ICT504

IT Networking and Communication

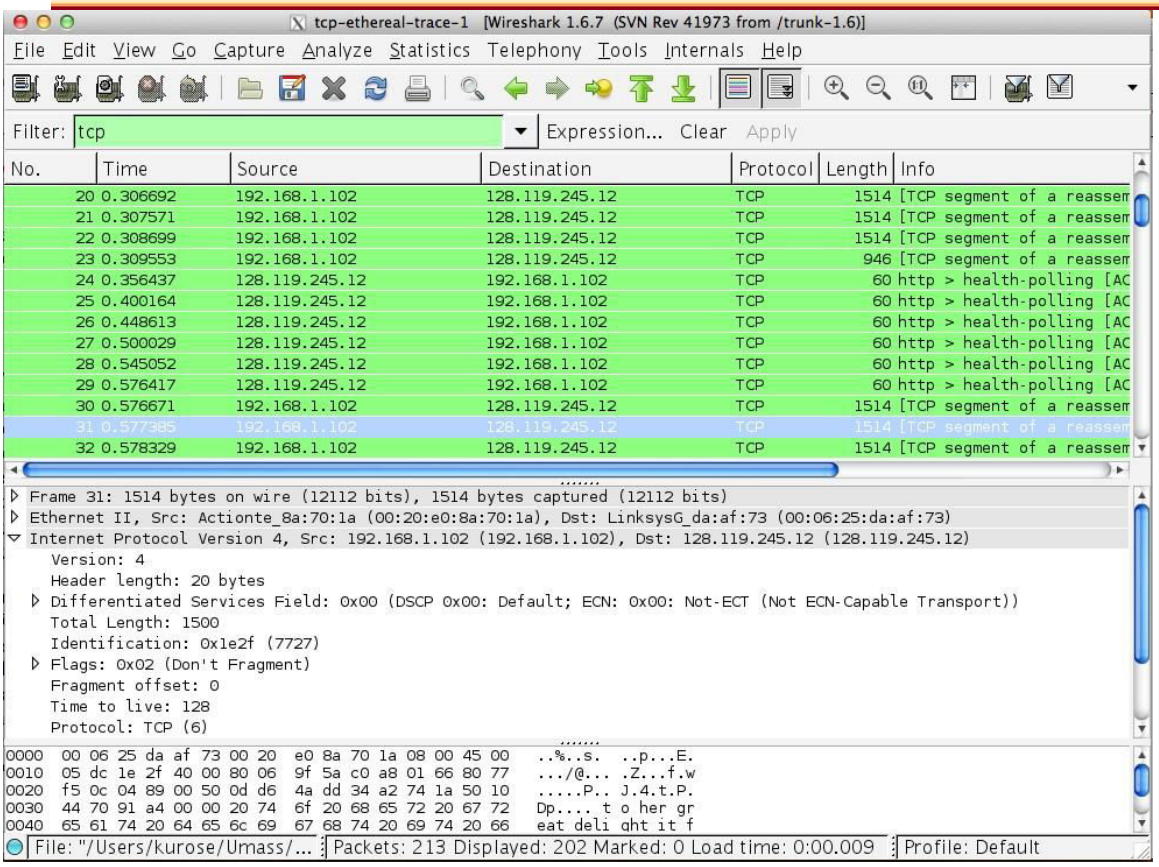
Tutorial 1: TCP 10 Marks

March 2024

In this tutorial, we'll investigate the behavior of the celebrated TCP protocol in detail. We'll do so by analyzing a trace of the TCP segments sent and received in transferring a 150KB file (containing the text of Lewis Carrol's *Alice's Adventures in Wonderland*) from your computer to a remote server. We'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer; we'll see TCP's congestion control algorithm – slow start and congestion avoidance – in action; and we'll look at TCP's receiver-advertised flow control mechanism. We'll also briefly consider TCP connection setup and we'll investigate the performance (throughput and round-trip time) of the TCP connection between your computer and the server.

Before beginning this lab, you'll probably want to review sections 3.5 and 3.7 in the text[1].

[1] References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach, 8th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.*



You can download a captured packet trace file (**tcp-ethereal-trace-1**) from the LMS and open it in the Wireshark.

1. A first look at the captured trace

Before analyzing the behavior of the TCP connection in detail, let's take a high level view of the trace.

- First, filter the packets displayed in the Wireshark window by entering "tcp" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.

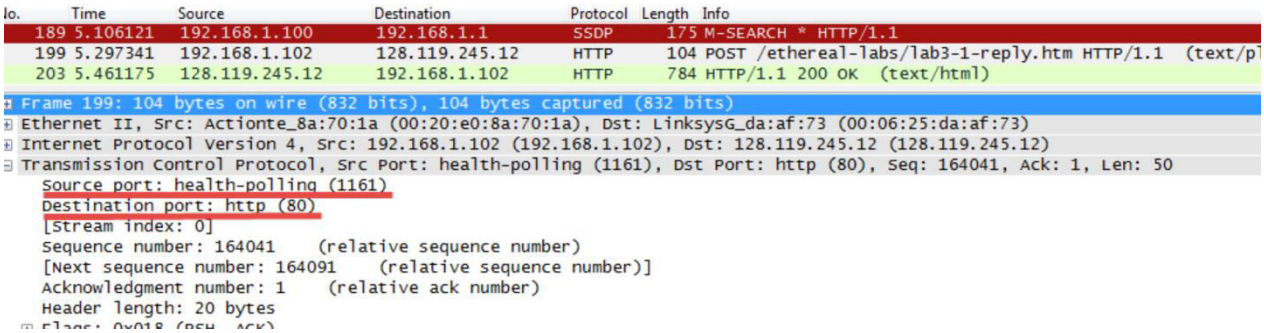
What you should see is series of TCP and HTTP messages between your computer and gaia.cs.umass.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of Wireshark you are using, you might see a series of "HTTP Continuation" messages being sent from your computer to gaia.cs.umass.edu. Recall from our discussion in the earlier HTTP Wireshark lab, that is no such thing as an HTTP Continuation message – this is Wireshark's way of indicating that there are multiple TCP segments being used to carry a single HTTP message. In more recent versions of Wireshark, you'll see "[TCP segment of a reassembled PDU]" in the Info column of the Wireshark display to indicate that this TCP segment contained data that belonged to an upper layer protocol message (in our case here, HTTP). You should also see TCP ACK segments being returned from gaia.cs.umass.edu to your computer.

Answer the following questions (1-11) by downloading the Wireshark captured packet file tcp-ethereal-trace-1 from the LMS and opening it in Wireshark. To justify your answers, you must provide a detailed screenshot from the captured Wireshark file. Otherwise, you will not receive marks for the question.

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? 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".

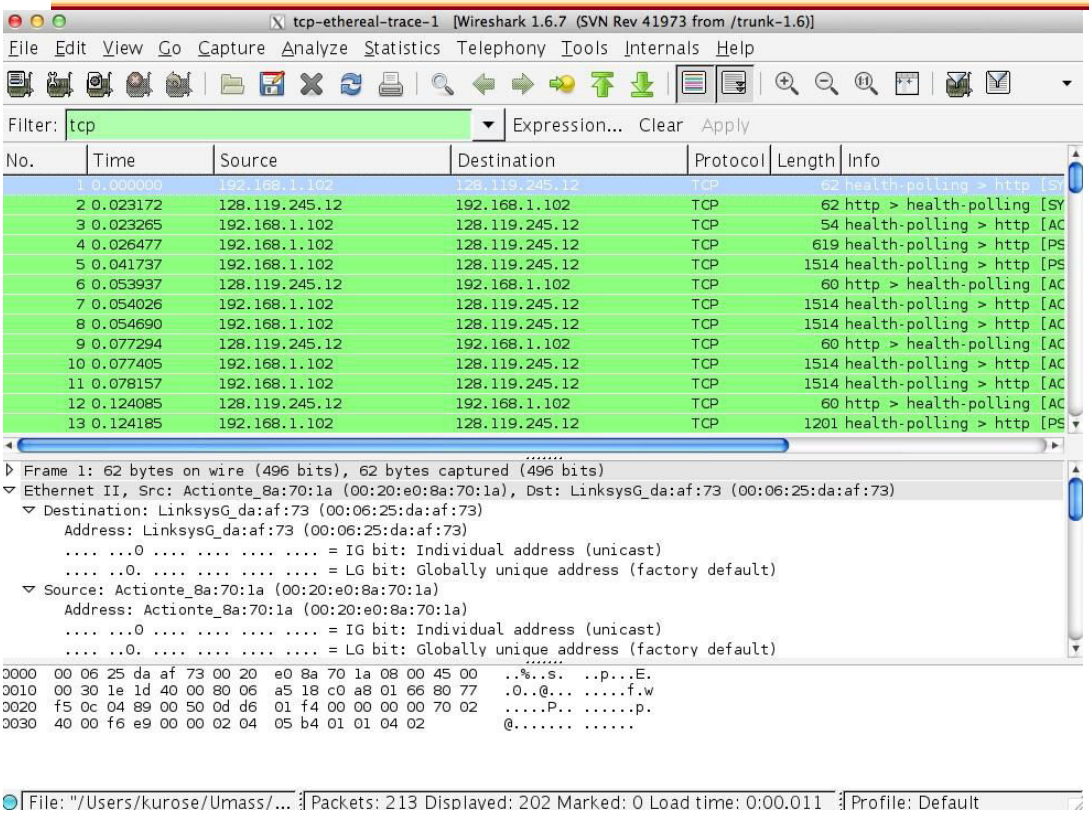
1. **What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?**

Answer:



Answer: IP: 192.168.1.102 and Port: 1161

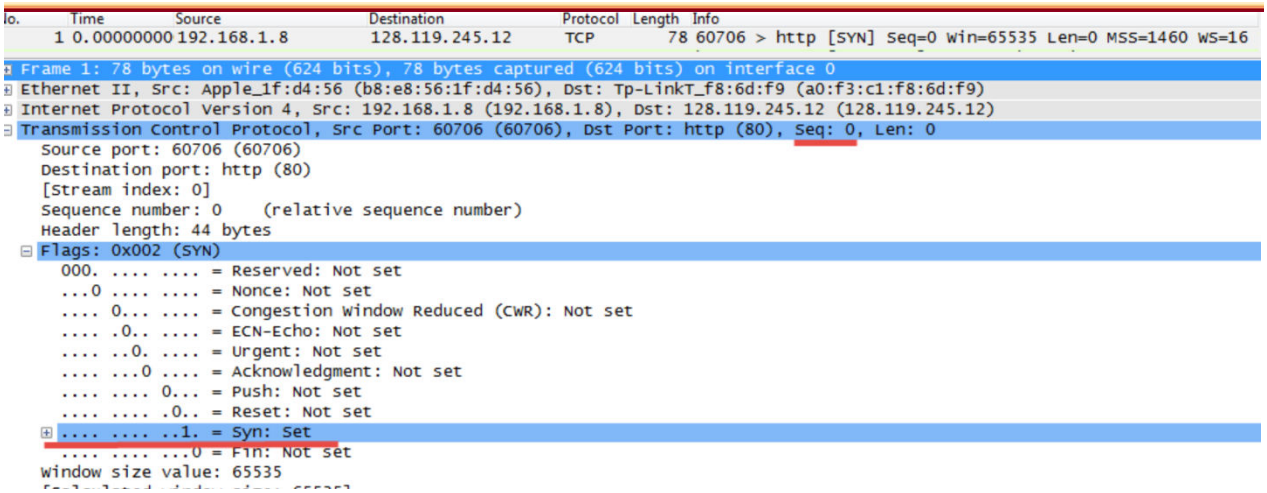
Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select *Analyze->Enabled Protocols.* Then uncheck the HTTP box and select *OK*. You should now see a Wireshark window that looks like:



2. TCP Basics

Answer the following questions for the TCP segments:

2. **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?**

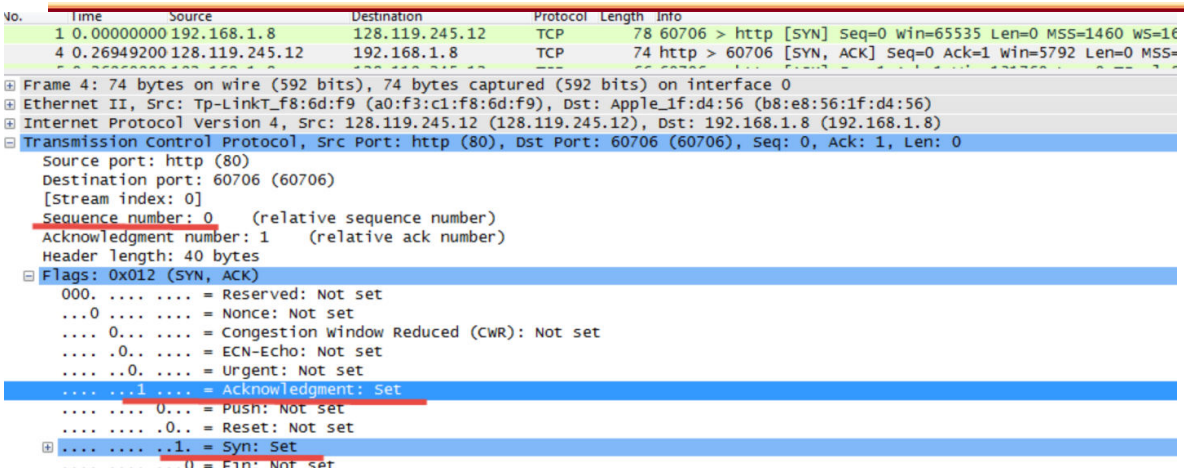


Answer:

In order to create a TCP connection between the client computer and gaia.cs.umass.edu, the sequence number 0 is used in the TCP SYN section. This allows for the establishment of an initial connection.

In the Flags section, which can be obtained by referring to the image that is shown, the Syn flag is set to 1, which indicates that this segment is a SYN segment. This information can be found by looking at the picture.

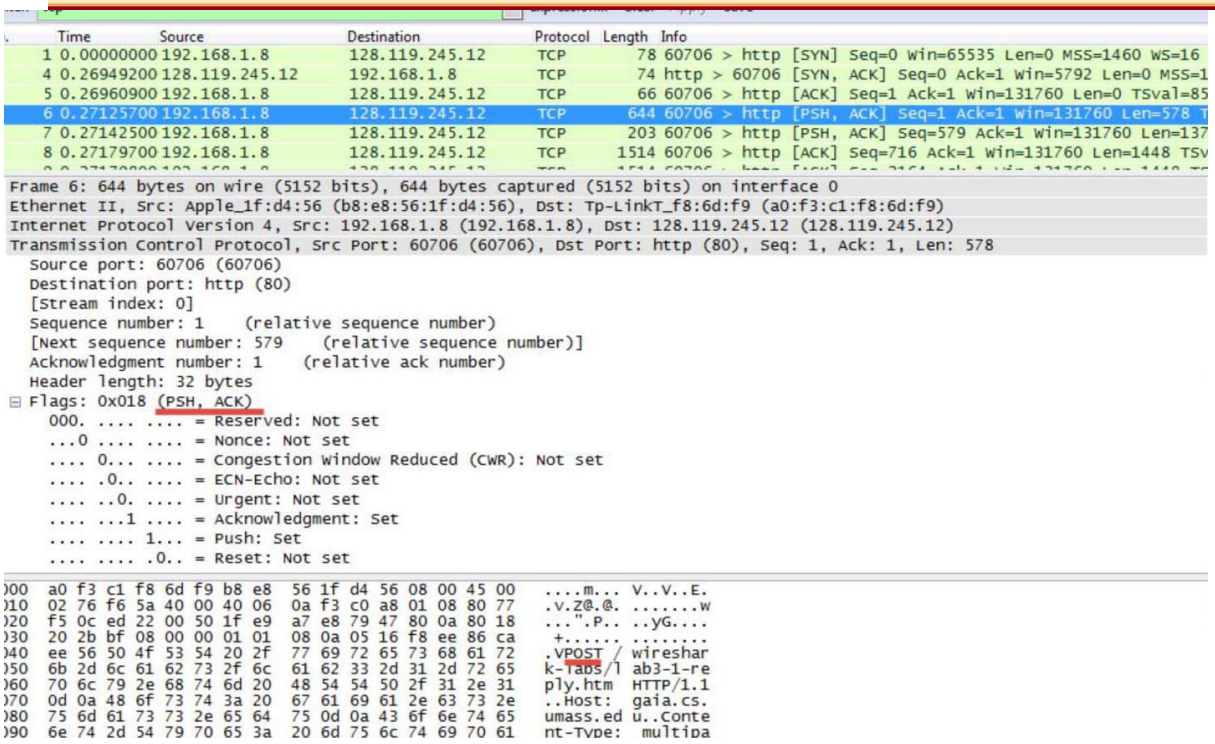
3. **What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a**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



Answer:

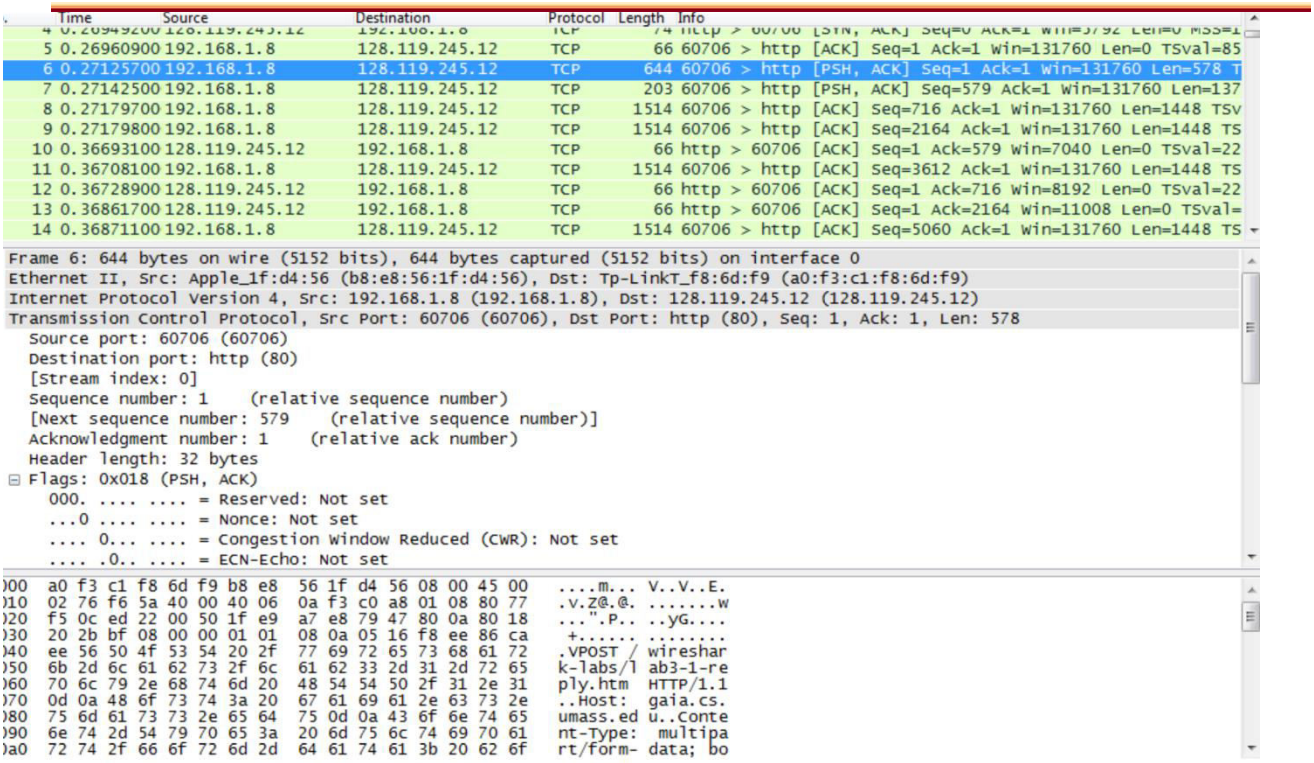
The SYNACK segment number 0 have been allocated with the SYD that was sent by the gateway gaia.cs.umass.edu geared towards opening up the communication port for data in response to the SYD that was supplied by our service provider. The SYNACK segment to the server has been made through acknowledgment field value being set to 1, this is for the purpose of notifying the server that the received SYN from the client. The server is the person who always attributing the particular value to the acknowledgment slot. The sequential number of the SYN segment in the same data is increased at the client side, that is, by one, in an effort to gain this result. When the pretaining sequence number of the Syn segment client is zero, the Ack field in the Synack socket is comparable to the value of one. It normally occurs because it is the SYN segment that is not being acknowledged.

Combined SYN flag and ack flag together set to a 1, means that the segment is a SYNACQ segment. This conclusion is met when they are both flagged into 1 bit.



As can be seen in the illustration, the HTTP POST instruction is included into segment No. 6, and the sequence number for this particular segment is really 1.

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



Answer:

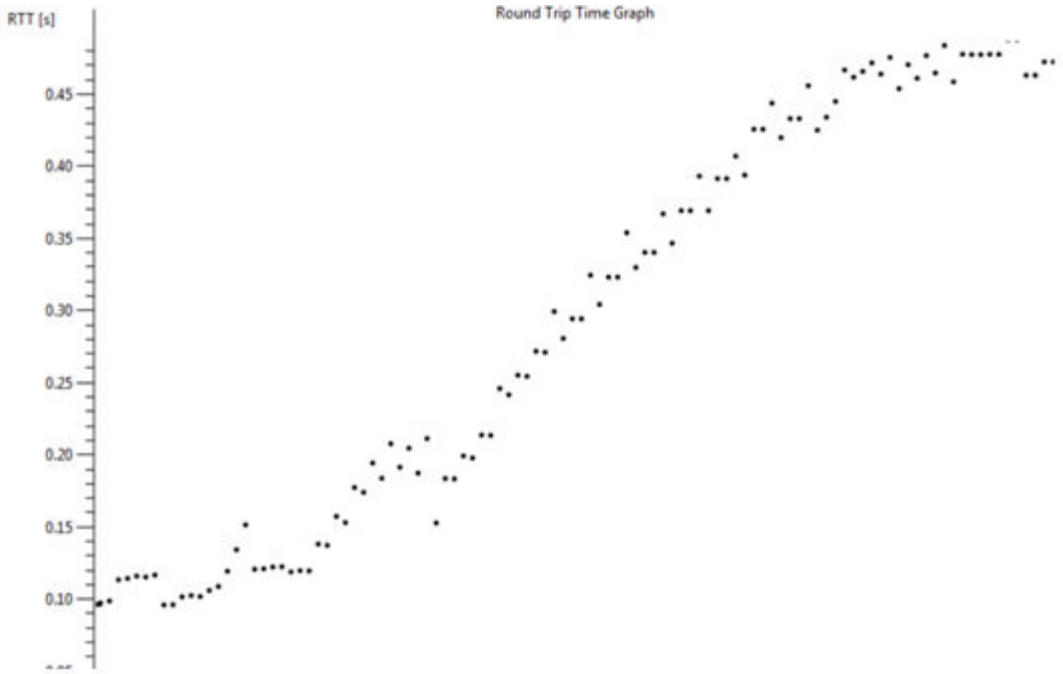
Symmetric key cryptography operates on the basis of a specific segment. One segment is equal to one sequence number. 2. The series has 579 segment. At the same time, there are 716 segments in the third transmission. The fourth series in return includes 2164 segments. The sixth part of the series is called 3612 Segment, which has 6 numbers. The index is 5060 for the number 6.

5. **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*.**

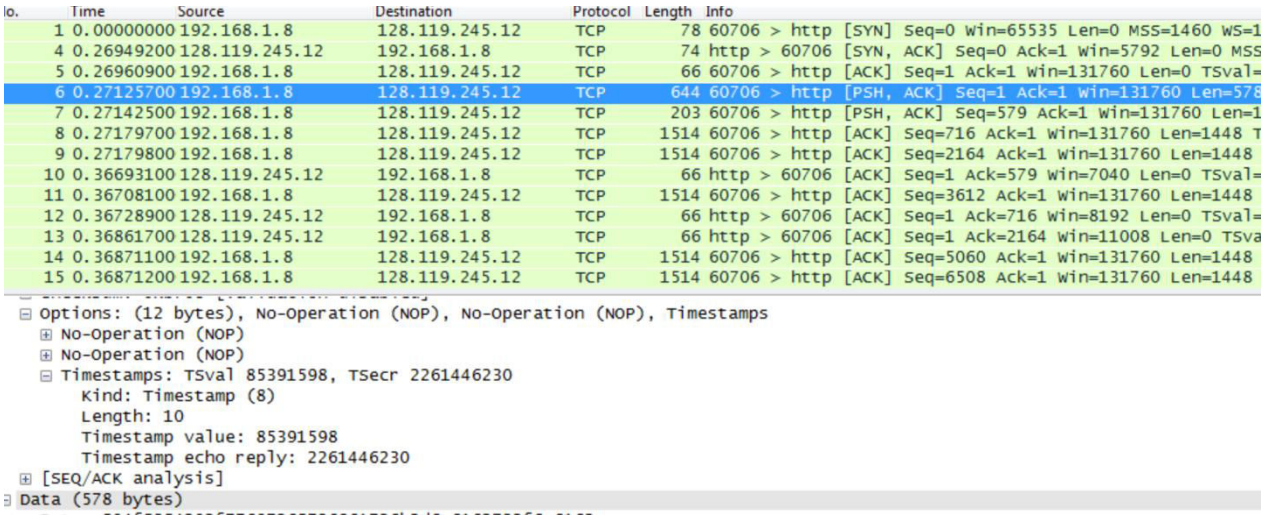


Answer:

TCP session initiates with the reception of a HA/HTP POST containing six additional parts, especially segment no. 1000, which had been preprogrammed to display the .20 second point precisely, and this was the only one that had been raised by the digit place of the segment that came before it. Every section was continued by the Acknowledgements and the RTTs (Round

Trip Time), that were break from other parts of the section due to their illustrating different information. After that iterating the following segment we reached the first round-trip time (RTT) of 0.10 seconds, using it as the first estimate for round trip time (EstimatedRTT) for the distance of 0.10 seconds. Then, the screening estimate was revised by weighted average being applied at the end of the acknowledgements of each of the ACK. These results are used to show the self-optimizing characteristic of the EstRTT, which adjusts the RTT accordingly to the anticipated change of the topology.

6. **What is the length of each of the first six TCP segments?**[2]



Answer: Size: 1448

7. **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?**

[2] The TCP segments in the tcp-ethereal-trace-1 trace file are all less that 1460 bytes. This is because the computer on which the trace was gathered has an Ethernet card that limits the length of the maximum IP packet to 1500 bytes (40 bytes of TCP/IP header data and 1460 bytes of TCP payload). This 1500 byte value is the standard maximum length allowed by Ethernet. If your trace indicates a TCP length greater than 1500 bytes, and your computer is using an Ethernet connection, then Wireshark is reporting the wrong TCP segment length; it will likely also show only one large TCP segment rather than multiple smaller segments. Your computer is indeed probably sending multiple smaller segments, as indicated by the ACKs it receives. This inconsistency in reported segment lengths is due to the interaction between the Ethernet driver and the Wireshark software. We recommend that if you have this inconsistency, that you perform this lab using the provided trace file.

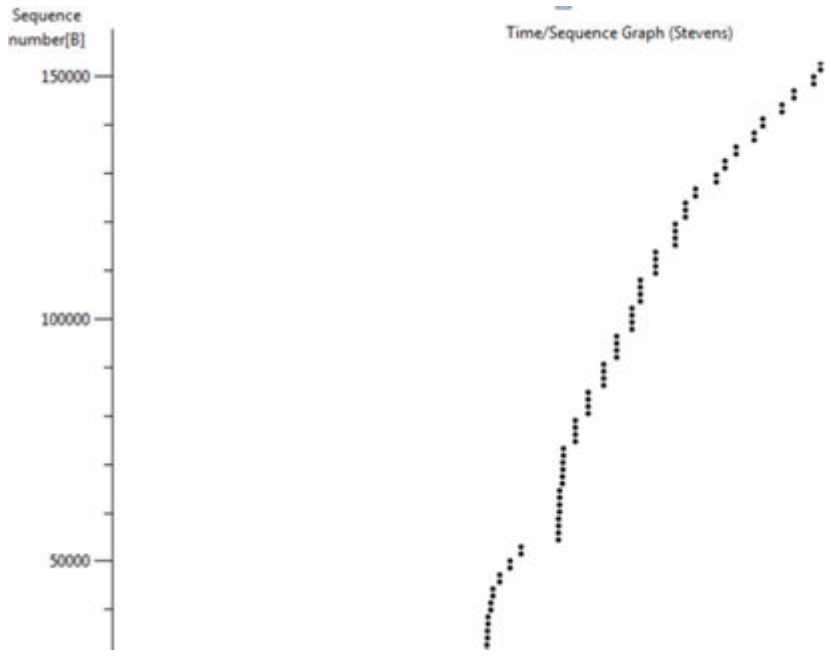
```
  4 0.26949200 128.119.245.12   192.168.1.8     TCP      74 http > 60706 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS
  5 0.26960900 192.168.1.8      128.119.245.12  TCP      66 60706 > http [ACK] Seq=1 Ack=1 Win=131760 Len=0 TSval=
  6 0.27125700 192.168.1.8      128.119.245.12  TCP     644 60706 > http [PSH, ACK] Seq=1 Ack=1 Win=131760 Len=578
  7 0.27142500 192.168.1.8      128.119.245.12  TCP     203 60706 > http [PSH, ACK] Seq=579 Ack=1 Win=131760 Len=1
  8 0.27179700 192.168.1.8      128.119.245.12  TCP    1514 60706 > http [ACK] Seq=716 Ack=1 Win=131760 Len=1448 T
  9 0.27179800 192.168.1.8      128.119.245.12  TCP    1514 60706 > http [ACK] Seq=2164 Ack=1 Win=131760 Len=1448
 10 0.36693100 128.119.245.12   192.168.1.8     TCP      66 http > 60706 [ACK] Seq=1 Ack=579 Win=7040 Len=0 TSval=
 11 0.36708100 192.168.1.8      128.119.245.12  TCP    1514 60706 > http [ACK] Seq=3612 Ack=1 Win=131760 Len=1448
 12 0.36728900 128.119.245.12   192.168.1.8     TCP      66 http > 60706 [ACK] Seq=1 Ack=716 Win=8192 Len=0 TSval=
 13 0.36861700 128.119.245.12   192.168.1.8     TCP      66 http > 60706 [ACK] Seq=1 Ack=2164 Win=11008 Len=0 TSva
 14 0.36871100 192.168.1.8      128.119.245.12  TCP    1514 60706 > http [ACK] Seq=5060 Ack=1 Win=131760 Len=1448
 15 0.36871200 192.168.1.8      128.119.245.12  TCP    1514 60706 > http [ACK] Seq=6508 Ack=1 Win=131760 Len=1448

    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  ⊞ .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
  Window size value: 5792
  [Calculated window size: 5792]
```

Answer:

As can be seen in the illustration, the HTTP POST instruction is included into segment No. 6, and the sequence number for this particular segment is really 1.

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



Answer:

The answer is no; there is no chance of retransmission occurring.

9. **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).**

Answer:

The difference in sequence numbers that are acknowledged between two consecutive acknowledgments (ACKs) is a reflection of the amount of data that the server received during the time period that transpired between these two acknowledgments. This difference is represented by the recognition of the sequence numbers. Within the parameters of this scenario, the receiver is able to identify each intermediate part. In this specific instance, the thirteenth segment is recognized, which signifies that 1430 bytes of data have been received. Other segments are also acknowledged.

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

Answer:

To calculate the throughput of the TCP connection, divide the size of the file, which is 152,138 bytes, by the length of time it takes for the file to be transmitted, which is 1.307479 seconds. This will give you the throughput of the connection. The throughput that is generated as a consequence of this is about 116,359.804 bytes per second.

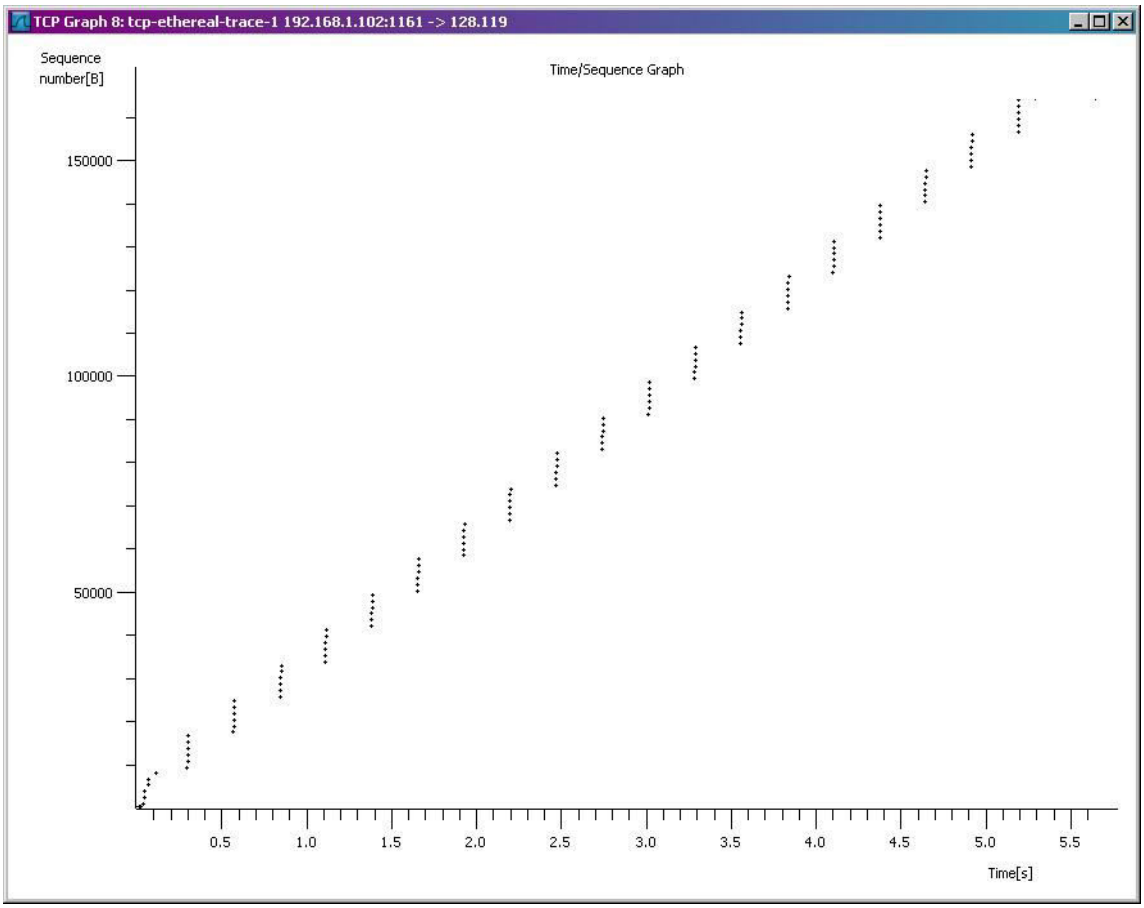
3. TCP congestion control in action

Let's now examine the amount of data sent per unit time from the client to the server. Rather than (tediously!) calculating this from the raw data in the Wireshark window,

we'll use one of Wireshark's TCP graphing utilities - *Time-Sequence-Graph(Stevens)* - to plot out data.

• Select a TCP segment in the Wireshark's "listing of captured-packets" window.

Then select the menu : *Statistics->TCP Stream Graph-> Time-SequenceGraph(Stevens)*. You should see a plot that looks similar to the following plot, which was created from the captured packets in the packet trace *tcp-etherealtrace-1*.



Each dot in this diagram represents a TCP segment that was sent, and the plot shows the sequence number of the segment in relation to the time at which it was transmitted. It is important to take

note that a sequence of packets that were transmitted back-to-back by the sender is represented by a set of dots that are stacked on top of each other.

Answer the following questions for the TCP segments the packet trace *tcpethereal-trace-1*.

11. 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.

Answer:

Slow Start: The Transmission Control Protocol (TCP) starts by sending a small number of segments, often one or two, and then gradually increases the number of segments it sends until it detects congestion before sending additional segments. This process is known as a slow start. This first phase is known as the slow start, and it is a precursor stage. According to the explanation you provided, the slow start begins roughly 0.27 seconds after the connection is formed and continues until approximately 0.35 seconds after that. An investigation of the network is being carried out by the sender during this time period in order to determine the appropriate sending rate.

Congestion Avoidance: The Transmission Control Protocol (TCP) automatically enters the mode known as congestion avoidance whenever it detects congestion in the network. This suggests that it reduces the rate at which it transmits data in order to reduce the amount of congestion that it experiences. According to your description, the congestion avoidance method takes over at around 0.7 seconds into the game. When compared to slow start, the Transmission Control

Protocol (TCP) raises the congestion window in a more progressive manner during congestion avoidance.