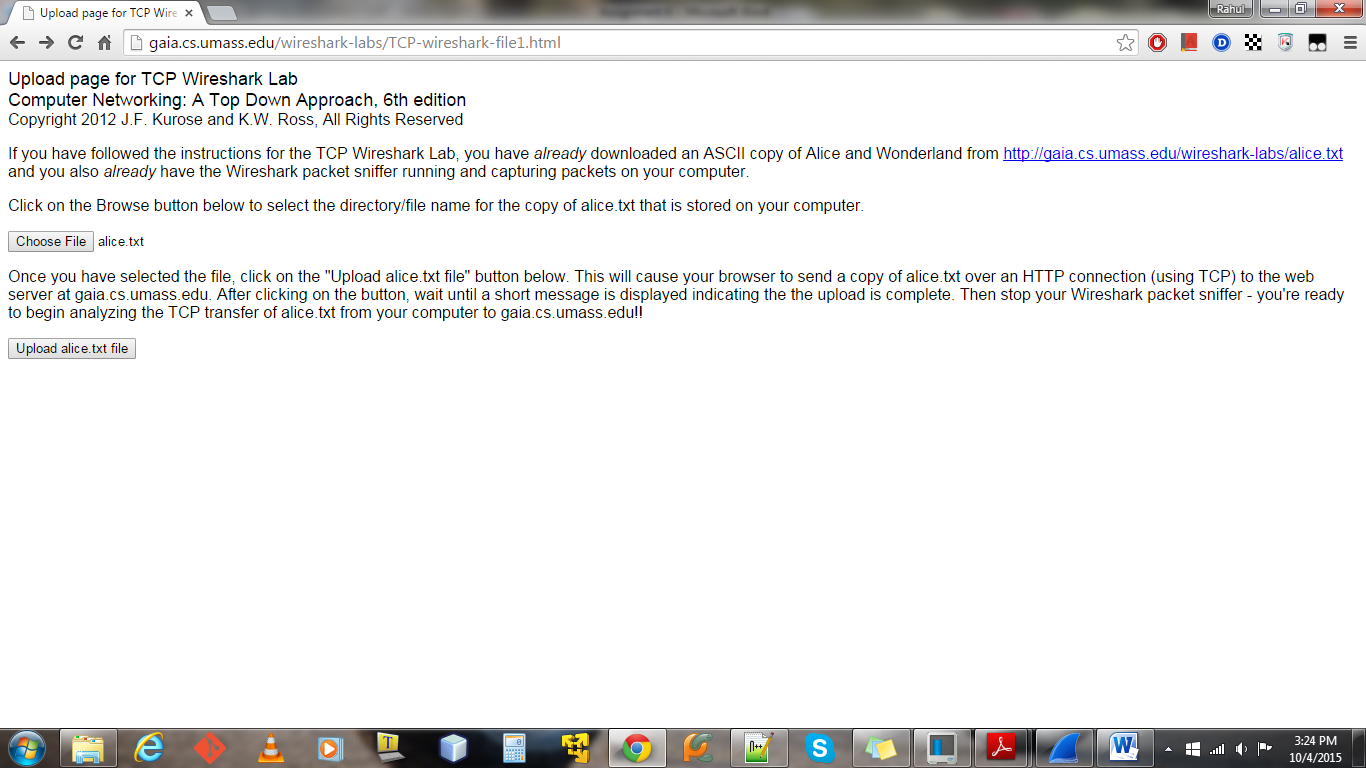
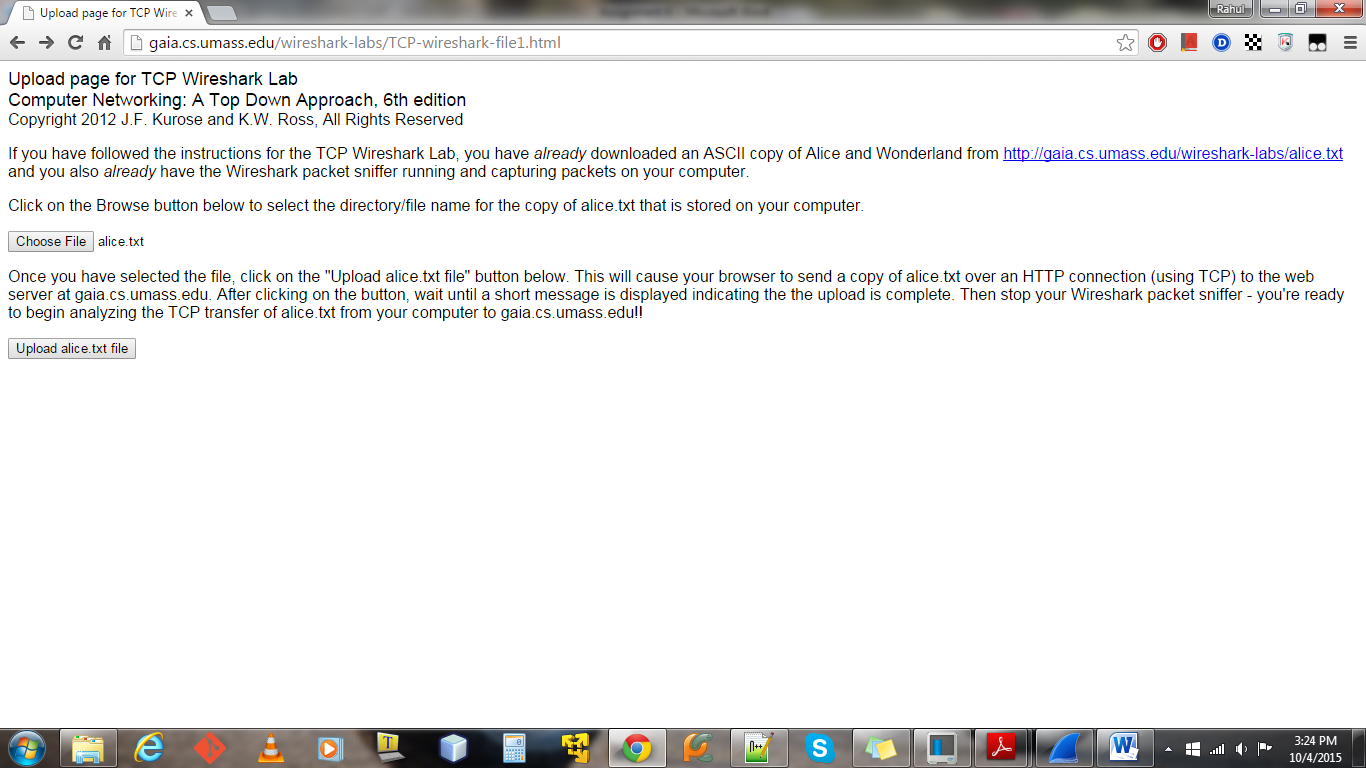
**Assignment 4 - Wireshark Lab: TCP v6.0**

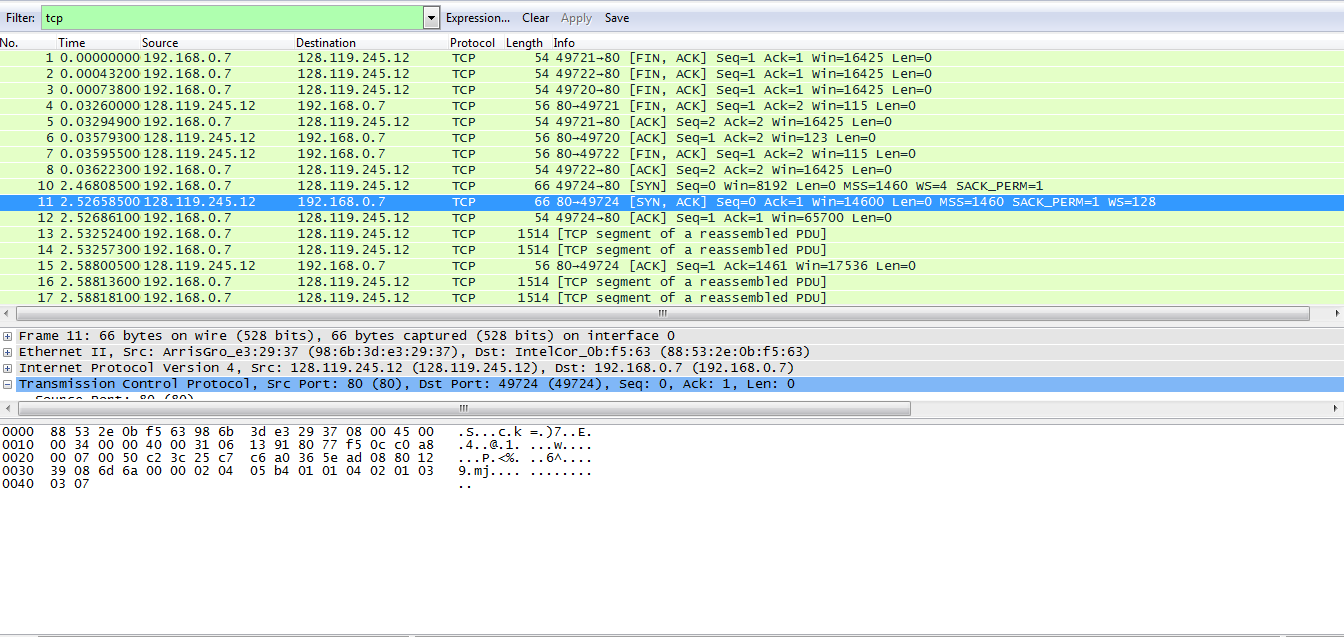
**Answer 1: Capturing a bulk TCP transfer from your computer to a remote server**



**1a. Browser Output**



**1b. Browser Output**

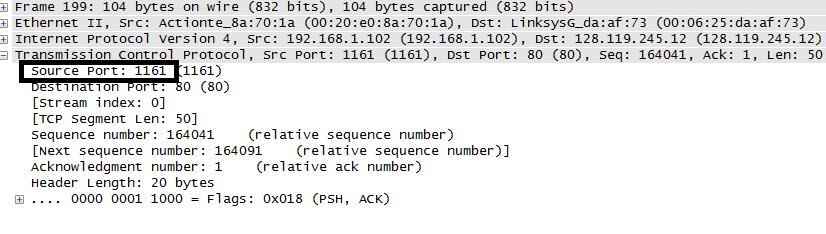
****

**1c. Wireshark Output**

**2. A first look at the captured trace**

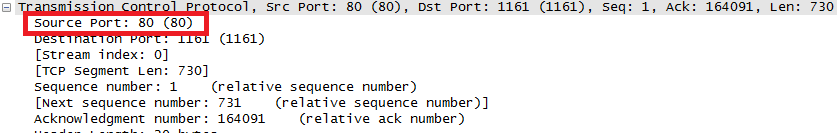
**1.** The IP address of the client is **192.168.1.102** and the Source Port is – **1161**





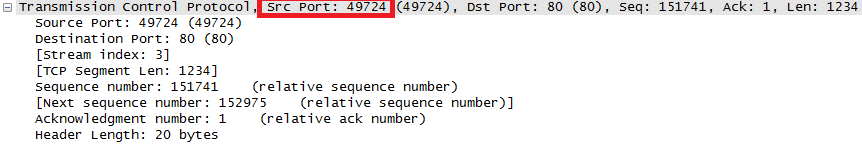
**2.** The IP address of gaia.cs.umass.edu is **128.119.245.12** and the port on which it’s sending and receiving port is **80** (default)





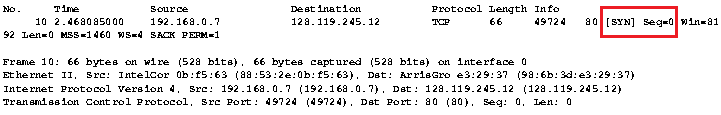
**3.** The IP address of client computer is **192.168.0.7** and port number is **49724**





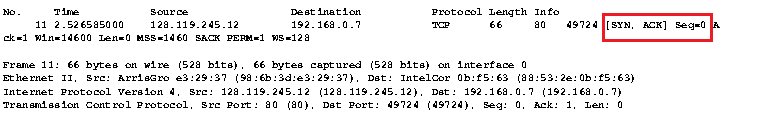
**3. TCP Basics**

**4.** 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 is **0.** The segment has the **SYN flag bit set (1<<1)** that identifies it as a SYN segment**.**

****

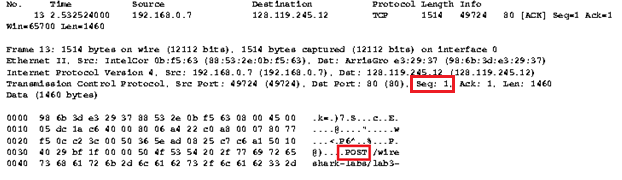
****

**5.** The sequence number of the SYN ACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is **= 0**. The value of the acknowledge field in the SYNACK segment is **= 1**. It’s determined by the sequence number of the next byte expected from the client computer. The segment has the **SYN (1<<1) and ACK(1<<4) bit set** in the flags field that identifies it as a SYN ACK segment.



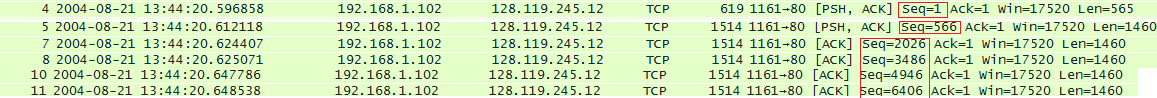


**6.** The sequence number of the TCP segment containing the HTTP POST command is **=1**.

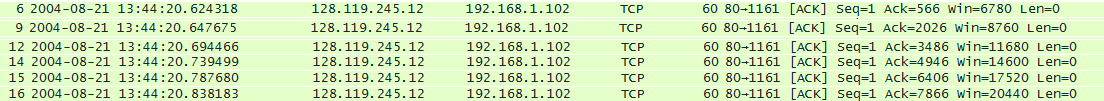


**7.** **Packet trace taken from author from here onwards**

Below table shows the sequence numbers, time sent and time ACK was received

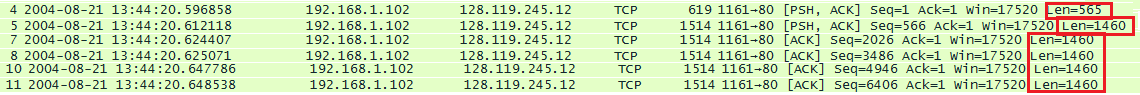


TCP Segments Received



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sequence number | Time Segment was sent | Time ACK was received | Sample RTT(Micros) | Estimated  RTT(Micros) | Estimated  RTT(seconds) |
| 1 | 13:44:20.596858 | 13:44:20.624318 | 27460 | 27460 | 0.027460 |
| 566 | 13:44:20.612118 | 13:44:20.647675 | 35557 | 28472.1 | 0.028472 |
| 2026 | 13:44:20.624407 | 13:44:20.694466 | 70059 | 33670.4 | 0.033670 |
| 3486 | 13:44:20.625071 | 13:44:20.739499 | 114428 | 43765.1 | 0.043765 |
| 4946 | 13:44:20.647786 | 13:44:20.787680 | 139894 | 55781.2 | 0.055781 |
| 6406 | 13:44:20.648538 | 13:44:20.838183 | 189645 | 72514.1 | 0.072514 |

**8.** The length of first Six TCP Segments is 565, 1460, 1460, 1460, 1460, and 1460

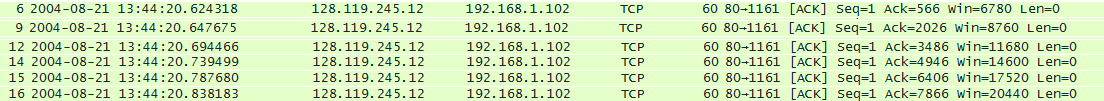


**9.** The receive window is conveyed to the client by the server during acknowledgement and that is **=5840 Bytes.** As per TCP flow control implementation as long as the server sends a receive window to client that it can process without dropping and the client makes sure that **LastByteSent – LastByteAcked <= Window,** there should not be any throttling at client side.



**10.** In the Authors Packet trace there are no re-transmitted segments. It can be seen on Wireshark where info is TCP Retransmission and the number of tries will be performed based on the timeout value. It can also be seen by looking at the sequence number of segment. Two or more segments present in the trace with the same sequence number and spaced between the timeout indicate a retransmission.

**11.** The amount of data ACK’ed by Server is known by which is the expected Byte value or sequence number expected from the sending host. Considering these 6 segments and subtracting each sequence will give the amount of data ACK’ed by server.



|  |  |
| --- | --- |
| 6 | 566 - 1 = 565(1 byte for SYN ACK) |
| 9 | 2026 - 566 = 1460 |
| 12 | 3486 - 2026 =1460 |
| 14 | 4946 – 3486 = 1460 |
| 15 | 6406 – 4946 = 1460 |
| 16 | 7866 – 6406 = 1460 |

On an average it is **1460 bytes**

Yes there can be a scenario where receiver is ACK’ing every other received segment. Here the server is ACK’ing for sequence number (164091) twice. The reason could be that there is no more data from client to send .



**12.** Bytes transferred per unit time for connection = Total bytes transferred for connection/Time from start to end of connection (seconds)

Since the number of bytes can be thought as the last sequence number that is ACK’ed by server and from the Packet trace, it is 164091



Since the first sequence is 1 (for TCP SYN),



This means **164091** bytes were transferred and time for this transfer is 13:44:26.031556 – 13:44:20.570381(SYN and Last ACK) = **5.461175 seconds**

164091/5.461175 = **30046.83 Bytes/sec** is the throughput for this connection

**4. TCP congestion control in action.**

**13, 14.** (13 and 14 answered here both the questions)

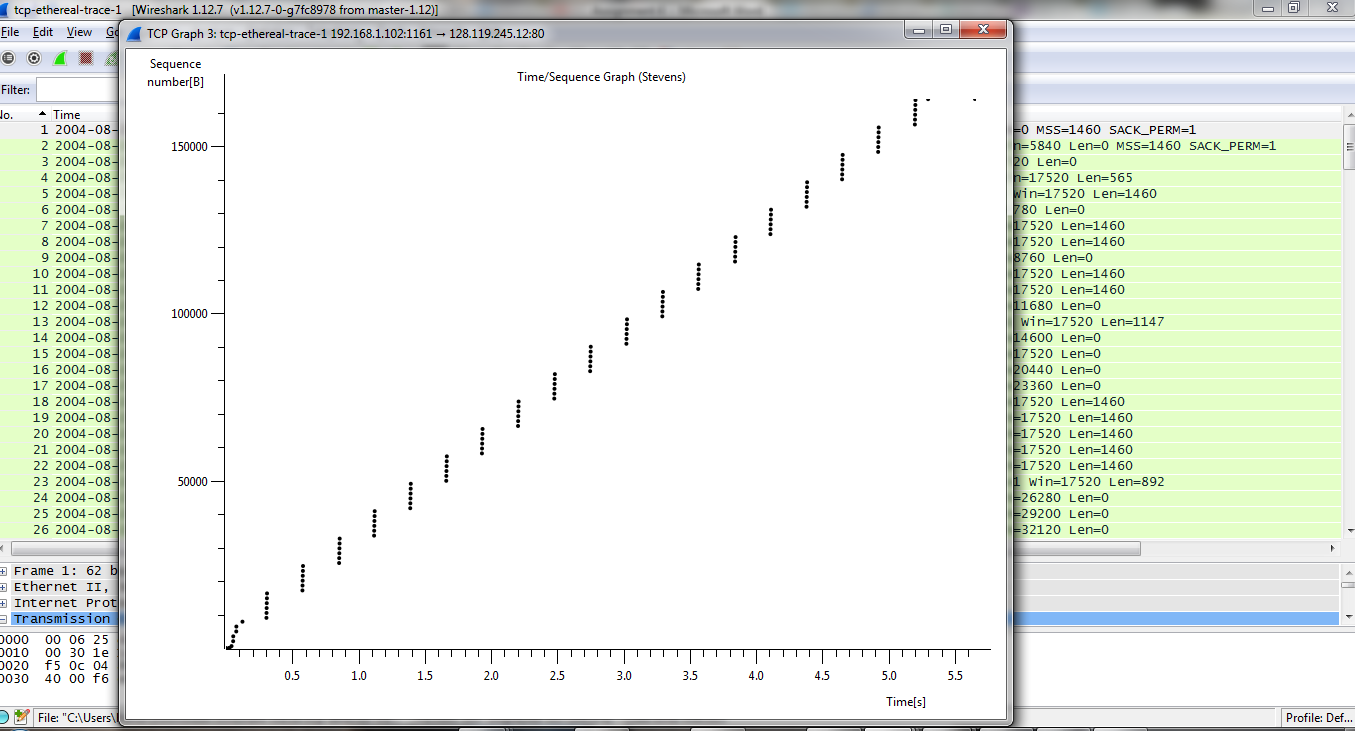


Fig 4.1 Wireshark Time-Sequence-Graph (Stevens)

After analyzing the Time-Sequence-Graph (Stevens), these are my observations:

1. The TCP slow start phase is during the initial sequence of 3 way handshake as we can see clearly from the graph that segments are not overlapping which means there is time taken for the TCP stack at either end systems to initialize .

2. But as the TCP segments between the hosts are exchanged during the 3-way handshake, the window size is clearly understood by both the hosts and thereby, the segments after this are sent uniformly (6 segments at a given time).

From the 3-way hand shake segments, it can be seen that the server has MSS = 1460 and the client has MSS = 1460.

From the graph and packet list, we can see that as time advances the window size of server increases and towards the end it stabilizes to 62780(bytes)



At this point the time elapsed is 21.610201 – 20.570381 is **1.039** at which the window size of server stabilizes and the client has agreed to this and send’s maximum 6 segments at a time.

**1460\*6 = 8760 and 8760 <=62780 hence there is no need of flow control in this case since the client is not overwhelming the server with lot of data and there is no congestion.**

To really understand if there is congestion, we have to check if server changes window size substantially less than the buffer size at client and also check on wireshark if any segments are lost at receiving end.

This is slightly different from the text, since when the window size is **62780** for server, then the client can send 43 (1460\*43 =62780) segments thereby reducing the time for transmission to almost 2 seconds including handshake. However TCP implementation in practice can vary across systems based on buffer sizes and assumptions made for all kinds of networks.