ICSI 416/516 Homework 3 – Transport
20 points

Due date: Wednesday 3/2 at 11:59PM as a single PDF file via Blackboard

All parts of the assignment are to be completed independently. Submission of the same text by multiple students will be considered cheating. Students caught cheating will receive 0 points for the assignment and will be reported.

Part 1 [14 points]: Answer the theoretical questions on the following two pages.
Part 2 [6 points]: Complete the Wireshark Lab and answer the questions in the lab. You can download the Wireshark trace for this assignment from
http://www.cs.albany.edu/~mariya/courses/csi416516S16/hw/tcp-wireshark-trace-1

[Theoretical questions]

1. [1 point] Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?

2. [1 point] Consider the basic operations of TCP. Is TCP a Go-Back-N or a Selective Repeat protocol? Provide a brief justification.

3. [2 points] The pipelined protocol in Figure 1 has a window size of N packets and runs over a physical link with bandwidth R=1Gbit/s. The link has a propagation delay of 10ms and each of the packets sent over this link is of size L=5KBytes. What is the achieved throughput of the above pipelined protocol if N=3, 6 and 12? Write the formula you are using to calculate throughput (using the same notation as in the assignment) along with your result. Comment on the trends you observe with increasing N.

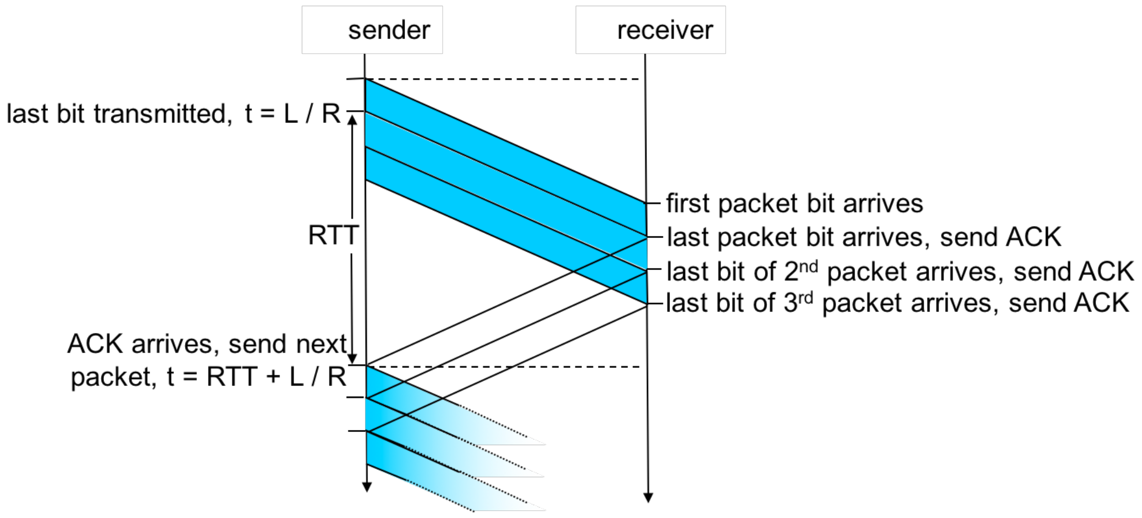

sender
receiver
last bit transmitted, t = L / R
RTT
ACK arrives, send next packet, t = RTT + L / R
first packet bit arrives
last packet bit arrives, send ACK
last bit of 2nd packet arrives, send ACK
last bit of 3rd packet arrives, send ACK

Figure1. Illustration of the pipelined protocol from theoretical problem 3.

4. [1 point] What are the similarities and differences between congestion control and flow control?

5. [2 point] We developed our realistic reliable data transfer (rdt3.0) protocol by assuming increasingly complicating circumstances (e.g. channel with error and loss) that required additional parameters to handle these circumstances. Why did we need to introduce sequence numbers? How about timers?

6. [2 points] Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol rdt 3.0, assuming that packets can be lost? Explain.

7. [5 points] In this question we will apply our knowledge of the operations of TCP Reno in order to study its behavior through a real-world measurement. Figure 2 shows the congestion window of the protocol over multiple transmission rounds. Study the graph and answer the following questions:

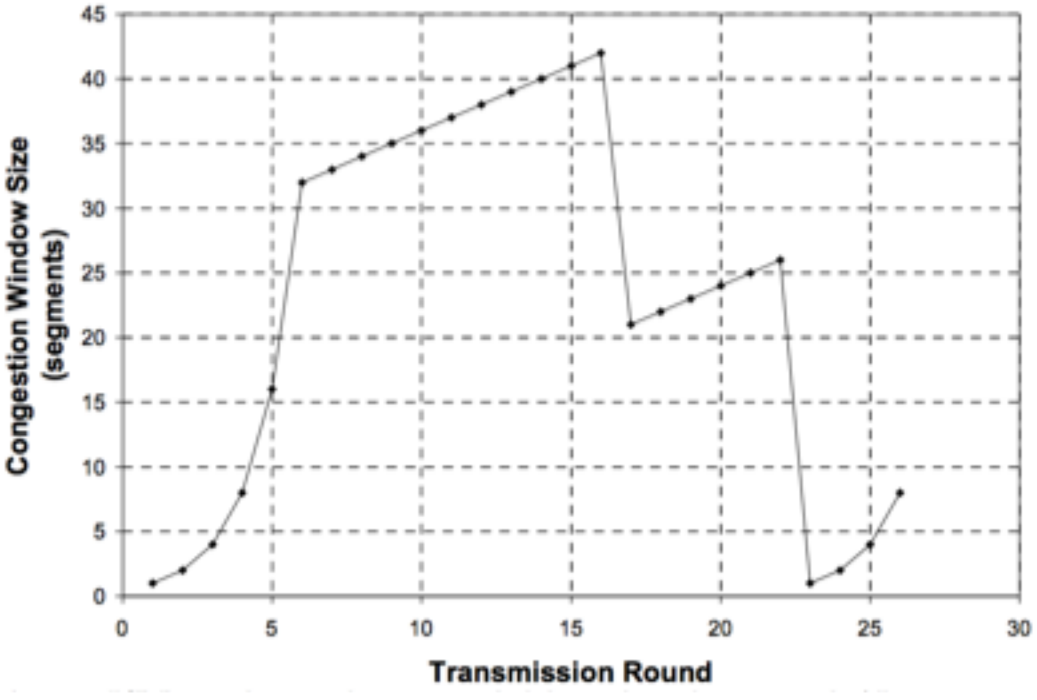


Figure 2. Graph of TCP Reno's congestion window over multiple consecutive transmission rounds.

a. In which round intervals is *slow start* operating?

b. Which *congestion avoidance* mechanism can be observed here?

c. In which intervals is *congestion avoidance* operating?

d. What caused the `cwnd` decrease in round 16: a triple duplicate ACK or a timeout?

e. What is the value of `ssthresh` in round 1?

f. What is the value of `ssthresh` in round 18?

g. What is the value of `ssthresh` in round 24?

h. In which transmission round is the 70th segment sent?

i. Imagine that the sender received a triple duplicate ACK after the 26th round. What will be the values of `cwnd` and `ssthresh` following this event?

Wireshark Lab: TCP

In this lab, we'll investigate the behavior of TCP 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 a 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 the client computer and the server.

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

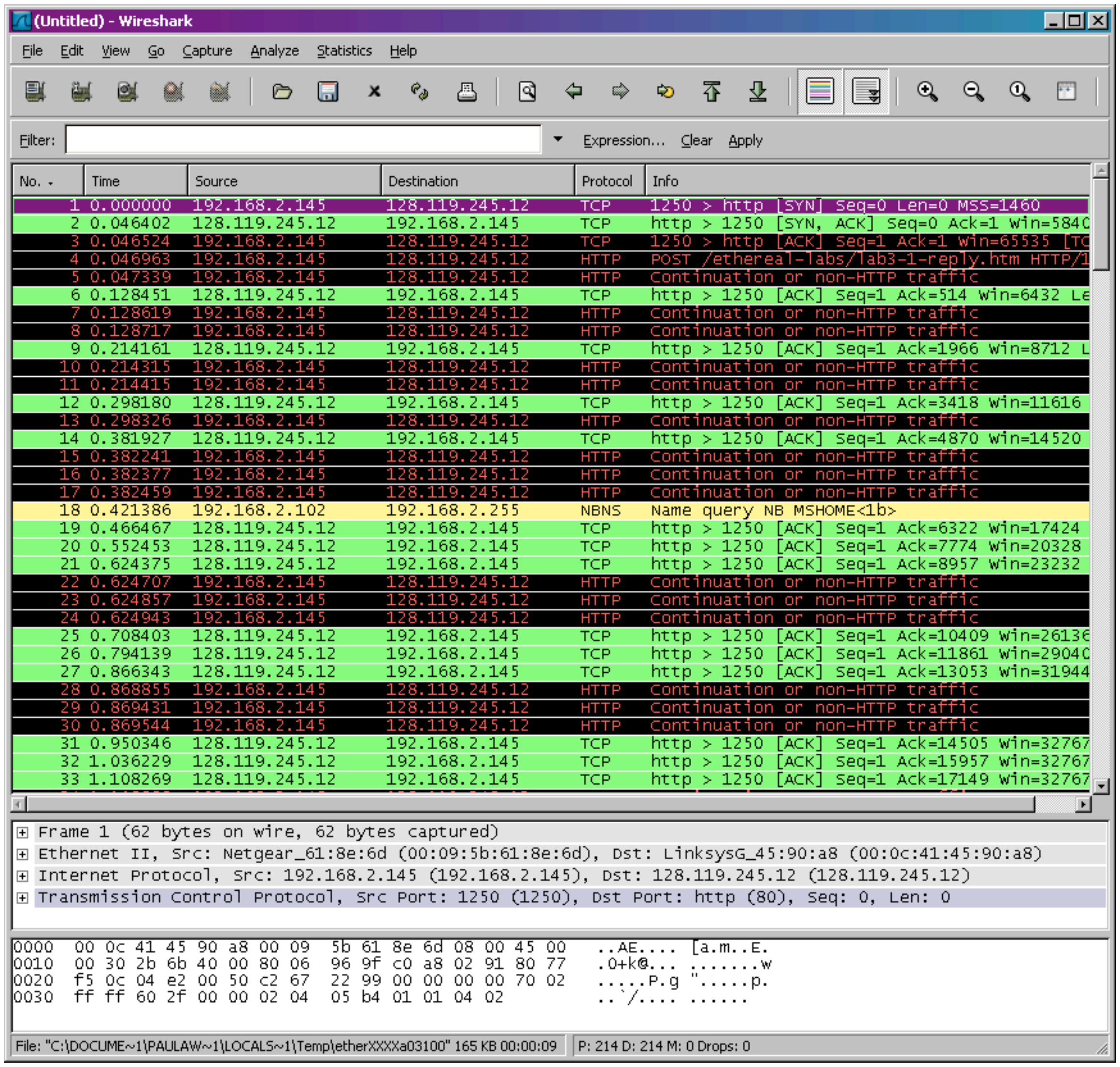
1. Capturing a bulk TCP transfer from the client computer to a remote server

To allow you to investigate TCP's behavior, Wireshark was used to obtain a packet trace of the TCP transfer of a file from a client computer to a remote server. The trace was created by accessing a Web page that will allowed the user to enter the name of a file stored on the client computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method (see section 2.2.3 in the text). The POST method was used rather than the GET method because we'd like to transfer a large amount of data *from* the client computer to another computer. During the transfer, Wireshark was running to obtain the trace of the TCP segments sent and received from the client computer.

The Wireshark packet trace was captured using the following steps:

- The user visited the URL http://gaia.cs.umass.edu/Wireshark-labs/TCP-Wireshark-file1.html
- The user entered the name of the file containing the text of *Alice in Wonderland.*
- Before pressing the Upload button, the packet capture with Wireshark was started.
- Then the user pressed the "*Upload alice.txt file*" button to upload the file to the gaia.cs.umass.edu server. Once the file had been uploaded, a short congratulations message was be displayed in the browser window.
- Wireshark packet capture was then stopped.

If you load the tcp_wireshark_trace_1 file into Wireshark, your Wireshark window should look like the window shown below.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. . Time     Source          Destination     Protocol Info
    1 0.000000 192.168.2.145   128.119.245.12  TCP      1250 > http [SYN] Seq=0 Len=0 MSS=1460
    2 0.046402 128.119.245.12  192.168.2.145   TCP      http > 1250 [SYN, ACK] Seq=0 Ack=1 Win=5840
    3 0.046524 192.168.2.145   128.119.245.12  TCP      1250 > http [ACK] Seq=1 Ack=1 Win=65535 [TC
    4 0.046963 192.168.2.145   128.119.245.12  HTTP     POST /ethereal-labs/lab3-1-reply.htm HTTP/1
    5 0.047339 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
    6 0.128451 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=514 Win=6432 Le
    7 0.128619 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
    8 0.128717 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
    9 0.214161 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=1966 Win=8712 L
   10 0.214315 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   11 0.214415 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   12 0.298180 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=3418 Win=11616
   13 0.298326 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   14 0.381927 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=4870 Win=14520
   15 0.382241 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   16 0.382377 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   17 0.382459 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   18 0.421386 192.168.2.102   192.168.2.255   NBNS     Name query NB MSHOME<1b>
   19 0.466467 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=6322 Win=17424
   20 0.552453 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=7774 Win=20328
   21 0.624375 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=8957 Win=23232
   22 0.624707 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   23 0.624857 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   24 0.624943 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   25 0.708403 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=10409 Win=26136
   26 0.794139 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=11861 Win=29040
   27 0.866343 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=13053 Win=31944
   28 0.868855 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   29 0.869431 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   30 0.869544 192.168.2.145   128.119.245.12  HTTP     Continuation or non-HTTP traffic
   31 0.950346 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=14505 Win=32767
   32 1.036229 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=15957 Win=32767
   33 1.108269 128.119.245.12  192.168.2.145   TCP      http > 1250 [ACK] Seq=1 Ack=17149 Win=32767
```

```
⊞ Frame 1 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: Netgear_61:8e:6d (00:09:5b:61:8e:6d), Dst: LinksysG_45:90:a8 (00:0c:41:45:90:a8)
⊞ Internet Protocol, Src: 192.168.2.145 (192.168.2.145), Dst: 128.119.245.12 (128.119.245.12)
⊞ Transmission Control Protocol, Src Port: 1250 (1250), Dst Port: http (80), Seq: 0, Len: 0
```

```
0000  00 0c 41 45 90 a8 00 09  5b 61 8e 6d 08 00 45 00   ..AE.... [a.m..E.
0010  00 30 2b 6b 40 00 80 06  96 9f c0 a8 02 91 80 77   .0+k@... .......w
0020  f5 0c 04 e2 00 50 c2 67  22 99 00 00 00 00 70 02   .....P.g ".....p.
0030  ff ff 60 2f 00 00 02 04  05 b4 01 01 04 02         ..`/.... ......
```

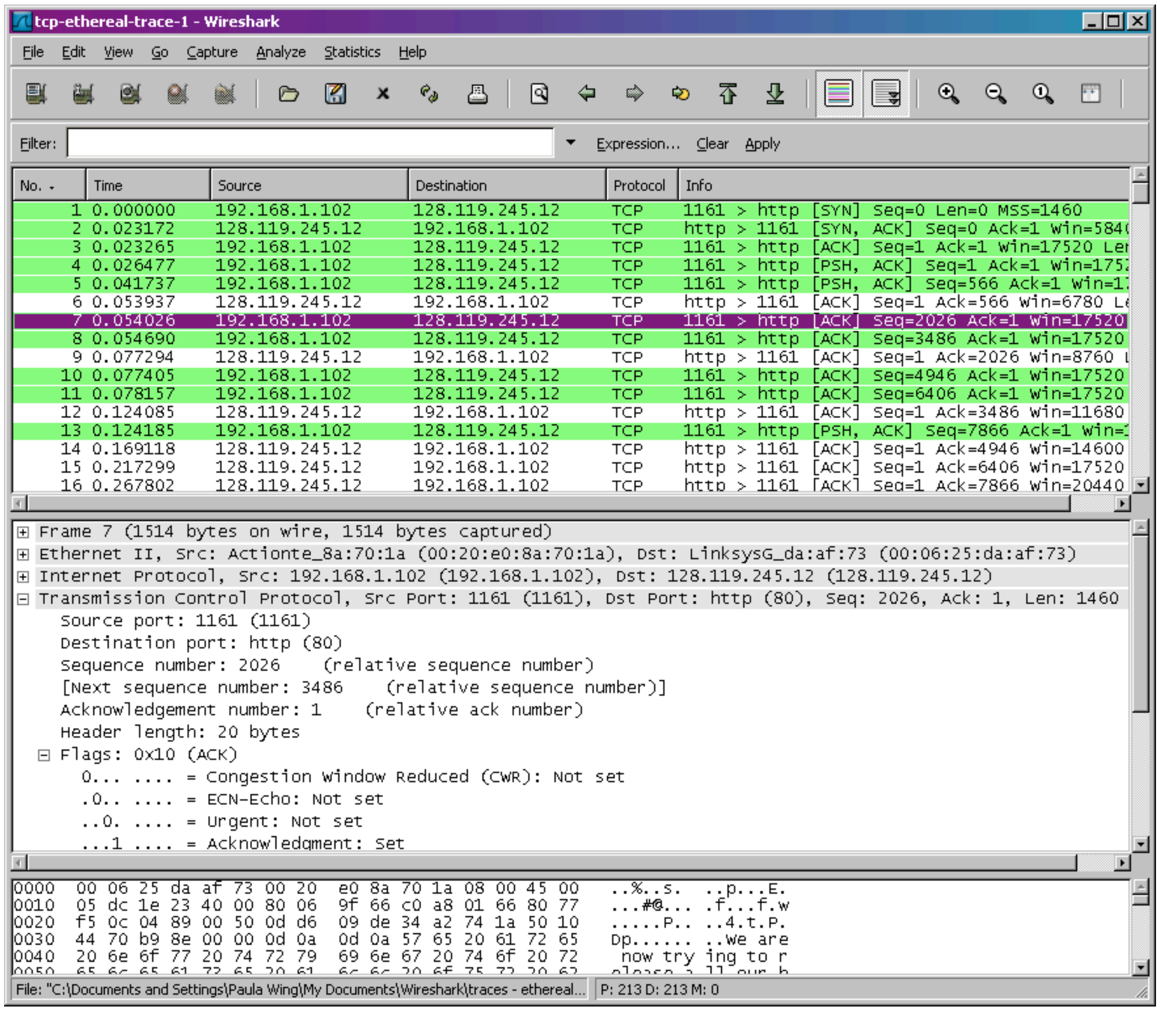
File: "C:\DOCUME~1\PAULAW~1\LOCALS~1\Temp\etherXXXXa03100" 165 KB 00:00:09 | P: 214 D: 214 M: 0 Drops: 0

2. 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" 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 the client 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 and a series of "HTTP Continuation" messages being sent from the client computer to gaia.cs.umass.edu. Note that there 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. You should also see TCP ACK segments being returned from gaia.cs.umass.edu to the client computer.

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:


tcp-ethereal-trace-1 - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. .  Time      Source          Destination     Protocol Info
    1 0.000000  192.168.1.102   128.119.245.12  TCP      1161 > http [SYN] Seq=0 Len=0 MSS=1460
    2 0.023172  128.119.245.12  192.168.1.102   TCP      http > 1161 [SYN, ACK] Seq=0 Ack=1 Win=584
    3 0.023265  192.168.1.102   128.119.245.12  TCP      1161 > http [ACK] Seq=1 Ack=1 Win=17520 Le
    4 0.026477  192.168.1.102   128.119.245.12  TCP      1161 > http [PSH, ACK] Seq=1 Ack=1 Win=175
    5 0.041737  192.168.1.102   128.119.245.12  TCP      1161 > http [PSH, ACK] Seq=566 Ack=1 Win=1
    6 0.053937  128.119.245.12  192.168.1.102   TCP      http > 1161 [ACK] Seq=1 Ack=566 Win=6780 L
    7 0.054026  192.168.1.102   128.119.245.12  TCP      1161 > http [ACK] Seq=2026 Ack=1 Win=17520
    8 0.054690  192.168.1.102   128.119.245.12  TCP      1161 > http [ACK] Seq=3486 Ack=1 Win=17520
    9 0.077294  128.119.245.12  192.168.1.102   TCP      http > 1161 [ACK] Seq=1 Ack=2026 Win=8760 L
   10 0.077405  192.168.1.102   128.119.245.12  TCP      1161 > http [ACK] Seq=4946 Ack=1 Win=17520
   11 0.078157  192.168.1.102   128.119.245.12  TCP      1161 > http [ACK] Seq=6406 Ack=1 Win=17520
   12 0.124085  128.119.245.12  192.168.1.102   TCP      http > 1161 [ACK] Seq=1 Ack=3486 Win=11680
   13 0.124185  192.168.1.102   128.119.245.12  TCP      1161 > http [PSH, ACK] Seq=7866 Ack=1 Win=
   14 0.169118  128.119.245.12  192.168.1.102   TCP      http > 1161 [ACK] Seq=1 Ack=4946 Win=14600
   15 0.217299  128.119.245.12  192.168.1.102   TCP      http > 1161 [ACK] Seq=1 Ack=6406 Win=17520
   16 0.267802  128.119.245.12  192.168.1.102   TCP      http > 1161 [ACK] Seq=1 Ack=7866 Win=20440
```

```
⊞ Frame 7 (1514 bytes on wire, 1514 bytes captured)
⊞ Ethernet II, Src: Actionte_8a:70:1a (00:20:e0:8a:70:1a), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
⊞ Internet Protocol, Src: 192.168.1.102 (192.168.1.102), Dst: 128.119.245.12 (128.119.245.12)
⊟ Transmission Control Protocol, Src Port: 1161 (1161), Dst Port: http (80), Seq: 2026, Ack: 1, Len: 1460
    Source port: 1161 (1161)
    Destination port: http (80)
    Sequence number: 2026    (relative sequence number)
    [Next sequence number: 3486    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
```

```
0000  00 06 25 da af 73 00 20  e0 8a 70 1a 08 00 45 00   ..%..s.  ..p...E.
0010  05 dc 1e 23 40 00 80 06  9f 66 c0 a8 01 66 80 77   ...#@... .f...f.w
0020  f5 0c 04 89 00 50 0d d6  09 de 34 a2 74 1a 50 10   .....P.. ..4.t.P.
0030  44 70 b9 8e 00 00 0d 0a  0d 0a 57 65 20 61 72 65   Dp...... ..We are
0040  20 6e 6f 77 20 74 72 79  69 6e 67 20 74 6f 20 72    now try ing to r
0050  65 6c 65 61 73 65 20 61  6c 6c 20 6f 75 72 20 62   please a ll our b
```

File: "C:\Documents and Settings\Paula Wing\My Documents\Wireshark\traces - ethereal... P: 213 D: 213 M: 0

This is what we're looking for - a series of TCP segments sent between the client computer and gaia.cs.umass.edu. We will use the packet trace to study TCP behavior in the rest of this lab.

3. TCP Basics

Answer the following questions for the TCP segments:

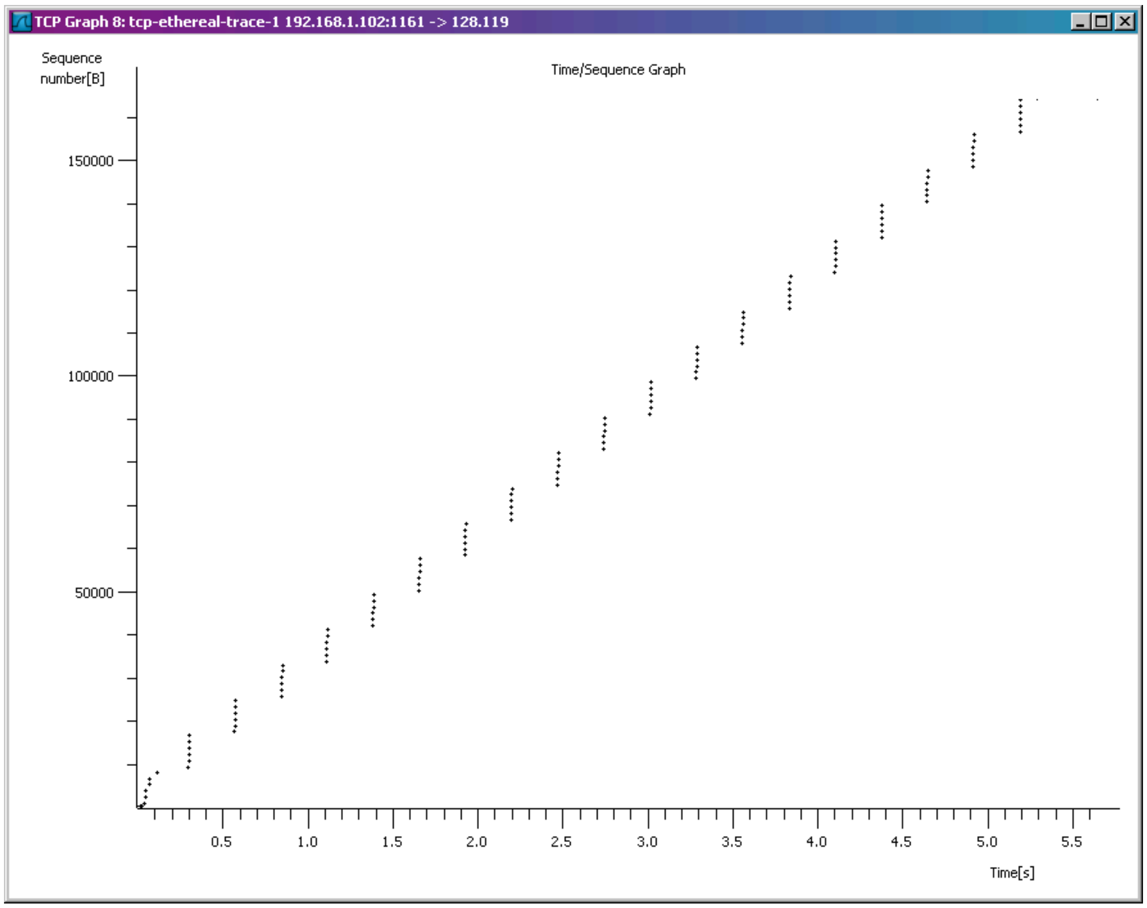
1. What is the IP address and TCP port number used by the client computer (source) to transfer the file to gaia.cs.umass.edu? What is the IP address and port number used by gaia.cs.umass.edu to receive the file?
2. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?
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?
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.
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 page 249 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 249 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.*

4. 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-Sequence-Graph(Stevens)*. To get the most interesting plot, select a packet with the source IP of 192.168.1.102, and select a packet towards the middle to end of the trace. You should see a plot that looks similar to the following plot:



Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

Answer the following question:

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