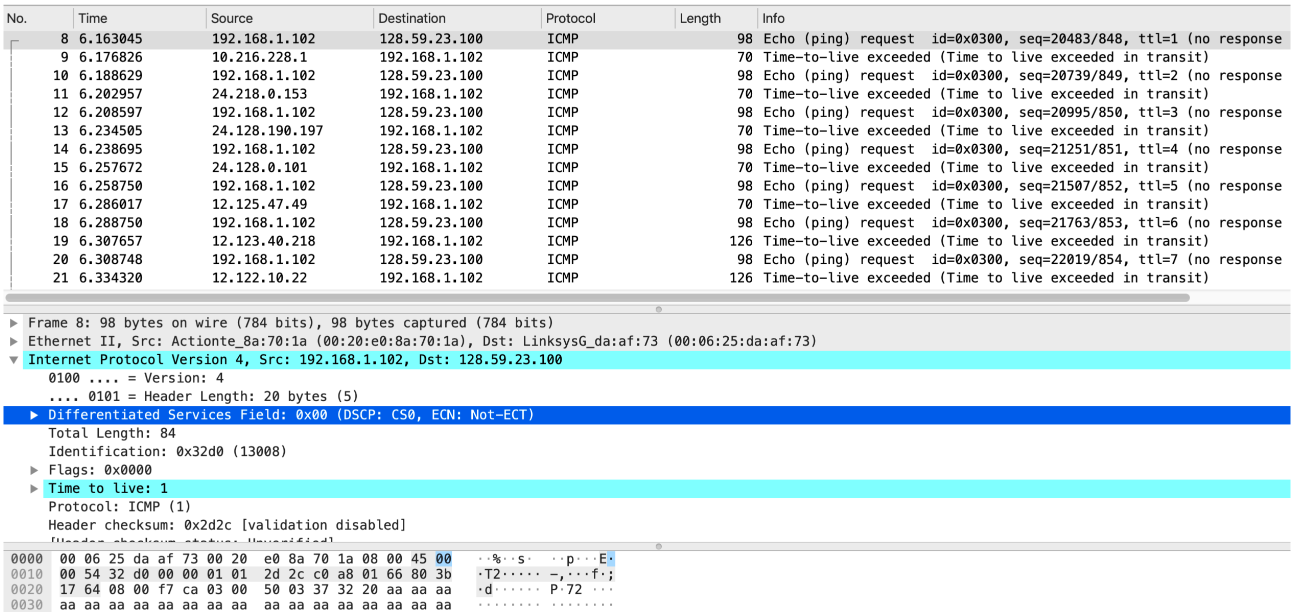
DE VO (1080326)
HOMEWORK 3 + LAB 4

1. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window.



What is the IP address of your computer?
192.168.1.102

2. Within the IP packet header, what is the value in the upper layer protocol field?
Protocol: ICMP (1)

3. How many bytes are in the IP header? How many bytes are in the payload *of the IP datagram*? Explain how you determined the number of payload bytes.

   Header Length: 20 bytes (5)
   Payload length: 64 bytes (total Length: 84 – 20) from screenshot

4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.
IP datagram has not been fragmented because the fragments bit is still 0

5. Which fields in the IP datagram *always* change from one datagram to the next within this series of ICMP messages sent by your computer?
The header checksum
Identification number
Time to live also change sometime

6. Which fields stay constant? Which of the fields *must* stay constant? Which fields must change? Why?

Constant:

- version (IPv4 always used)
- header length (doesn't change becayse using IPv4)
- source IP
- destination IP
- differentiated services (same protocol)
- upper layer protocol (same protocol)

Must change:

- time to live (this is how traceroute send signal to different router)
- identification (unique for each datagram)

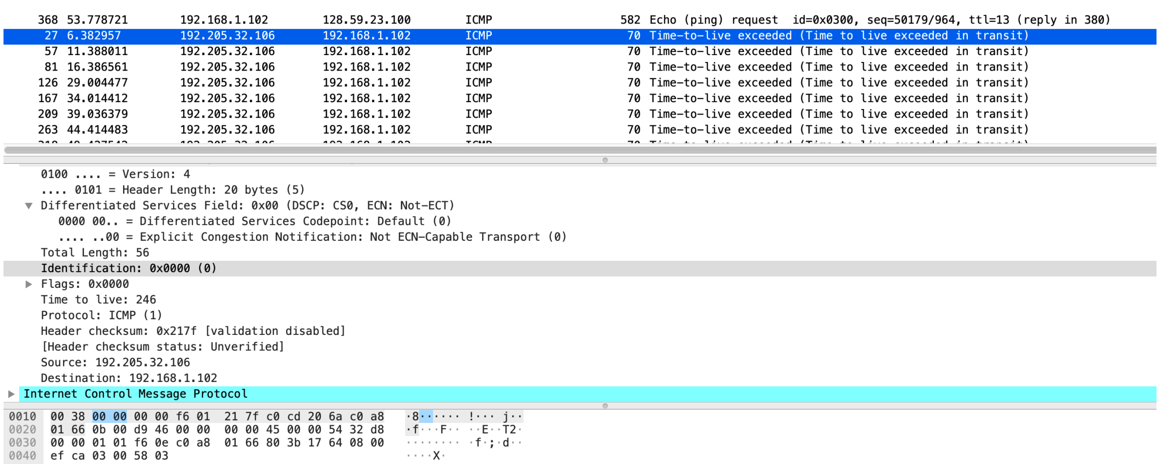
7. Describe the pattern you see in the values in the Identification field of the IP datagram

- They are incrementing with each datagram.

Next (with the packets still sorted by source address) find the series of ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router.

8. What is the value in the Identification field and the TTL field?

   Nearest router: 192.205.32.106
   Identification: 0
   TTL filed: 246



9. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

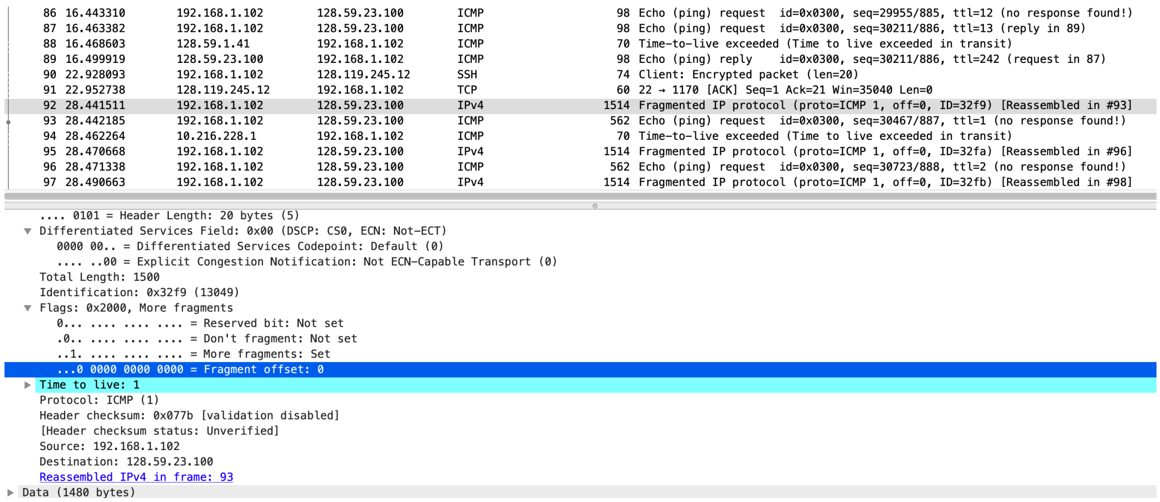
- Identification value changes because it needs to be unique
- TTL value stay constant because its same first hope router

Fragmentation

Sort the packet listing according to time again by clicking on the *Time* column.

10. Find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* in *pingplotter* to be 2000. Has that message been fragmented across more than one IP datagram?

- Yes, the message has been fragmented across more than one IP

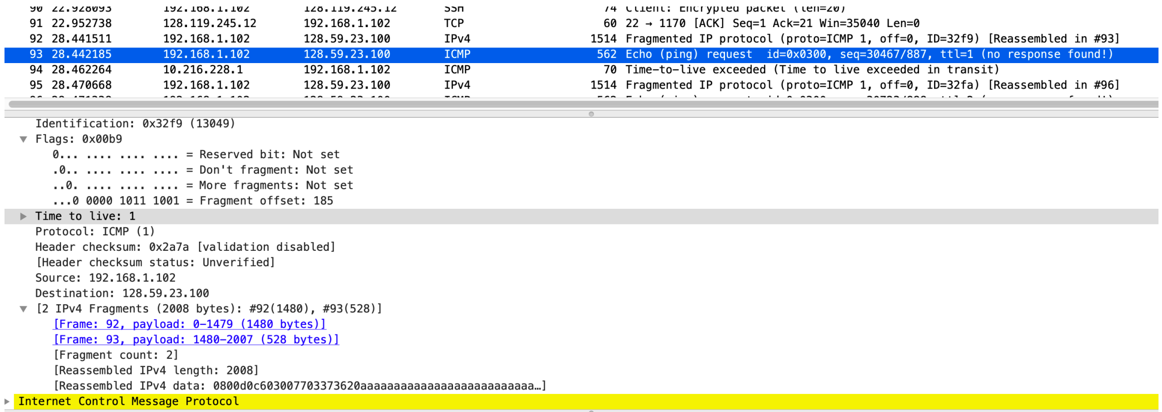


11. Print out the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?

- The more fragments flag bit is set indicates the datagram has been fragmented.
- Since the fragment offset is 0, it is the first fragment and not latter.
- This first datagram has a total length of 1500 bytes

12. Print out the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?

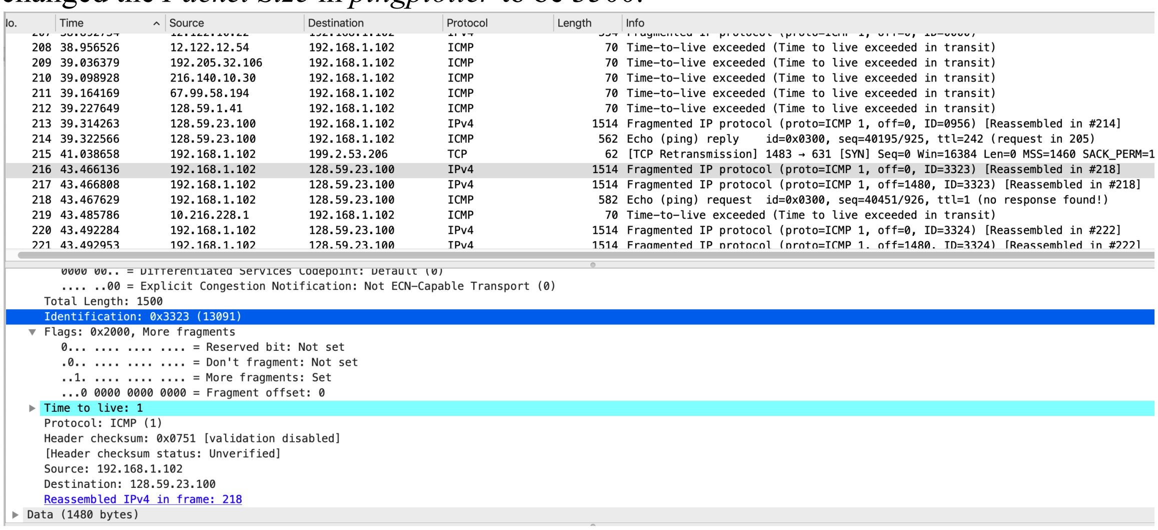
- It is not the first fragment because the fragment offset is 1480
- There is not any more fragments because the more fragments flag is not set.



13. What fields change in the IP header between the first and second fragment?

- The IP header fields that changed between the fragments are: total length, more fragment flag, fragment offset, and checksum.

Now find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* in *pingplotter* to be 3500.



14. How many fragments were created from the original datagram?

There are 3 fragments were created for 3500 packet size

15. What fields change in the IP header among the fragments?

- Among all 3 packets: fragment offset, and checksum.
- Between the first two packets and the last packet: total length, and also in the more fragment flag

DE VO (1080326) – Homework 3
Summer 2019

1. (15 pts) Wireshark Lab #4 – IP
   Complete the fourth Wireshark Lab, #4, from the class drive. You do not need to supply your answers in a separate document; you can copy/paste them here if you like. It might be pages and pages of material – that's fine.

2. (20 pts, 4x5) From the HTTP Delay Estimation animations at https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/http-delay-estimation/index.html, play around and answer the following questions:
   a. Which is generally faster with a large number of objects, non-persistent or persistent connections?
   Persistent connections

   b. Which is generally faster with a large number of objects, connections with or without pipelining?

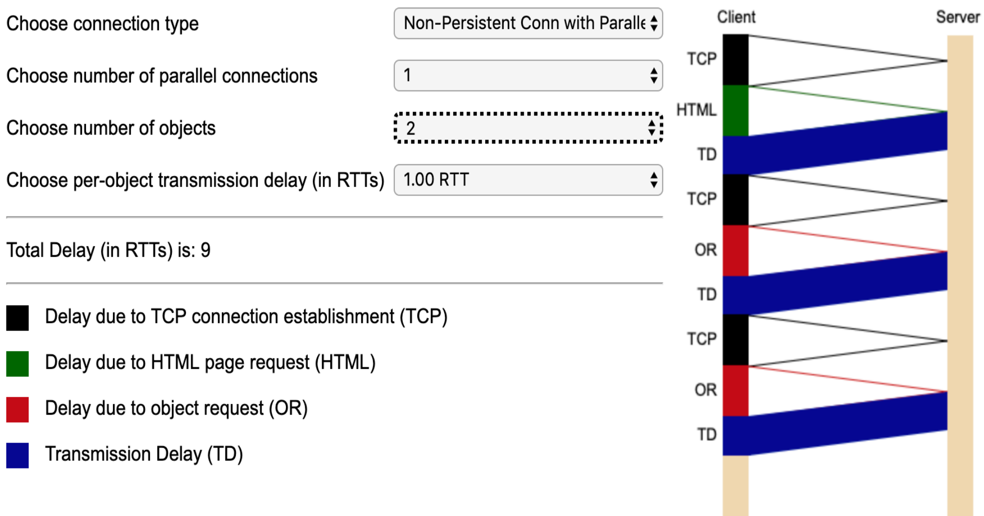
   Persistent connection with pipelining is faster

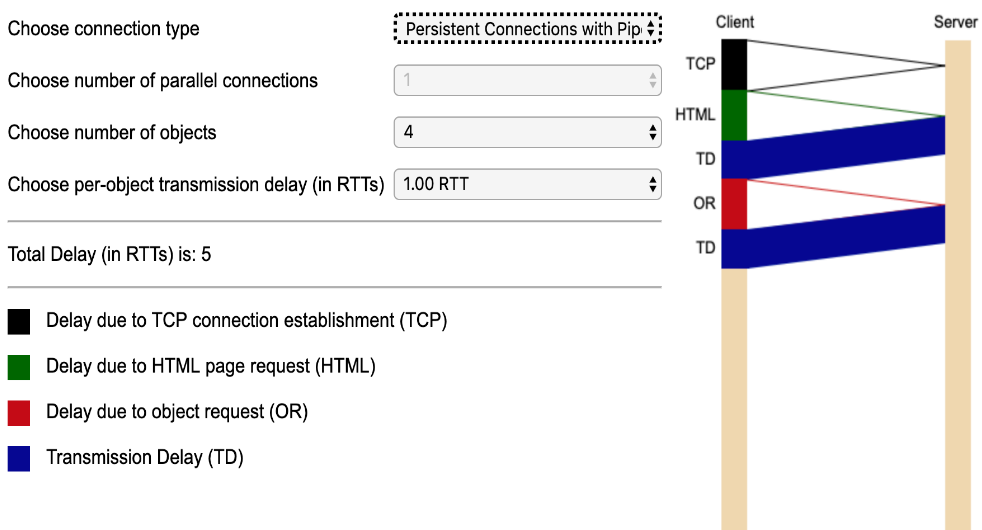
   c. Why will Non-Persistent Connections with Parallel Connections generally win or lose for the fastest response time if up against Persistent Connections with Pipelining? Would the # of parallel connections be to blame? The number of objects? Or the per-object transmission delay? Choose one.

   Blame the number of parallel connections because a parallel connection spends more time establish TCP connection establishment if it's a single parallel

   d. How many more or less RTTs is required for a:
   * non-persistent connection with 4 parallel connections sending 8 objects with a 1 RTT transmission delay per object, than for a
   * persistent connection with pipelining sending the same 8 objects with the same transmission delay?

   Assume you can have no more than 4 parallel connections or pipelined objects per request. Your answer should be a positive integer if non-persistent requires more RTTs, a negative integer if it requires fewer.



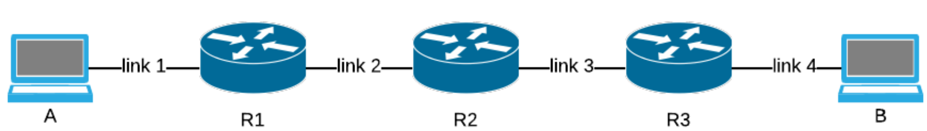


4 non-persistent parallel + 8 obj is same as 1 parallel and 2 obj

⇨ Difference: 9 – 5 = 4

None persistent is slower by 2 RTT

3. (15 pts) P10. Consider a packet of length L that begins at end system A and travels over four links to a destination end system B. These four links are connected by three packet switches (routers). Let d_i, s_i, and R_i denote the length, propagation speed, and the transmission rate of link i, for i=1,2,3,4. The packet switch delays each packet by d_{proc}.



a. Assuming no queuing delays, in terms of d_i, s_i, R_i, (i=1,2,3,4), and L, what is the total end-to-end delay for the packet? Measure from the moment you begin transmitting the first bit of the first packet at A until the first bit of the last packet arrives at the last system B. It might be easier to group the d's together and then the s's and so on when you write out the answer. Order doesn't matter. Just remember that d, s, and R are different for each link and router.

End to end = N*(d_proc + d_trans + d_prop)
= N*(d_proc + L/Ri + d/si)

b. Suppose now the packet is 3125 bytes, the propagation speed on all 4 links is $2.5 \cdot 10^8$ m/s, the transmission rates of all 4 links are 2.5 Mbps, each packet switch (router) introduces a processing delay of 5 msec, and the lengths of the links are 2,500 km, 3,500 km, 3,000 km, and 4,000 km. For these values, what is the end-to-end delay?

i=1 => 4*(0.005 + (3125*8 / 2,500,000) + (2,500,000/2.5E8)) = 0.1 s

i=2 => 4*(0.005 + (3125*8 / 2,500,000) + (3,500,000/2.5E8)) = 0.116 s

i=3 => 4*(0.005 + (3125*8 / 2,500,000) + (3,000,000/2.5E8)) = 0.108 secs

i=4 => 4*(0.005 + (3125*8 / 2,500,000) + (4,000,000/2.5E8)) = 0.124 secs

Total_delay =D1 +D2 +D3 + D4 = 0.448 s

c. Now suppose $R1=R2=R3=R$ and $d_{proc}=0$. Further suppose the packet switch does not store-and-forward packets but instead immediately transmits each bit it receives before waiting for the entire packet to arrive. The processing to figure that out which would usually happen after enough of the packet has arrived to see the addresses inside of it is not required (this is how circuit-switched networks behave). So what is the end-to-end delay?

i=1 => (4*3125*8 / 2,500,000) + (2,500,000/2.5E8) = a s

i=2 => (4*3125*8 / 2,500,000) + (3,500,000/2.5E8) = b s

i=3 => (4*3125*8 / 2,500,000) + (3,000,000/2.5E8) = c secs

i=4 => (4*3125*8 / 2,500,000) + (4,000,000/2.5E8) = d secs

end to end = a + b+ c + d = 0.212 s

4. (20 pts, 4x5) From the DNS and HTTP delays interactive exercise from the text at http://gaia.cs.umass.edu/kurose_ross/interactive/DNS_HTTP_delay.php, solve the following problem and answer the questions below. The web site will step you through similar randomly-generated problems for practice. You will see this problem on the mid-term exam.

   Before doing this question, you might want to review sections 2.2.1 and 2.2.2 on HTTP (in particular the text surrounding Figure 2.7) and the operation of the DNS (in particular the text surrounding Figure 2.19).

   Suppose within your Web browser you click on a link to obtain a Web page.
   The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address.

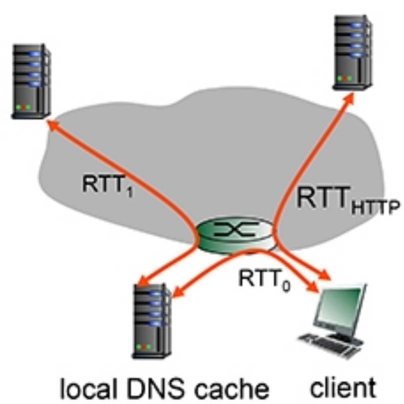
   Suppose that two DNS servers are visited before your host receives the IP address from DNS. The first DNS server visited is the local DNS cache, with an RTT delay of

   $RTT_0 = 4$ msecs.

   The second DNS server contacted has an

   RTT of 44 msecs.

   Initially, let's suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Suppose the RTT between the local host and the Web server containing the object is
   $RTT_{HTTP} = 67$ msecs.



   a. Assuming zero transmission time for the HTML object, how much time elapses from when the client clicks on the link until the client receives the object?

      Time = RTT0 + RTT1 + 2* RTT_{http} = 4 + 44 + 2*67 = 182 mS

   b. Now suppose the HTML object references **9** very small objects on the same web server. Neglecting transmission times, how much time elapses from when the client clicks on the link until the base object and all **9** additional objects are received from web server at the client, assuming non-persistent HTTP and no parallel TCP connections?

      RTT_0+…+ RTT_n+2*RTT_{http}+9*2*RTT_{http} = 4+44+134+ (9*2*67)= 1388 mS

c. Repeat the previous step but assume that the client is configured to support a maximum of **5** parallel TCP connections, with **non-persistent** HTTP.

RTT_0+RTT_1+2*RTT_{http}+ 2*RTT_{http}+ 2*RTT_{http}= 4 + 44 + 134. + 134 + 134 = 450 mS

d. Repeat again but assume that the client is configured to support a maximum of **5** parallel TCP connections, with **persistent** HTTP.

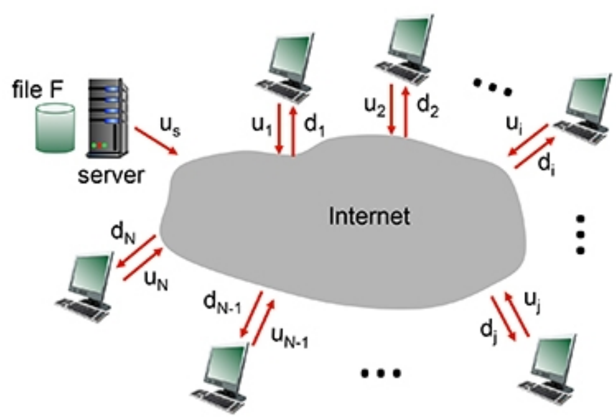
RTT_0 + RTT_1 + 2*RTT_{HTTP} + RTT_{HTTP} + RTT_{HTTP}= 4 + 44 + 134 + 67+67 = 316 mS

e. What do you notice about the overall delays (taking into account both DNS and HTTP delays) that you computed in the three cases above?

b -> largest dealy
c -> middle
d -> smallest delay

5. (10 pts) Recall that TCP can be enhanced with TLS/SSL to provide process-to-process security services, including encryption. Does it operate at the transport layer or the application layer? If the application developer wants TCP to be enhanced with TLS/SSL, what does the developer have to do?
   - Developer needs to write their own code to encrypt data at application layer

6. (20 pts, 3x4) Go to the CS vs P2P interactive exercise at:
   http://gaia.cs.umass.edu/kurose_ross/interactive/CS_vs_P2P_download.php
   And practice until ready to solve the following problem, providing the answers below.



The problem is to distribute a file of size F = 4 Gbits to each of these 9 peers. Suppose the server has an upload rate of u = 70 Mbps, and that the 9 peers have upload rates of:
u_1 = 12 Mbps, u_2 = 24 Mbps, u_3 = 28 Mbps, u_4 = 29 Mbps, u_5 = 21 Mbps, u_6 = 15 Mbps, u_7 = 12 Mbps, u_8 = 22 Mbps, u_9 = 12 Mbps,
and download rates of:
d_1 = 10 Mbps, d_2 = 39 Mbps, d_3 = 21 Mbps, d_4 = 15 Mbps, d_5 = 32 Mbps, d_6 = 31 Mbps, d_7 = 19 Mbps, d_8 = 34 Mbps, d_9 = 21 Mbps,
Answer the following questions:

a. What is the minimum time needed to distribute this file from the central server to the 9 peers using the client-server model? (Hint: see equation 2.1 in the text).

$D_{cs} = \max\{NF/u_s , F/d_{min}\} = \max\{ 514.28, 400 \} = 514.28$ secs

b. For question 1, what is the root case of this specific minimum time: the server upload rate, or a specific client's download rate (and if so, which client?)? Explain your answer.

The root case is the server upload rate.
$D_{cs} = \max\{NF/u_s , F/d_{min}\} = \max[514.28, 400]$

c. What is the minimum time needed to distribute this file using peer-to-peer download? (Hint: see equation 2.2 in the text).

From part b the answer is 400 secs

d. For question 3, what is the root case of this specific minimum time: the server upload rate, a specific client's download rate (amd of so, which client?), or the sum of the server and peer upload rates? Explain your answer.

The root case is client download rate – specially client 1.
$D_{P2P} = \max\{F/u_s , F/d_{min} , NF/(u_s + \sum u_i)\} = \max\{57.14, 400, 146.93\}$