

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.

No.	Time	Source	Destination	Protocol	Length	Info
8	6.163045	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=20483/848, ttl=1 (no response)
9	6.176826	10.216.228.1	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
10	6.188629	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=20739/849, ttl=2 (no response)
11	6.202957	24.218.0.153	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
12	6.208597	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=20995/850, ttl=3 (no response)
13	6.234505	24.128.190.197	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
14	6.238695	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=21251/851, ttl=4 (no response)
15	6.257672	24.128.0.101	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
16	6.258750	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=21507/852, ttl=5 (no response)
17	6.286017	12.125.47.49	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18	6.288750	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=21763/853, ttl=6 (no response)
19	6.307657	12.123.40.218	192.168.1.102	ICMP	126	Time-to-live exceeded (Time to live exceeded in transit)
20	6.308748	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=22019/854, ttl=7 (no response)
21	6.334320	12.122.10.22	192.168.1.102	ICMP	126	Time-to-live exceeded (Time to live exceeded in transit)

►	Frame 8: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
►	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 Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
0100 = Version: 4
....	0101 = Header Length: 20 bytes (5)
►	Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length:	84
Identification:	0x32d0 (13008)
►	Flags: 0x0000
►	Time to live: 1
Protocol:	ICMP (1)
Header checksum:	0x2d2c [validation disabled]
Header checksum status:	Unverified

0000	00 06 25 da af 73 00 20	e0 8a 70 1a 08 00 45 00	..%.S. .p...E.
0010	00 54 32 d0 00 00 01 01	2d 2c c0 a8 01 66 80 3b	.T2.....-...f;
0020	17 64 08 00 f7 ca 03 00	50 03 37 32 20 aa aa aa	.d.....P:72...
0030	aa aa aa aa aa aa aa aa	aa aa aa aa aa aa aa aa

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?
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 because 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

368	53.778721	192.168.1.102	128.59.23.100	ICMP	582 Echo (ping) request id=0x0300, seq=50179/964, ttl=13 (reply in 380)
27	6.382957	192.205.32.106	192.168.1.102	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
57	11.388011	192.205.32.106	192.168.1.102	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
81	16.386561	192.205.32.106	192.168.1.102	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
126	29.004477	192.205.32.106	192.168.1.102	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
167	34.014412	192.205.32.106	192.168.1.102	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
209	39.036379	192.205.32.106	192.168.1.102	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
263	44.414483	192.205.32.106	192.168.1.102	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

0100	...	= Version: 4
...	0101	= Header Length: 20 bytes (5)
▼	Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)	
	0000 00..	= Differentiated Services Codepoint: Default (0)
00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 56		
Identification: 0x0000 (0)		
►	Flags: 0x0000	
	Time to live: 246	
	Protocol: ICMP (1)	
	Header checksum: 0x217f [validation disabled]	
	[Header checksum status: Unverified]	
	Source: 192.205.32.106	
	Destination: 192.168.1.102	
►	Internet Control Message Protocol	

0010	00 38 00 00 00 00 f6 01 21 7f c0 cd 20 6a c0 a8	·8·...·...·...·j·...
0020	01 66 0b 00 d9 46 00 00 00 00 45 00 00 54 32 d8	·f·...·F·...·E·...T2·
0030	00 00 01 01 f6 0e c0 a8 01 66 80 3b 17 64 08 00	...·...·f·...·d·...
0040	ef ca 03 00 58 03	...·X·

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

```

86 16.443310 192.168.1.102 128.59.23.100 ICMP 98 Echo (ping) request id=0x0300, seq=29955/885, ttl=12 (no response found!)
87 16.463382 192.168.1.102 128.59.23.100 ICMP 98 Echo (ping) request id=0x0300, seq=30211/886, ttl=13 (reply in 89)
88 16.468603 128.59.1.41 192.168.1.102 ICMP 70 Time-to-live exceeded (Time to live exceeded in transit)
89 16.499919 128.59.23.100 192.168.1.102 ICMP 98 Echo (ping) reply id=0x0300, seq=30211/886, ttl=242 (request in 87)
90 22.928093 192.168.1.102 128.119.245.12 SSH 74 Client: Encrypted packet (len=20)
91 22.952738 128.119.245.12 192.168.1.102 TCP 60 22 -> 1170 [ACK] Seq=1 Ack=21 Win=35040 Len=0
92 28.441511 192.168.1.102 128.59.23.100 IPv4 1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=32f9) [Reassembled in #93]
93 28.442185 192.168.1.102 128.59.23.100 ICMP 562 Echo (ping) request id=0x0300, seq=30467/887, ttl=1 (no response found!)
94 28.462264 10.216.228.1 192.168.1.102 ICMP 70 Time-to-live exceeded (Time to live exceeded in transit)
95 28.470668 192.168.1.102 128.59.23.100 IPv4 1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=32fa) [Reassembled in #96]
96 28.471338 192.168.1.102 128.59.23.100 ICMP 562 Echo (ping) request id=0x0300, seq=30723/888, ttl=2 (no response found!)
97 28.490663 192.168.1.102 128.59.23.100 IPv4 1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=32fb) [Reassembled in #98]

.... 0101 = Header Length: 20 bytes (5)
▼ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
0000 00.. = Differentiated Services Codepoint: Default (0)
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 1500
Identification: 0x32f9 (13049)
▼ Flags: 0x2000, More fragments
0... .. = Reserved bit: Not set
.0.. .. = Don't fragment: Not set
..1. .... = More fragments: Set
...0 0000 0000 0000 = Fragment offset: 0
► Time to live: 1
Protocol: ICMP (1)
Header checksum: 0x077b [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.1.102
Destination: 128.59.23.100
Reassembled IPv4 in frame: 93
► Data (1480 bytes)

```

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.

90	22.920953	192.168.1.102	128.119.245.12	TCP	60	22 → 1170 [ACK] Seq=1 Ack=21 Win=35040 Len=0
91	22.952738	128.119.245.12	192.168.1.102	TCP	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=32f9) [Reassembled in #93]
92	28.441511	192.168.1.102	128.59.23.100	IPv4	562	Echo (ping) request id=0x0300, seq=3046/887, ttl=1 (no response found!)
93	28.442185	192.168.1.102	128.59.23.100	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
94	28.462264	10.216.228.1	192.168.1.102	ICMP	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=32fa) [Reassembled in #96]
95	28.470668	192.168.1.102	128.59.23.100	IPv4		

Identification:	0x32f9 (13049)
Flags:	0x00b9
0...	... = Reserved bit: Not set
.0.	... = Don't fragment: Not set
.0.	... = More fragments: Not set
...0	0000 1011 1001 = Fragment offset: 185
Time to live:	1
Protocol:	ICMP (1)
Header checksum:	0x2a7a [validation disabled]
[Header checksum status:	Unverified]
Source:	192.168.1.102
Destination:	128.59.23.100
[2 IPv4 Fragments (2008 bytes): #92(1480), #93(528)]	
[Frame: 92, payload: 0-1479 (1480 bytes)]	
[Frame: 93, payload: 1480-2007 (528 bytes)]	
[Fragment count: 2]	
[Reassembled IPv4 length: 2008]	
[Reassembled IPv4 data: 0000d0c6030077037373620aaaaaaaaaaaaaaaaaaaaaa...]	

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.

Io.	Time	Source	Destination	Protocol	Length	Info
208	38.956526	12.122.12.54	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
209	39.036379	192.205.32.106	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
210	39.098928	216.140.10.30	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
211	39.164169	67.99.58.194	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
212	39.227649	128.59.1.41	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
213	39.314263	128.59.23.100	192.168.1.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0956) [Reassembled in #214]
214	39.322566	128.59.23.100	192.168.1.102	ICMP	562	Echo (ping) reply id=0x0300, seq=40195/925, ttl=242 (request in 205)
215	41.038658	192.168.1.102	199.2.53.206	TCP	62	[TCP Retransmission] 1483 → 631 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1
216	43.466136	192.168.1.102	128.59.23.100	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=3323) [Reassembled in #218]
217	43.466808	192.168.1.102	128.59.23.100	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=3323) [Reassembled in #218]
218	43.467629	192.168.1.102	128.59.23.100	ICMP	582	Echo (ping) request id=0x0300, seq=40451/926, ttl=1 (no response found!)
219	43.485786	10.216.228.1	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
220	43.492284	192.168.1.102	128.59.23.100	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=3324) [Reassembled in #222]
221	43.492953	192.168.1.102	128.59.23.100	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=3324) [Reassembled in #222]

www.wireshark.org	... = Differentiated Services Codepoint: Default (0)
....	... = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length:	1500
Identification:	0x3323 (13091)
Flags:	0x2000, More fragments
0...	... = Reserved bit: Not set
.0.	... = Don't fragment: Not set
.1.	... = More fragments: Set
...0	0000 0000 0000 = Fragment offset: 0
Time to live:	1
Protocol:	ICMP (1)
Header checksum:	0x0751 [validation disabled]
[Header checksum status:	Unverified]
Source:	192.168.1.102
Destination:	128.59.23.100
Reassembled IPv4 in frame: 218	
Data (1480 bytes)	

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

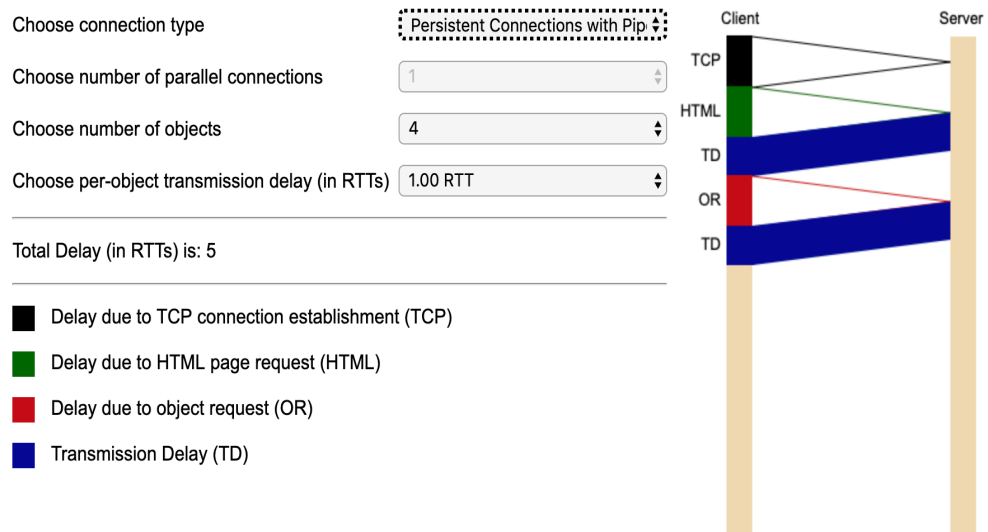
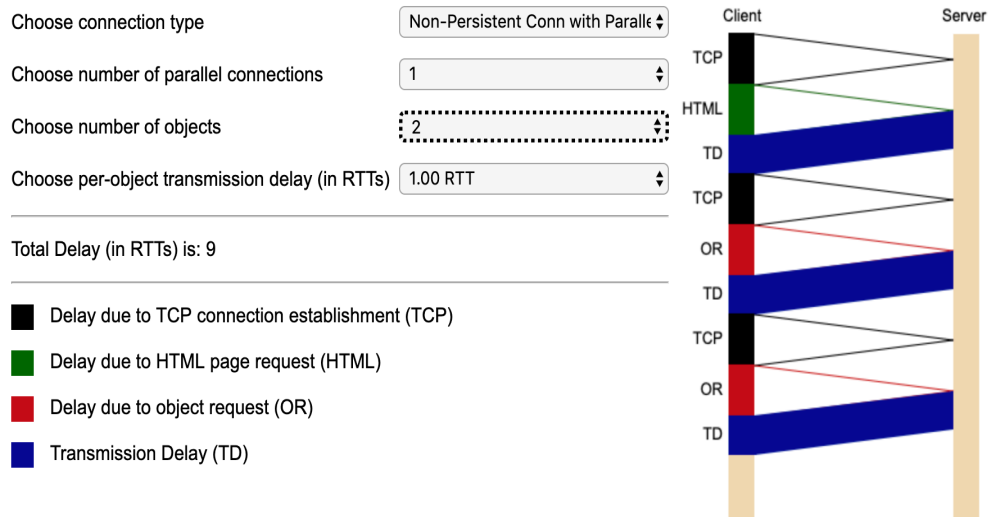
1. (15 pts) Wireshark Lab #4 – IP
Complete the fourth Wireshark Lab, #4, from the class drive. You do not need to supply your answers in a separate document; you can copy/paste them here if you like. It might be pages and pages of material – that's fine.

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

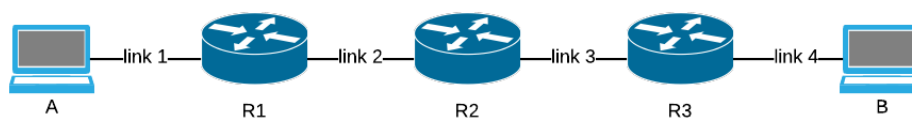
 - c. Why will Non-Persistent Connections with Parallel Connections generally win or lose for the fastest response time if up against Persistent Connections with Pipelining? Would the # of parallel connections be to blame? The number of objects? Or the per-object transmission delay? Choose one.
Blame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

$$\begin{aligned}\text{End to end} &= N \cdot (d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}}) \\ &= N \cdot (d_{\text{proc}} + L/R_i + d/s_i)\end{aligned}$$

- 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?

$$i=1 \Rightarrow 4 \cdot (0.005 + (3125 \cdot 8 / 2,500,000) + (2,500,000 / 2.5E8)) = 0.1 \text{ s}$$

$$i=2 \Rightarrow 4 \cdot (0.005 + (3125 \cdot 8 / 2,500,000) + (3,500,000 / 2.5E8)) = 0.116 \text{ s}$$

$$i=3 \Rightarrow 4 \cdot (0.005 + (3125 \cdot 8 / 2,500,000) + (3,000,000 / 2.5E8)) = 0.108 \text{ secs}$$

$$i=4 \Rightarrow 4 \cdot (0.005 + (3125 \cdot 8 / 2,500,000) + (4,000,000 / 2.5E8)) = 0.124 \text{ secs}$$

$$\text{Total_delay} = D1 + D2 + D3 + D4 = 0.448 \text{ s}$$

- c. Now suppose $R1=R2=R3=R$ and $d_{\text{proc}}=0$. 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?

$$i=1 \Rightarrow (4 \cdot 3125 \cdot 8 / 2,500,000) + (2,500,000 / 2.5E8) = a \text{ s}$$

$$i=2 \Rightarrow (4 \cdot 3125 \cdot 8 / 2,500,000) + (3,500,000 / 2.5E8) = b \text{ s}$$

$$i=3 \Rightarrow (4 \cdot 3125 \cdot 8 / 2,500,000) + (3,000,000 / 2.5E8) = c \text{ secs}$$

$$i=4 \Rightarrow (4 \cdot 3125 \cdot 8 / 2,500,000) + (4,000,000 / 2.5E8) = d \text{ secs}$$

$$\text{end to end} = a + b + c + d = 0.212 \text{ s}$$

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.

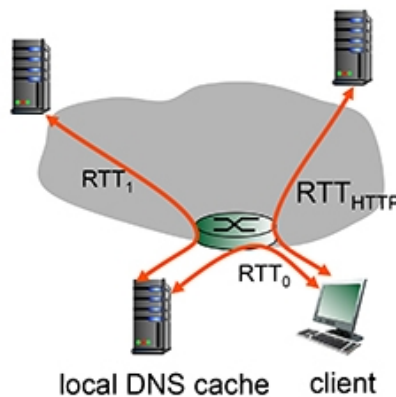
Suppose that two DNS servers are visited before your host receives the IP address from DNS. The first DNS server visited is the local DNS cache, with an RTT delay of

$$RTT_0 = 4 \text{ msec.}$$

The second DNS server contacted has an

$$RTT \text{ of } 44 \text{ msec.}$$

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

$$RTT_{\text{HTTP}} = 67 \text{ msec.}$$


- 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?

$$\text{Time} = RTT_0 + RTT_1 + 2 * RTT_{\text{http}} = 4 + 44 + 2 * 67 = 182 \text{ mS}$$

- 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?

$$RTT_0 + \dots + RTT_n + 2 * RTT_{\text{http}} + 9 * 2 * RTT_{\text{http}} = 4 + 44 + 134 + (9 * 2 * 67) = 1388 \text{ mS}$$

- 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.

$$RTT_0 + RTT_1 + 2 * RTT_{http} + 2 * RTT_{http} + 2 * RTT_{http} = 4 + 44 + 134 + 134 + 134 = 450 \text{ mS}$$

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

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

- 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 delay

c -> middle

d -> smallest delay

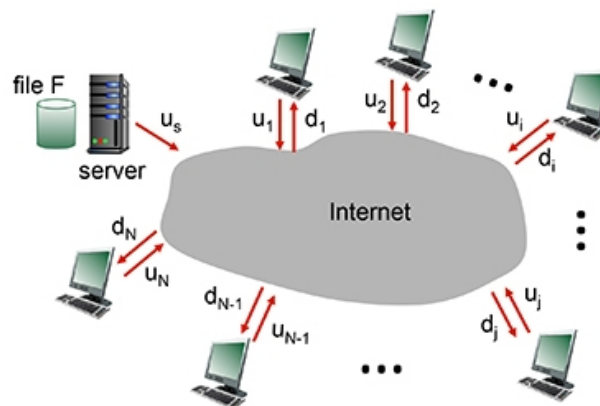
5. (10 pts) Recall that TCP can be enhanced with TLS/SSL to provide process-to-process security services, including encryption. Does it operate at the transport layer or the application layer? If the application developer wants TCP to be enhanced with TLS/SSL, what does the developer have to do?

- 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).

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

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

The root cause is client download rate – specially client 1.

$$D_{P2P} = \max\{F/u_s, F/d_{\min}, NF/(u_s + \sum u_i)\} = \max\{57.14, 400, 146.93\}$$