Nathan Flack

Wireshark Lab #3

1. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?

**Answer: Source IP: 129.168.0.106, Source Port Number: 64845**

**A screenshot of a cell phone

Description automatically generated**



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

**Answer: Dest IP: 128.119.245.12, Dest Port Number: 80**

**A screenshot of a cell phone

Description automatically generated**



If you have been able to create your own trace, answer the following question:

1. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

**Answer: Source IP: 129.168.0.106, Source Port Number: 64845**

**A screenshot of a cell phone

Description automatically generated**



TCP Basics

1. 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? What is it in the segment that identifies the segment as a SYN segment?

**Answer: Relative Sequence Number: “0” (real number is 65F71461 in Hexadecimal); There is a flag that identifies the segment. In this case the “SYN” flag is set to “1” identifying the packet as a SYN segment.**

**A screenshot of a cell phone

Description automatically generated**



1. 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 the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

**Answer: Relative Sequence Number: “0”; Acknowledgement Number: “1”; gaia.cs.umass.edu added 1 to the relative sequence number sent by the source (my computer). Both SYN and ACK flags are set to 1 to identify the segment as a SYN-ACK segment.**

**A screenshot of a cell phone

Description automatically generated**



1. What is the sequence number of the TCP segment containing the HTTP POST command?

**Answer: Relative Sequence Number: “1”**

**A screenshot of a cell phone

Description automatically generated**



1. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3, page 242 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 242 for all subsequent segments.

*Note:* Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the “listing of captured packets” window that is being sent from the client to the gaia.cs.umass.edu server. Then select: *Statistics->TCP Stream Graph->Round Trip Time Graph.*

1. What is the length of each of the first six TCP segments?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sequence # | Time sent | Time ACK rcd | RTT  (seconds) | Est RTT (seconds)  = (1-0.125) \*EstimatedRTT + 0.125\*SampleRTT | Seg Length (bytes) |
| 1 | 23:04:04.903673 | 23:04:04.963112 | 0.0594 | 0.0594 | 674 |
| 675 | 23:04:04.903832 | No ACK | N/A | N/A | 1448 |
| 2123 | 23:04:04.903833 | 23:04:04.963118 | 0.059285 | 0.059386  = .875\*.0594+.125\*.059285 | 1448 |
| 3571 | 23:04:04.963201 | 23:04:05.022891 | 0.05969 | 0.059424  = .875\*.059386+.125\*.05969 | 1448 |
| 5019 | 23:04:04.963202 | No ACK | N/A | N/A | 1448 |
| 6467 | 23:04:04.963203 | 23:04:05.022898 | 0.059695 | 0.059458  = .875\*.059424+.125\*.059695 | 1448 |
| 7915 | 23:04:04.963204 | 23:04:05.022899 | 0.059695 | 0.059458  = .875\*.059458+.125\*.059695 | 1448 |

A screenshot of a cell phone

Description automatically generated



This is an example of one of the packets that provided the information in the table:

A screenshot of a cell phone

Description automatically generated



1. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

**Smallest widow size value is 237 with a calculated window size of 30336.**

A screenshot of a cell phone

Description automatically generated



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

**There are no repeated sequence numbers in all the segments sent, therefore no segments needed to be retransmitted.**

1. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 250 in the text)?

**It is typical for the receiver to acknowledge each segment; however it is often the case that two or three segments will be received before an ACK is sent. In this screenshot, the ACK in line 54 is responding to one TCP segment, but line 95 is ACKing two received segments since the different in the ACK number between 94 and 95 is 2,896 bytes.**

A screenshot of a cell phone

Description automatically generated



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

A close up of a device

Description automatically generated



A screenshot of a cell phone

Description automatically generated



**Throughput = data transferred / time difference**

**= 153361 bytes (from ACK number in line 230) / (23:04:04.903673 – 23.04.05.324665)**

**= 153361 / 0.420992 = 364,285 Bytes/sec = 2,914,278.65 bits/sec = 2.914 Mbps**

TCP congestion control in action

1. 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. Can you identify where TCP’s slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we’ve studied in the text.

**Yes, the “slow start” begins at about 0.04 seconds (seq=556) and continues through sequence 7866 at .124 seconds.**

**Congestion avoidance takes over at 0.3 seconds (packet 18 with seq=9013)**

**In the provided trace, the number of segments sent at one time seem to level off at 6 instead of growing linearly in the idealized behavior in the book.**

This is the Stevens Graph built from the provided Wireshark trace:





1. Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu
2. Can you identify where TCP’s “slow start” phase begins and ends, and where congestion avoidance takes over?

**No, I cannot identify those trends in my graph. It looks like it is showing some kind of “slow start” strategy, but I cannot identify if it changes to congestion avoidance or not. I do not see a linear increase anywhere on the graph. It seems to be increase and decreasing the number of segments sent without a clear pattern.**

1. Comment on ways in which the measured data differs from the idealized behavior of TCP that we’ve studied in the text.

**This behavior seems rather erotic, given what the idealized behavior is designed to be. However, since there were no timeouts, lost packets, or three duplicate acks received during the file transfer, I can see how it might be hard to identify the trends.**

**I am wondering if this graph showed more fidelity on when each segment was sent (instead of just at the 20 millisecond intervals if it would show a continuous “slow start” phase until 0.38 seconds where it might change to a congestion avoidance strategy. See the red line drawn on the graph below. However, the leveling out could be showing the end of the file transfer.**

This is the Stevens Graph built from my Wireshark trace:



