the selected packet header window” (refer to Figure 2 in the “Getting Started with Wireshark” Lab if you’re uncertain about the Wireshark windows).

A:

The IP address is 192.168.86.68. The TCP port number is 55639.

图形用户界面, 文本, 应用程序

描述已自动生成

Q2:

What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

A:

The IP address is 128.119.245.12. The port number is 80.

图形用户界面, 应用程序

描述已自动生成

Q3:

What is the *sequence number* of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? (Note: this is the “raw” sequence number carried in the TCP segment itself; it is *NOT* the packet # in the “No.” column in the Wireshark window. Remember there is no such thing as a “packet number” in TCP or UDP; as you know, there *are* sequence numbers in TCP and that’s what we’re after here. Also note that this is not the relative sequence number with respect to the starting sequence number of this TCP session.). What is it in this TCP segment that identifies the segment as a SYN segment?

A：

The sequence number is 0. The SYN flag is set to 1, which identifies the segments as a SYN segment.

应用程序

低可信度描述已自动生成

Q4:

What is the *sequence number* of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is it in the segment that identifies the segment as a SYNACK segment? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value?

A:

The sequence number is 0. The SYN flag and ACK flag are set to 1, which identify the segment as a SYNACK segment. The value of the ACK field is 1. This value equals the initial sequence number+1.

图形用户界面, 应用程序

描述已自动生成

Q5:

What is the sequence number of the TCP segment containing the header of the HTTP POST command? Note that in order to find the POST message header, you’ll need to dig into the packet content field at the bottom of the Wireshark window, *looking for a segment with the ASCII text “POST” within its DATA field.*

A:

The sequence number is 1.

图形用户界面, 应用程序

中度可信度描述已自动生成

Q6:

At what time was the first segment (the one containing the HTTP POST) in the data-transfer part of the TCP connection sent?

A:

The time is 0.024047.

日历

低可信度描述已自动生成

At what time was the ACK for this first data-containing segment received?

A:

The time is 0.052671.

日历

描述已自动生成

What is the RTT for this first data-containing segment?

A: RTT = 0.052671 - 0.024047 = 0.028624.

What is the RTT value between the second data-carrying TCP segment and its ACK?

A:

RTT = 0.052676 – 0.024048 = 0.028628.

日历

描述已自动生成

What is the EstimatedRTT value (see Section 3.5.3, in the text) after the ACK for the second data-carrying segment is received? Assume that in making this calculation after the received of the ACK for the second segment, that the initial value of EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 242, and a value of α = 0.125.

**A**: EstimatedRTT = 0.875\*EstimatedRTT + 0.125\*SampleRTT = 0.875 \* 0.028624 + 0.125 \* 0.028628 = 0.0286245.

Q7:

What is the length (header plus payload) of each of the first four data-carrying TCP segments?

A:

Length of all TCP segment = 32 + 1448 = 1480 bytes.

图形用户界面, 表格

中度可信度描述已自动生成

1st TCP segment

图形用户界面, 表格

中度可信度描述已自动生成

2nd TCP segment

表格

中度可信度描述已自动生成

3th TCP segment

图形用户界面, 表格

描述已自动生成

4th TCP segment

**Q8:**

What is the minimum amount of available buffer space advertised to the client by gaia.cs.umass.edu among these first four data-carrying TCP segments? Does the lack of receiver buffer space ever throttle the sender for these first four data-carrying segments?

A:

The minimum amount is 31872. According to the first to the fourth TCP segment, the lack of receiver buffer space does not throttle the sender for these first four data-carrying segments.

表格

描述已自动生成

1st TCP ACK segment

图形用户界面, 表格

描述已自动生成

2nd TCP ACK segment

图形用户界面, 表格

描述已自动生成

3th TCP ACK segment

表格

描述已自动生成

4th TCP ACK segment

图形用户界面, 应用程序, 表格

描述已自动生成

1st TCP segment

Q9:

Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

A:

No. The sequence number of all segments increases with time. The sequence number of one TCP segment is larger than previous segment and smaller than later one; thus, there are no retransmitted segments in the trace file.

表格

描述已自动生成

2nd TCP segment

表格

描述已自动生成

3th TCP segment

表格

描述已自动生成

4th TCP segment

Q10：

How much data does the receiver typically acknowledge in an ACK among the first ten data-carrying segments sent from the client to gaia.cs.umass.edu? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 in the text) among these first ten data-carrying segments?

A:

According to the formula: acknowledged data = #ACK - previous #ACK, ACK1 has 1449 acknowledged data. ACK2 has 1448 (2897 - 1449) acknowledged data. ACK3 has 1448 (4345 - 2897) acknowledged data. ACK4 has 1448 (5793 - 4345) acknowledged data. ACK5 has 1448 acknowledged data. ACK6 has 1448 acknowledged data. ACK7 has 1448 acknowledged data. ACK8 has 1448 acknowledged data. ACK9 has 1448 acknowledged data. ACK10 has 1448 acknowledged data.

There is no case where the receiver is ACKing every other received segment among first ten data-carrying segments. It is because the length of each TCP segment is equal to the acknowledged data.

日历

中度可信度描述已自动生成

日历

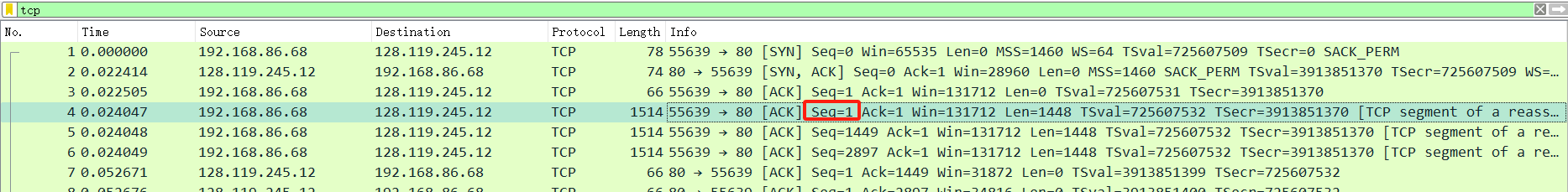
描述已自动生成

**Q11:**

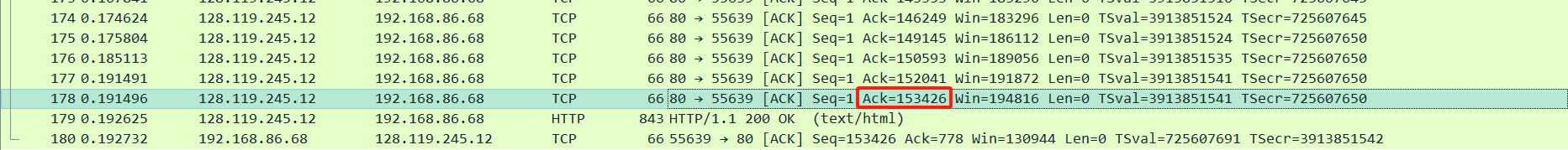
What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

A:

We can assume that time period as the whole TCP connection time. Thus, the throughput = total bytes transferred / total transmission time. The total bytes transferred equals to the difference between the sequence number of first TCP segment and the last ACK value. In this case, the total mount of data = 153426 – 1 = 153425 bytes.



The sequence number of 1st TCP segment = 1, time=0.024047



The sequence number of last ACK = 153426, time = 0.191496

The total transmission time = the time of last ACK - the time of first TCP segment. In this case, the total transmission time = 0.191496 - 0.024047 = 0.167449 second. Therefore, the throughput = 153425 bytes / 0.167449 second = 916249.127 bytes / second

**Q12:**

Consider the “fleets” of packets sent around *t* = 0.025, *t* = 0.053, *t* = 0.082 and *t* = 0.1 in Figure 5. Comment on whether this looks as if TCP is in its slow start phase, congestion avoidance phase or some other phase. Figure 6 shows a slightly different view of this data.

A:

According to the figure 5 and figure 6, when t=0.025, 0.053, 0.082, it looks like as if TCP is in slow start phase because the number of TCP segment increases exponentially. The number of packets is twice the number of packets sent for the previous time. When t=0.1, the delay may increase, causing a delay in sending packets, but still slow start phase.

**Q13: 自己抓包**

Use the *Time-Sequence-Graph(Stevens*) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server in the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu. Answer again question 12 for your trace.

A:

According to the figure 5 and figure 6, when t=0.025, 0.053, 0.082, it looks like as if TCP is in slow start phase because the number of TCP segment increases exponentially. When t=0.1, TCP is in slow start phase because the ...图表, 箱线图

描述已自动生成

5. Measuring bandwidth with Iperf3

a) Start a 10-second TCP transfer from a public server to your computer. Use the –i option on the client to report the TCP throughput every 100 ms. Plot the TCP throughput on the client as a function of time.

A: Input the command “iperf3 -R -p 5201 -c 54.197.223.49 -t 10 -i 0.1”.

图表, 折线图

描述已自动生成

图表, 折线图

描述已自动生成

b) Start a 10-second UDP transfer from a public server to your computer. Use the –b option to set the sending rate to a very high value (higher than the downlink speed of your Internet connection – 200 Mbps should be sufficient). Use the –i option on the client to report the UDP throughput every 100 ms. Plot the UDP throughput on the client as a function of time.

A: Input the command “iperf3 -R -p 5201 -c 54.197.223.49 -u -b 200m -t 10 -i 0.1”.

图形用户界面, 图表, 折线图

描述已自动生成

**c)** Compare the two graphs you plotted in (a) and (b) and write down your observations and possible causes for the observed performance, in case it differs from the expected one.

A:

Throughput fluctuation of TCP is larger than UDP because TCP has congestion control. The throughput of TCP is much higher than UDP.

d) Start a 10-second TCP transfer from a public server to your computer. After 3 seconds, start another 10-second TCP transfer from a different public server (this means that you will have to run two iperf3 clients on your computer). Use the –i option on the client to report the TCP throughput every 100 ms. Plot the TCP throughput of each connection as a function of time on the same graph. Comment on how the two TCP sessions share the bandwidth.

A:

The green line stands for first iperf3 client. The blue line stands for the second iperf3 client. When the throughput of the second session go up, the throughput of the first session go down. When t=6s, throughput of two sessions is consistent. Thus, the two sessions share the bandwidth equally.

图形用户界面, 图表, 折线图

描述已自动生成

e) Start a 10-second TCP transfer from a public server to your computer. After 3 seconds, start a 10-second UDP transfer from a different public server (this means that you will have to run two iperf3 clients on your computer). Use the –i option on the client to report the TCP and UDP throughput every 100 ms. Plot the TCP and UDP throughput as a function of time on the same graph. Comment on how a UDP and a TCP connection share the bandwidth.

A: The green line stands for TCP transfer. The red line stands for UDP transfer. According to the graph, from the beginning (t=3s) to the end of UDP transfer, the bandwidth is always low, while the bandwidth of TCP transfer is always high. Thus, the TCP connection takes up most of bandwidth.

图表, 折线图

描述已自动生成