

COMP9121 Assignment 2 2024 S2

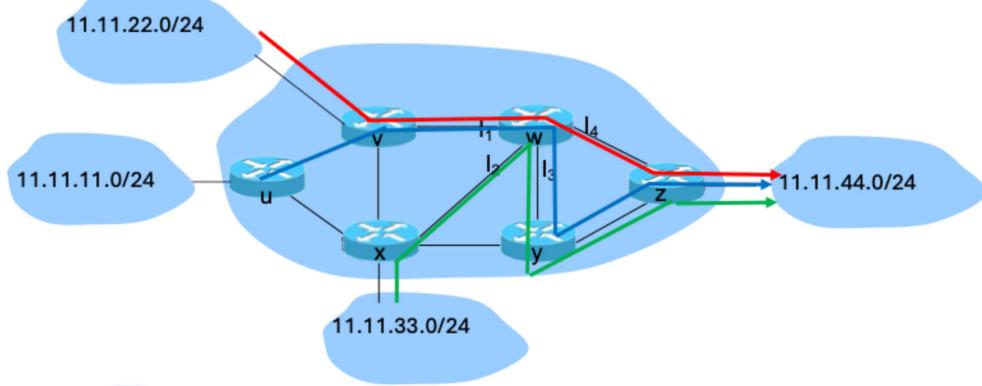
In this assignment, some questions are student number dependent; you will get zero in that question if you use another student's number.

The due date is 27 Oct 2024 at 23:59 and we will open the submission site on Canvas one week before the due date.

Submission instructions. In the “main file submission”, you can only upload your answers in a single pdf file. In the “supplementary file submission”, zip all your codes in a single zip file and submit. For Q9: you must submit your code; otherwise Q9 will not be marked. Q6: you must submit your code if you did any programming; otherwise your will not get full progress mark. This is one assignment with multiple pieces to submit. Your submission time equals to the submission time of the last piece. Late penalty will be calculated based on this submission time.

1. SDN.

In the following network, the network administrator should achieve the load balancing function: If the traffic is from 11.11.11.0/24 to 11.11.44.0/24, the blue path should be used; If the traffic is from 11.11.22.0/24 to 11.11.44.0/24, the red path should be used; If the traffic is from 11.11.33.0/24 to 11.11.44.0/24, the green path should be used;



- (1) Can this load balancing function be achieved by traditional non-SDN routers? Why or why not?
- (2) If SDN is used in the network, please specify the SDN switch table at w. The four interfaces (ports) of the SDN switch are labelled as I₁—I₄ respectively.

Please write necessary entries based on the following flow table format.

Ingress Port	Source MAC	Dest MAC	Eth Type	VLAN ID	Source IP	Dest IP	IP Protocol	Source Port	Dest Port	Action
--------------	------------	----------	----------	---------	-----------	---------	-------------	-------------	-----------	--------

(1) No, traditional non-SDN routers implemented in routers, and individual routing algorithm components in each and every router interact in the control plane (week 7 lecture P3)

As you may notice, every router will use their own algorithm to decide the path. Namely, they do not have a global view, so it cannot deal with load balancing function above since the path is not unique for every router

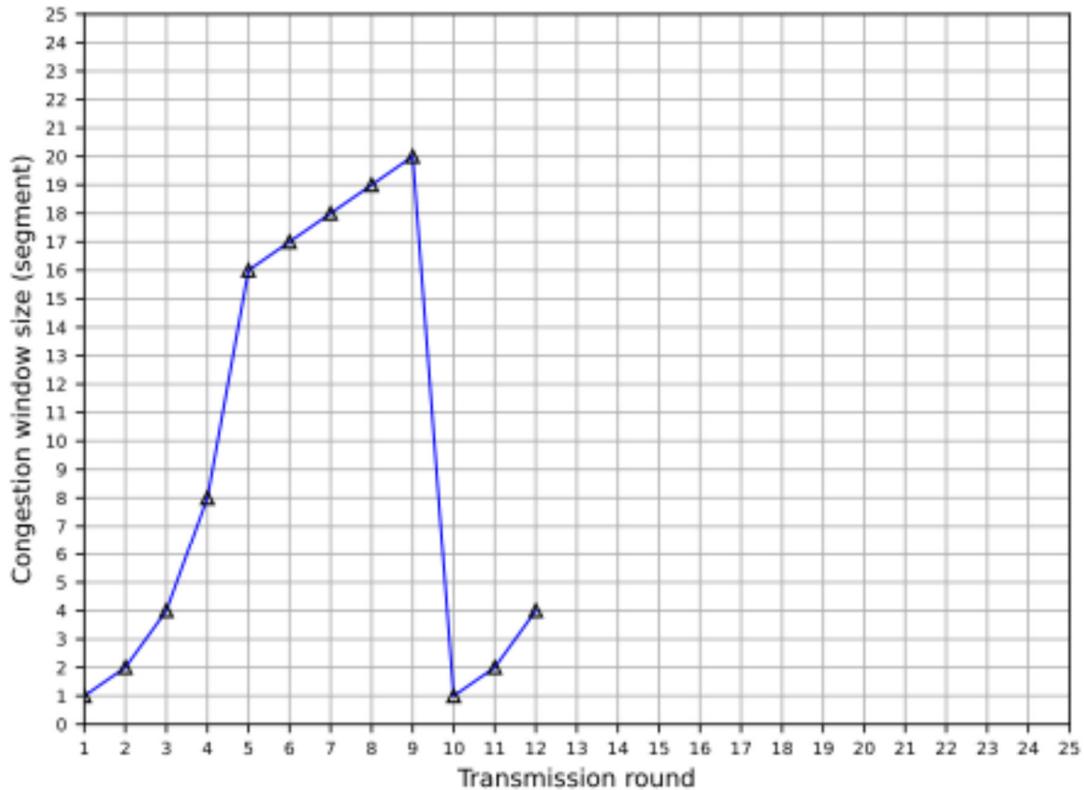
(2)

Ingress Port	Source MAC	Dest MAC	Eth Type	VLAN ID	Source IP	Dest IP	IP Protocol	Source Port	Dest Port	Action
I ₁ (Red)	*	*	*	*	11.11.22.0 124	11.11.44.0 124	*	*	*	forward I ₄
I ₁ (Blue)	*	*	*	*	11.11.11.0 124	11.11.44.0 124	*	*	*	forward I ₃
I ₂ (green)	*	*	*	*	11.11.33.0 124	11.11.44.0 124	*	*	*	forward I ₃

2. TCP

Consider the figure below. Assume TCP Reno is the protocol experiencing the behaviour, and the TCP session has a large number of packets to send. Answer the following questions (1)–(3). There is only one single TCP session in this figure.

- (1) The congestion window size is decreased to 1 (segment) at round 10. Is it caused by three duplicate ACKs or timeout? Why?
- (2) What is ssthresh at round 1? Why?
- (3) Suppose there is no packet loss after round 10. In the figure, what are the congestion window sizes from round 12 to round 25?



Key points: ① TCP RENO
② One TCP SESSION

(1) Since it's TCP RENO, so only when Timeout happens, cwnd will be reset to 1.

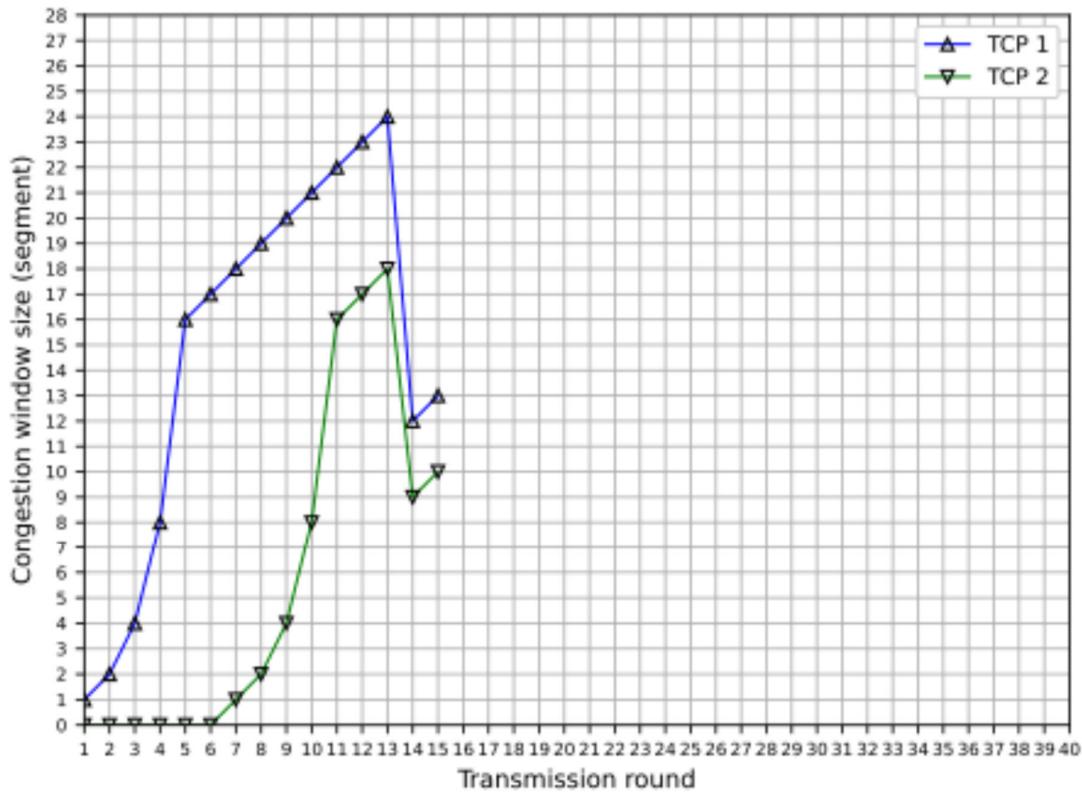
3 Duplicate ACKS will cause we half the Cwnd (fast recovery)

(2) The $\text{sc} \text{ thresh}$ is 16.

As you can see, the slow start will stop at round 5, and it correspond to cwnd with size 16

(3) there is a Timeout happened at round 9,
at that time, the size of cwnd is 20,
so the new ssthresh should be $20/2 = 10$

- (4) In Figure below. Two TCP (Reno) sessions are sharing the same link and the link capacity is limited by 42 segments. Whenever the sum of the congestion window sizes reaches (or exceeds) 42, a packet is lost for each TCP session and the packet loss is detected by three duplicate ACKs. TCP session 2 starts later than TCP session 1. What are the congestion window sizes of the two sessions from round 15 to round 40? If the congestion window size is not an integer (in segment), you need to round it down.



- Same link; limited by 42 segments
↳ namely when the sum of cwnds reach 42, will lead to package loss
- loss - 3 Duplicate ACKs,

	15	16	17	18	19	20	21	22	23	24	25	26	27	28
TCP1	13	14	15	16	17	18	19	20	21	22	23	11	12	13
TCP2	10	11	12	13	14	15	16	17	18	19	20	10	11	12

	29	30	31	32	33	34	35	36	37	38	39	40
TCP1	14	15	16	17	18	19	20	21	22	11	12	13
TCP2	13	14	15	16	17	18	19	20	21	10	11	12

43

3. Estimated RTT.

Consider the TCP procedure for estimating RTT. Let $\text{EstimatedRTT}_0 = 100$ ms be the estimated RTT when a TCP is initialised. Then, the TCP sender receives 5 ACKs and sample RTTs are measured as $\text{SampleRTT}_1, \text{SampleRTT}_2, \text{SampleRTT}_3, \text{SampleRTT}_4$, and SampleRTT_5 . All of them are 110 ms. Let EstimatedRTT_i denote the estimated RTT right after the i th ACK. We assume $\alpha = 0.125$ in this question.

(1) Calculate EstimatedRTT_4 and EstimatedRTT_5 .

(2) Generalise your solution to n sample RTTs. The TCP sender receives n ACKs, with i th sample RTT SampleRTT_i . We assume all SampleRTT_i are 110 ms. Express EstimatedRTT_n as a function of n .

(3) For the formula in part (2), let n approach infinity. What is EstimatedRTT_n ? Comment on why this averaging procedure is called an exponential moving average.

$$\alpha = 0.125 \quad \text{EstimatedRTT}_0 = 100 \text{ ms}$$

Formula: $\text{EstimatedRTT}_i = (1-\alpha) \times \text{EstimatedRTT}_{i-1} + \alpha \times \text{SampleRTT}_i$

(1) For simplicity, I will write Estimated RTT as ERTT
Sample RTT as SRTT

$$\begin{array}{c} \text{ERTT} \\ \hline \text{SRTT} \end{array}$$

$$\begin{aligned} \text{ERTT}_1 &= (1 - 0.125) \times \text{ERTT}_0 + 0.125 \times \text{SRTT}_1 = 0.875 \times 100 + 0.125 \times 110 \\ &= 101.25 \end{aligned}$$

$$\begin{aligned} \text{ERTT}_2 &= 0.875 \times \text{ERTT}_1 + 0.125 \times 110 = 0.875 \times 101.25 + 0.125 \times 110 \\ &= 102.34375 \end{aligned}$$

$$\begin{aligned} \text{ERTT}_3 &= 0.875 \times \text{ERTT}_2 + 0.125 \times 110 = 0.875 \times 102.34375 + 0.125 \times 110 \\ &= 103.3007813 \end{aligned}$$

$$\begin{aligned} \text{ERTT}_4 &= 0.875 \times \text{ERTT}_3 + 0.125 \times 110 = 0.875 \times 103.3007813 + 0.125 \times 110 \\ &= 104.13818359375 \end{aligned}$$

$$\begin{aligned}\widehat{ERTT}_5 &= 0.875 \times ERTT_4 + 0.125 \times 110 = 0.875 \times 104.13818359375 \\ &\quad + 0.125 \times 110 \\ &= 104.87091064453125\end{aligned}$$

$$\begin{aligned}(2) \quad ERTT_n &= 0.875 \times ERTT_{n-1} + 0.125 \times 110 \\ &= 0.875 \left[0.875 \times ERTT_{n-2} + 0.125 \times 110 \right] + 0.125 \times 110 \\ &= 0.875^2 \times ERTT_{n-2} + \underbrace{0.875 \times 0.125 \times 110}_{\text{underbrace}} + \underbrace{0.125 \times 110}_{\text{underbrace}} \\ &= 0.875^2 \times ERTT_{n-2} + 0.125 \times 110 (0.875 + 1) \\ &= 0.875^2 \left[0.875 \times ERTT_{n-3} + 0.125 \times 110 \right] + \\ &\quad 0.125 \times 110 (0.875 + 1) \\ &= 0.875^3 \times ERTT_{n-3} + 0.125 \times 110 \times 0.875^2 + 0.125 \times 110 (0.875 + 1) \\ &= 0.875^3 \times ERTT_{n-3} + 0.125 \times 110 \left(1 + 0.875 + 0.875^2 \right) \\ &\quad \xrightarrow{\text{...}} 0.875^0 + \dots + 0.875^{n-1}\end{aligned}$$

Using above formula, I find the pattern below

$$ERTT_n = 0.875^n \times ERTT_0 + 0.125 \times 110 \times \sum_{i=0}^{n-1} 0.875^i$$

$$= 0.875^n \times 100 + 0.125 \times 110 \times 8 \times (1 - 0.875^n)$$

Let's check:

when $n=5$, use python

`>>> import math`

`>>> pow(0.875, 5) * 100 + 0.125 * 110 * 8 * (1 - pow(0.875, 5))`

104.87091064453125

$$(3) \lim_{\substack{n \rightarrow \infty \\ \uparrow}} 0.875^n \times 100 + 0.125 \times 110 \times 8 \times (1 - 0.875^n)$$

$$= 0 + 0.125 \times 110 \times 8 \times (1 - 0)$$

$$= 110$$

notice that
n can only
be positive
number

. It's called exponential moving average because when $SRTT_i$ is fixed, the influence of $SRTT_i$ becomes smaller and smaller. As you can see, in (1)'s formula, the most recent $ERTT$ and $SRTT$ have the biggest influence (i.e. weight) and you can find former $ERTT$ and $SRTT$ influence decrease

exponentially (see the $(1-\alpha) ERT_{i-1}$ term). That's why it is called exponential moving average.

mainly refer to week 8 lecture Pg

4. HTTP and TCP.

Client C requests a webpage from Server A. We assume that RTT between Client C and Server A is 10 ms. After obtaining the main page, Client C finds that there are 2 objects to be fetched. Both objects are also in Server A.

Client C starts to request for object 1 when the main page has been successfully downloaded. It starts to request for object 2 when object 1 has been successfully downloaded. The size of the main web page is small. It fits into 1 TCP segment. Each object fits into n TCP segments, where n is the last 3 digits of your student number plus 1000. For example, if you last three digits are 123, then $n = 1123$. We have $ssthresh=64$ segments at the beginning of a TCP session. Non-persistent HTTP is used. TCP termination delay is ignored. There is no packet loss.

How long in total does it take for Client C to successfully obtain the webpage (including the main page and two objects).

ignore fast
reply

Webpage : using HTTP protocol

- Main web page has 1 TCP segment
- Each object fits into 1481 TCP segments
- $ssthresh = 64$
- Using Non-persistent:
can send at most one object over TCP connection
 - no package loss: window can increase unlimited
 - Transmission delay ignored

$$\begin{array}{r}
 \text{④ } \underbrace{1+2+4+8+16+32+64+65+66+67}_{\text{Cwdl size}} \\
 + \underbrace{68+69+70+71+72+73+74+75}_{\text{Summation}} \\
 + \underbrace{76+77+78+79+80+81+82}_{\text{ }} = 1450
 \end{array}$$

$$3 \times 7 + 1+2+1 = 25$$

- So we need to set 25 connections for each object (non-persistent connection can at most send one "object")

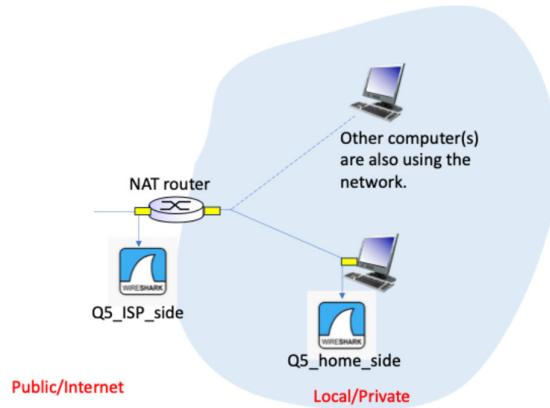
① get main page : $2RTT$

② get object 1 : $1RTT + 25RTT$
 connection transfer

③ get object 2 : $1RTT + 25RTT$

① + ② + ③ : $2 + 1 + 25 + 1 + 25 = 54 RTT$
 $= 54 \times 10$
 $= 540 \text{ ms}$

Q5. Wireshark (NAT). In this question, we investigate packets of one client PC in a home network. The PC is communicating with a remote server. The home network router provides an NAT service. The figure below shows our Wireshark trace collection scenario.



We collected Wireshark trace file (you can find them at modules – assignment 2) on the client PC in the local home network. The file is called Q5_home_side. Meanwhile, we also collected a second trace file at the interface of the home router connecting to the public Internet, as shown in the figure above. The file is called Q5_ISP_side.

(1) Use the trace files and investigate them carefully. Please write down the NAT translation table entries in the NAT router for the traffic you can find in Q5_home_side and Q5_ISP_side. Find as many entries as possible. For each entry, please give screenshots to justify your answer. (If your table has more than 5 entries, just give the screenshots for the first 5 entries in your table.) Please note that some other computers are also using the local network, and their traffic may also appear in Q5_ISP_side but not in Q5_home_side. You should ignore them when you figure out the NAT translation table.

(2) What is the MAC address of the client PC? What are the MAC addresses at the two interfaces of the NAT router? Please give you screenshots.

LAN:

IP Addr of client: 192.168.137.20

55645
 55646
 55647
 55648
 55650
 55651
 55652

NAT

IP addr of NAT: 10.66.30.63

63677 Local
 63678 Local
 63679 Local
 63683 other
 63684 other
 63685 other
 63686 Local
 63688 Local
 63689 Local
 63690 other
 63692 other
 63693 other
 63694 Local

not green part
should be ignored
in(r)

WAN

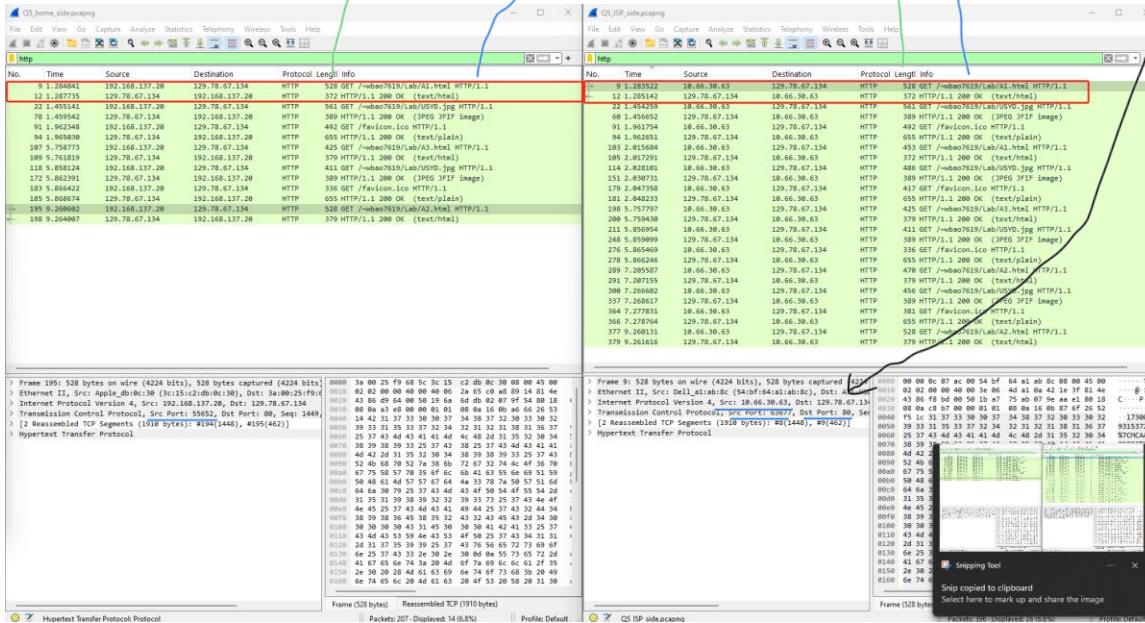
IP Addr of server: 129.78.67.134 80

	PC IP	PC Port	NAT IP	NAT Port
#	Internal IP	Internal Port	External IP	External Port
1	192.168.137.20	55645	10.66.30.63	63677
2	192.168.137.20	55646	10.66.30.63	63678
3	192.168.137.20	55647	10.66.30.63	63679
4	192.168.137.20	55648	10.66.30.63	63686
5	192.168.137.20	55650	10.66.30.63	63688
6	192.168.137.20	55651	10.66.30.63	63689
7	192.168.137.20	55652	10.66.30.63	63694

. See below blue lines, it tells NAT router info

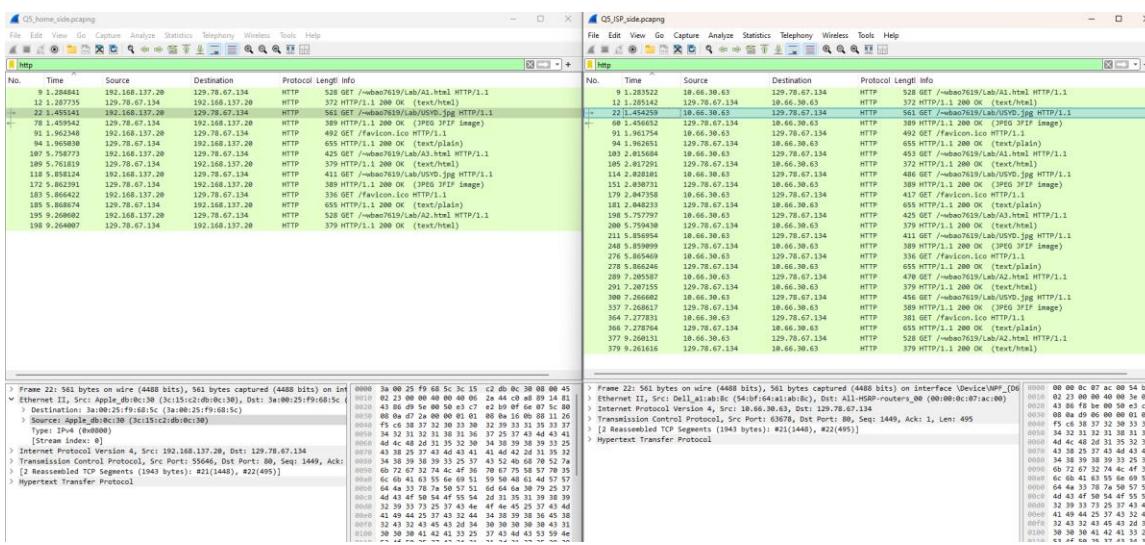
- Length are the same
- Info are same

63677



Below Images have same reason

63678



63679

Frame 91: 492 bytes on wire (3936 bits), 492 bytes captured (3936 bits) on interface \Device\NPF_{D6E8A8C0-0000-0000-0000-000000000000	Frame 91: 492 bytes on wire (3936 bits), 492 bytes captured (3936 bits) on interface \Device\NPF_{D6E8A8C0-0000-0000-0000-000000000000
> Ethernet II, Src: Apple_48:00:00 (08:00:23:ff:ff:ff), Dst: 3a:00:29:ff:68:9c (08:00:23:ff:68:9c)	> Ethernet II, Src: Apple_48:00:00 (08:00:23:ff:ff:ff), Dst: 3a:00:29:ff:68:9c (08:00:23:ff:68:9c)
> Source: Apple [192.168.1.10] (08:00:23:ff:ff:ff)	> Source: Apple [192.168.1.10] (08:00:23:ff:ff:ff)
> Type: IPv4 (0x0800)	> Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.1.107, Dst: 192.178.67.134	> Internet Protocol Version 4, Src: 192.168.1.107, Dst: 192.178.67.134
> Transmission Control Protocol, Src Port: 55647, Dst Port: 80, Seq: 1449, Ack: 1	> Transmission Control Protocol, Src Port: 55647, Dst Port: 80, Seq: 1449, Ack: 1
> [2 Reseembled TCP Segments (1874 bytes): #0(1448), #1(426)]	> [2 Reseembled TCP Segments (1874 bytes): #0(1448), #1(426)]
> Hypertext Transfer Protocol	> Hypertext Transfer Protocol

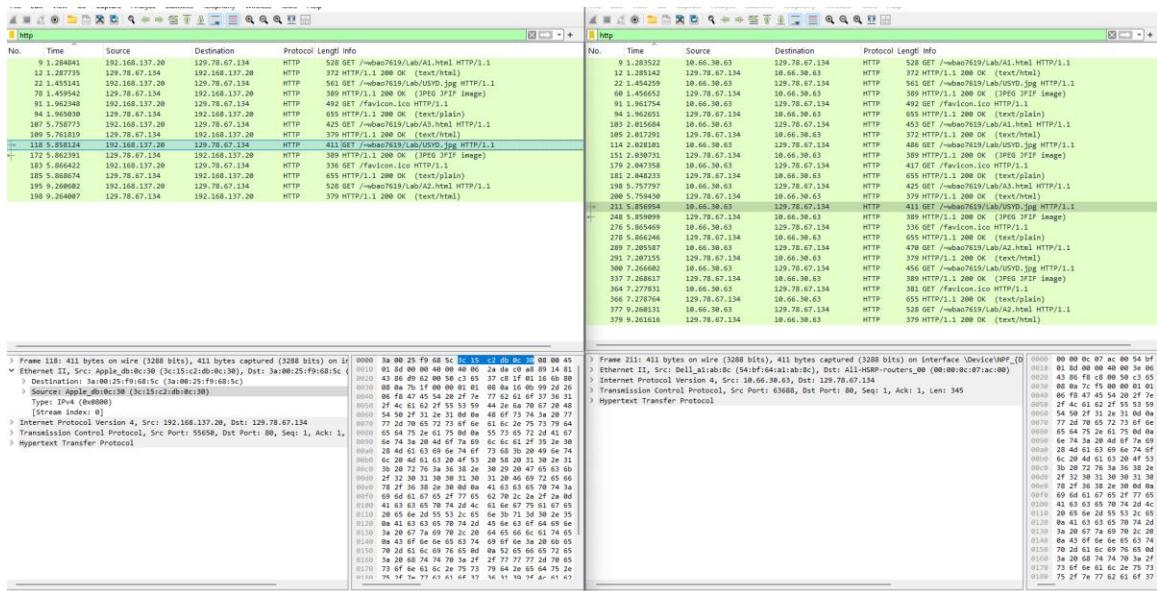
63686

No.	Time	Source	Destination	Protocol	Length	Info
8	1.288481	192.168.137.20	129.78.67.134	HTTP	528	GET /~webdir79/19/ab/A1.htm HTTP/1.1
12	1.287735	192.168.137.20	129.78.67.134	HTTP	372	HTTP/1.1 200 OK (JPEG 32PF Image)
22	1.455191	192.168.137.20	129.78.67.134	HTTP	561	GET /~webdir76/19/ab/MS00.jpg HTTP/1.1
19	1.459942	192.168.137.20	129.78.67.134	HTTP	389	HTTP/1.1 200 OK (JPEG 32PF Image)
94	1.963969	129.78.67.134	192.168.137.20	HTTP	655	HTTP/1.1 200 OK (text/plain)
1	5.785773	192.168.137.20	129.78.67.134	HTTP	425	GET /~webdir79/19/ab/A1.htm HTTP/1.1
195	5.761819	129.78.67.134	192.168.137.20	HTTP	379	HTTP/1.1 200 OK (text/html)
131	5.761819	129.78.67.134	192.168.137.20	HTTP	412	HTTP/1.1 200 OK (text/html)
172	5.862391	129.78.67.134	192.168.137.20	HTTP	389	HTTP/1.1 200 OK (JPEG 32PF Image)
183	5.866422	192.168.137.20	129.78.67.134	HTTP	336	GET /favicon.ico HTTP/1.1
5	5.866874	129.78.67.134	192.168.137.20	HTTP	655	HTTP/1.1 200 OK (text/plain)
199	5.960682	192.168.137.20	129.78.67.134	HTTP	528	GET /~webdir79/19/ab/MS02.htm HTTP/1.1
199	5.960682	192.168.137.20	129.78.67.134	HTTP	379	HTTP/1.1 200 OK (text/html)
1	5.960682	192.168.137.20	129.78.67.134	HTTP	379	HTTP/1.1 200 OK (text/html)
9	6.283522	129.78.67.134	129.78.67.134	HTTP	46	HTTP/1.1 200 OK (text/html)
12	6.285424	129.78.67.134	129.78.67.134	HTTP	166	HTTP/1.1 200 OK (text/html)
22	6.454059	129.78.67.134	129.78.67.134	HTTP	66	HTTP/1.1 200 OK (text/html)
91	6.454059	129.78.67.134	129.78.67.134	HTTP	166	HTTP/1.1 200 OK (text/html)
94	6.962651	129.78.67.134	129.78.67.134	HTTP	66	HTTP/1.1 200 OK (text/plain)
103	6.958684	129.78.67.134	129.78.67.134	HTTP	459	GET /~webdir79/19/ab/A1.htm HTTP/1.1
185	6.972197	129.78.67.134	129.78.67.134	HTTP	379	HTTP/1.1 200 OK (text/html)
185	6.972197	129.78.67.134	129.78.67.134	HTTP	412	HTTP/1.1 200 OK (text/html)
185	6.972197	129.78.67.134	129.78.67.134	HTTP	379	HTTP/1.1 200 OK (text/html)
151	6.980731	129.78.67.134	129.78.67.134	HTTP	166	HTTP/1.1 200 OK (text/html)
179	6.987358	129.78.67.134	129.78.67.134	HTTP	66	HTTP/1.1 200 OK (text/plain)
181	6.984023	129.78.67.134	129.78.67.134	HTTP	655	HTTP/1.1 200 OK (text/plain)
199	6.984023	129.78.67.134	129.78.67.134	HTTP	379	HTTP/1.1 200 OK (text/html)
209	6.957938	129.78.67.134	129.78.67.134	HTTP	66	HTTP/1.1 200 OK (text/html)
211	6.958694	129.78.67.134	129.78.67.134	HTTP	476	GET /~webdir79/19/ab/MS00.jpg HTTP/1.1
276	6.958999	129.78.67.134	129.78.67.134	HTTP	336	GET /favicon.ico HTTP/1.1
276	6.958649	129.78.67.134	129.78.67.134	HTTP	66	HTTP/1.1 200 OK (text/plain)
289	7.295857	129.78.67.134	129.78.67.134	HTTP	476	GET /~webdir79/19/ab/A2.htm HTTP/1.1
300	7.296682	129.78.67.134	129.78.67.134	HTTP	476	GET /~webdir79/19/ab/A2.htm HTTP/1.1
308	7.296682	129.78.67.134	129.78.67.134	HTTP	456	GET /~webdir79/19/ab/MS00.jpg HTTP/1.1
337	7.268613	129.78.67.134	129.78.67.134	HTTP	166	HTTP/1.1 200 OK (text/html)
364	7.277831	129.78.67.134	129.78.67.134	HTTP	389	HTTP/1.1 200 OK (JPEG 32PF Image)
364	7.277831	129.78.67.134	129.78.67.134	HTTP	389	HTTP/1.1 200 OK (JPEG 32PF Image)
372	7.299113	129.78.67.134	129.78.67.134	HTTP	220	GET /~webdir79/19/ab/MS02.htm HTTP/1.1

	Frame 187: 425 bytes on wire (3400 bits), 425 bytes captured (3400 bits) on interface L2	Frame 188: 425 bytes on wire (3400 bits), 425 bytes captured (3400 bits) on interface L2
Ethernet II, Src: Apple_00:00:c0:30 (08:00:21:c0:00:30), Dst: 3:00:25:f9:0b:5c [0x0] <-->	Ethernet II, Src: Dell_00:a1:b8:8c (08:00:41:a1:b8:8c), Dst: All-HSM-rovers-0c:00:00:07:ac:80 [0x0]	
> Destination: 3:00:25:f9:0b:5c (3:00:25:f9:0b:5c)	> Destination: All-HSM-rovers-0c:00:00:07:ac:80 [0x0]	
> Source: Apple_00:00:c0:30 (08:00:21:c0:00:30)	> Source: Dell_00:a1:b8:8c (08:00:41:a1:b8:8c)	
> Type: IP (0x0800)	> Type: IP (0x0800)	
[Stream Index: 8]	[Stream Index: 8]	
> Internet Protocol Version 4, Src: 192.168.137.20, Dst: 129.78.67.134	> Internet Protocol Version 4, Src: 192.168.137.20, Dst: 129.78.67.134	
> Transmission Control Protocol, Src Port: 55646, Dst Port: 80, Seq: 1, Ack: 1,	> Transmission Control Protocol, Src Port: 55646, Dst Port: 80, Seq: 1, Ack: 1,	
> Hypertext Transfer Protocol	> Hypertext Transfer Protocol	

63688

#5



(2) MAC address of the client PC:

3C:1S:C2:DB:OC:3D

MAC address of LAN side interface of NAT

3A:00:25:J9:68:5C

MAC address of WLAN side interface of NAT

54:6f:64:a1:ab:8c

See below image green line

(2)

6. P2P. A server distributes a file with the size of 10^9 bytes to n hosts. The upload rate of the server is 3.5 Mbps. The i -th host ($i = 1, 2, \dots, n$) has a download rate of $0.5i$ Mbps. For all hosts, the upload rate is 1 Mbps. Calculate the minimum distribution time as a function of n for P2P distribution and client-server distribution, and plot the two curves. ($n = 1, 2, \dots, 100$). You should submit your code if you did any programming to complete this task.

- Server upload rate: $3.5 \text{ Mbps} = U_s$
- i -th host download rate: $0.5i \text{ Mbps}$
- host upload rate: $1 \text{ Mbps} = U_i$
- $F = 10^9 \text{ byte} = 8 \times 10^9 \text{ bits}$
- $d_{\min} = 0.5 \text{ Mbps}$ where $i=1$

Server-client: $\frac{NF}{U_s}$ $\frac{F}{d_{\min}}$

$$D_{C-S} \geq \max \left\{ \frac{NF}{U_s}, \frac{F}{d_{\min}} \right\}$$

$$\geq \max \left\{ n \cdot \frac{8 \times 10^9}{3.5 \times 10^6}, \frac{8 \times 10^9}{0.5 \times 10^6} \right\}$$



$$D_{C-S} = \begin{cases} n \times 2285.714286 & n > 7 \\ 16000 & n \leq 7 \end{cases}$$

• Client - client :

$$D_{P2P} \geq \max \left\{ \frac{F}{U_s}, \frac{F}{d_{min}}, \frac{NF}{U_s + \sum U_i} \right\}$$

$$\geq \max \left\{ \frac{8 \times 10^9}{3.5 \times 10^6}, \frac{8 \times 10^9}{0.5 \times 10^6}, \frac{n \times 8 \times 10^9}{3.5 \times 10^6 + n \times 1 \times 10^6} \right\}$$

↓

$$D_{P2P} = \begin{cases} \frac{8 \times 10^9}{0.5 \times 10^6} = 16000, & \text{otherwise} \leftarrow \text{I got this by using desmos} \\ \frac{n \times 8 \times 10^9}{10^6 (3.5 + n)} = \frac{n \times 8 \times 10^3}{3.5 + n} - 7 \leq n \leq -4 \end{cases}$$

Simplify it, we get,

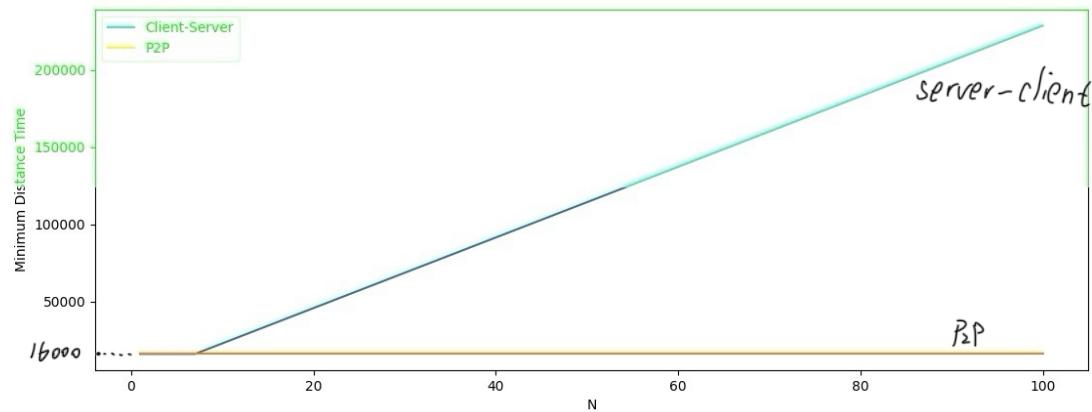
, with download speed 0.5Mbps.

$$D_{P2P} = 16000$$

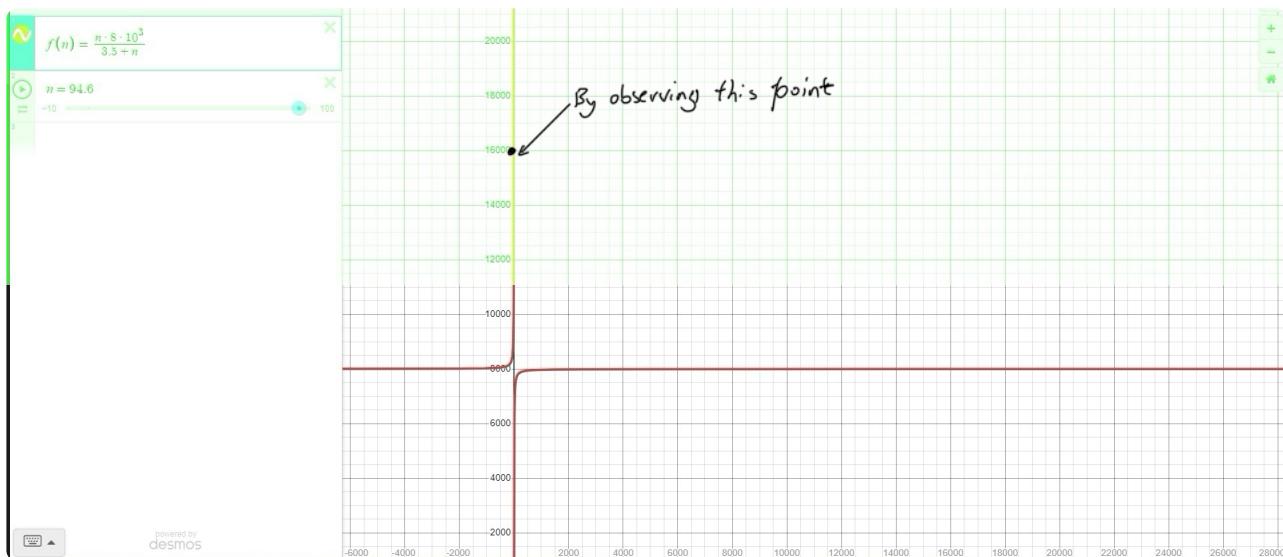
it seems the 1st client lower the distribution time

Since n cannot be negative

plot If you think below image is unclear, please run my python code



Desmos



$$\begin{aligned}
 & \text{if } 3.5 + n > 0 \text{ calculate } \frac{n \times 8 \times 10^3}{3.5 + n} > 16000 \\
 & \Rightarrow -7 < n < 3.5 \\
 & \text{if } 3.5 + n < 0 \text{ calculate } \frac{n \times 8 \times 10^3}{3.5 + n} > 16000
 \end{aligned}$$

7. Cross layer: Routing, UDP/TCP, DNS/HTTP

Consider the network shown below. In the figure, AA is the Authoritative DNS server for Web Servers A and B, and LC is the local DNS server for Client C. Client C wishes to see a webpage on Server A. The IP address of Server A is not cached in the client. The address resolution is done iteratively, i.e., all levels of DNS servers (root, TLD, and AA) should be consulted.

Assume that the one-way delay through each link inside AS1, and one-way delay through each AS outside AS1 is labeled in the figure, in ms. We assume that all dashed links have zero delay. For example, the one-way delay from Router C to AA is 30 ms. The one-way delay from Router E to Root via AS4 AS5 is 45 ms. OSPF routing protocol is used within AS1 (link cost is equivalent to one-way delay inside AS1). BGP is used among the ASes.

After resolving the IP address, Client C can visit Server A. After obtaining the main page, Client C finds that there are 2 objects to be fetched. One object is stored in Server A, but the other object is stored in Server B. Unfortunately, Client C does not know the IP address of Server B, so that Client C has to resolve the IP again.

The size of the main web page is small. It fits into 1 TCP segment. Each object is also small and fits into 1 TCP segment. No packet is lost. Persistent HTTP is used.

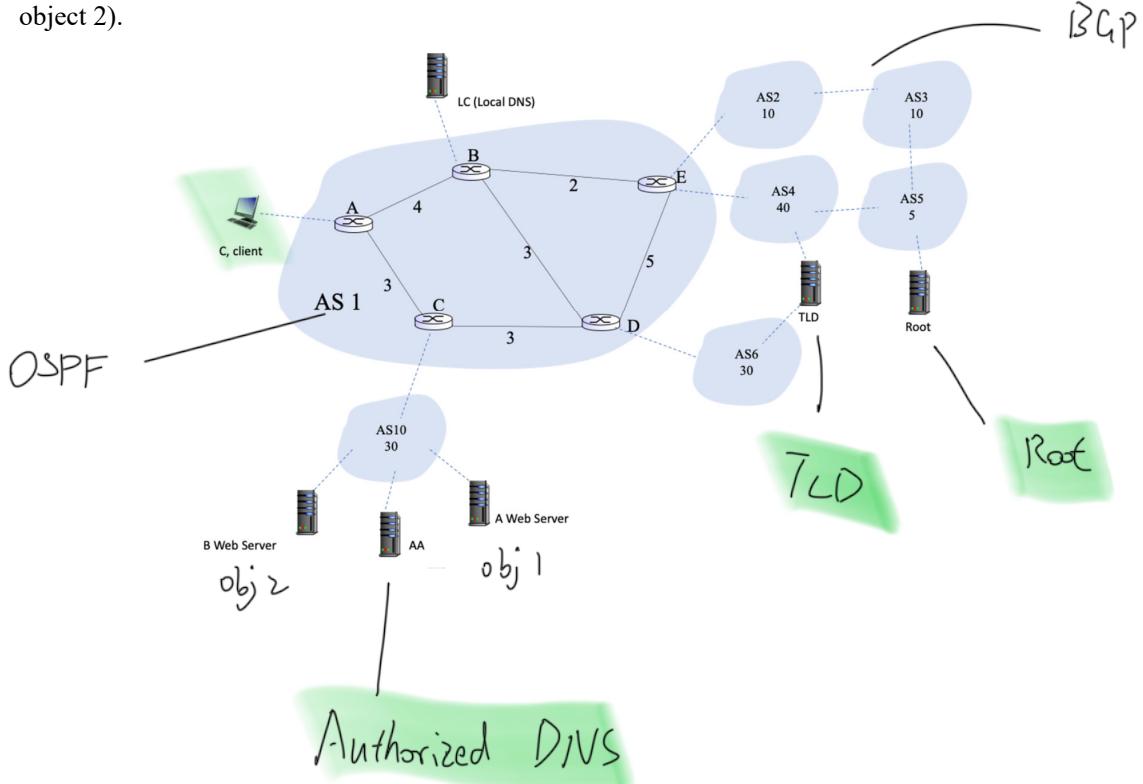
Assumption 1: All inter-AS paths are allowed, and these inter-AS paths have been known by all routers in all ASes.

Assumption 2: All levels of DNS servers should be consulted iteratively for address resolution.

Assumption 3: Client C starts to request the IP address of object 2 when object 1 has been successfully downloaded.

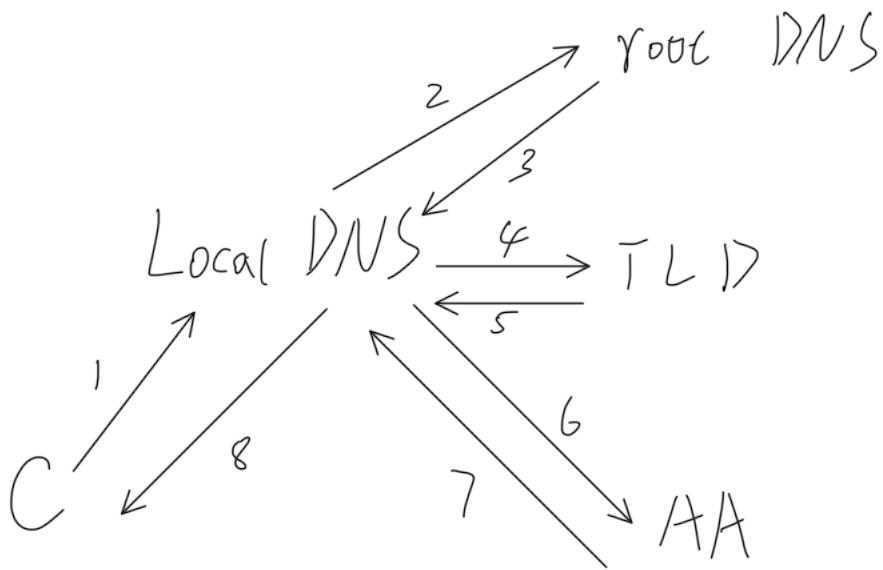
Assumption 4: TCP termination delay is ignored.

Question: How long in total does it take for Client C to successfully obtain the webpage (including: DNS of main paper, download of main page, download of object 1, DNS of object 2, and download of object 2).



Step 1 : Find A web server IP

- UDP is used for DNS
- TCP is used for HTTP



1 : $A \rightarrow B$ takes 4 ms

2 $B \rightarrow E \rightarrow \text{AS2} \rightarrow \text{AS3} \rightarrow \text{AS5}$ takes 27 ms

3 takes 27ms

4 $B \rightarrow D \rightarrow \text{AS6}$ takes 33ms

5 takes 33ms

6 $B \rightarrow D \rightarrow C \rightarrow \text{AS10}$ takes 36ms

7 takes 36ms

8 takes 4ms

In total, we got $4 \times 2 + 27 \times 2 + 33 \times 2 + 36 \times 2 =$ 200 ms

Step 2: Fetch main page from A web server

A \rightarrow C \rightarrow AS10 takes 33 ms

Get main page need 2 RTT:

$$2 \times 33 = 66 \text{ ms}$$

Step 3: Fetch object 1 from A web server

need 1 RTT: since persistent connection

$$33 \text{ ms}$$

Step 4: Find B web server IP

takes 200 ms

Step 5: Fetch object 1 from B web server

takes 2 RTT: 66 ms

In total, we have

$$200 + 66 + 33 + 200 + 66 = 565 \text{ ms}$$

Q8. RSA. Use RSA algorithm to answer the following questions. You should use decimal numbers instead of binary numbers in this question. Correct solution is not unique.

(1) Select proper values of p and q , larger than 10 but less than 20. Then, calculate n and φ .

(2) Select proper values of e and d . Then, encrypt the last two digits of your student number.

(3) Decrypt your answer for Part (2), and verify if the last two digits of your student number can be recovered.

(4) Trudy (the intruder), eavesdropped on the encrypted number and the public key (n, e) , how can she decrypt the number in this example? In reality, p and q value are very large numbers; if Trudy eavesdropped on the encrypted number and the public key, can she still decrypt the number? Why or why not?

(1), since we need to use large p, q , so I decided to use

$$p = 17 \quad q = 19$$

$$\therefore n = pq = 17 \times 19 = 323$$

$$\varphi = (p-1)(q-1) = 288$$

$$(2). \quad \gcd(e, \varphi) = 1 \quad \text{and} \quad 288 = 2^5 \times 3^2$$

Hence we can choose $e = 5$

$$\cdot \quad ed \bmod \varphi = 1$$

$$5d \bmod 288 = 1$$

By using Euclidean Function, we can get

$$5x + 288y = 1$$

$$288 = 5 \times 57 + 3$$

$$5 = 3 \times 1 + 2$$

$$3 = 2 \times 1 + 1$$

$$2 = 1 \times 2 + 0$$

$$1 = 3 - 1 \times 2 = 3 - 1 \times (5 - 1 \times 3) = 2 \times 3 - 1 \times 5$$

$$\boxed{3 = 288 - 5 \times 57} \longrightarrow = \cancel{2} \times (288 - \cancel{5} \times \cancel{57}) - \cancel{1} \times \cancel{5} \\ = 5 \times (-115) + 288 \times 2$$

so we got $d = -115$

we want to use positive number

$$\boxed{d = 288 - 115 = 173}$$

Now let's encrypt 81 # last 2 digits of student number

$$C = 81^e \bmod n = 81^5 \bmod 323 = 47$$

$$\# 81^5 = 47 + k \cdot 323$$

$$(3) m = c^d \bmod n$$

$$= 47^{173} \bmod 323$$

$$= 81 \quad \checkmark$$

(4). Trudy knows $n = 323$

$$c = 5$$

$$c = 47$$

she can try to find all possible prime

$$p' \text{ and } q' \text{ which } p'q' = 323$$

It's possible because the number of prime less than 323 is very few

$$\text{Then she can use } \gcd[(p'-1)(q'-1), e] = 1$$

to find the true value of p and q from those p' and q'

- After finding true p and q, she can easily get d by guess and check
- Namely she get the private key
- However, if p and q is very large, Trudy cannot factor big number, so she cannot get private key. (see Week 11 Pz8)
Lecture PPT

Q9. Hash. In this question, we aim to use SHA384 hash algorithm to find the nonce of the following magic sentence so that the hash has 6 leading zeros (24-bit zeros) in the beginning. Easy Python programming (less than 15 lines) will be needed in this question.

Magic sentence:

49005148 |
My number is ***** and I love COMP9121.#####

Here ***** indicates your student number (so this should be different for everyone) and ##### is the “nonce” you need to find. ##### is a string, which can be with any length. For example, the magic sentence below is the outcome (the answer is not unique). Please note, there is no limitations on the length of the nonce.

My number is 500123456 and I love COMP9121.22076159

You can verify it at many websites, such as, <https://emn178.github.io/online-tools/sha384.html>, where you can see that the hash has 6 zeros in the beginning.

The screenshot shows the SHA384 online tool interface. On the left, under 'Settings', the 'Hash' tab is selected. Other tabs include 'Auto Update' (on), 'Remember Input' (off), 'Input Encoding' (UTF-8), 'Output Encoding' (Hex (Lower Case)), and 'Enable HMAC' (off). The 'Input' field contains the text 'My number is 500123456 and I love COMP9121.22076159'. The 'Output' field displays the SHA384 hash in hex format: '00000ea6c66cd14d1e6265ff1430ae9595095638f5f53c15476140d8d8a85f6d332b815814778fd0cbf16e117b6d835'. There are also 'Share Link' and copy/paste icons.

Please note that in the “input” in the above screenshot, there should be no carriage return or line feed. Otherwise, the hash will be different.

You need to use Python3 to find a nonce. (Since the answer is not unique, you only need to find one answer.) You may use hashlib and use a correct function in this library. For example, the following program will generate SHA384 of “My number is 500123456 and I love COMP9121.”

```
import hashlib
s="My number is 500123456 and I love COMP9121."
s=bytes(s, 'utf-8')
hashresult=hashlib.sha384(s).hexdigest()
```

- (1) Write down your magic sentence and submit Python 3 code as supplementary material.
- (2) How long does it take to find the nonce?
- (3) In this case we find the nonce generating for 6 leading zeros. Comment on what will happen if you are required to find a nonce that generates a hash with 20 leading zeros. (This is what miners of bitcoins should do!)

(1) My number is 490051481 and I love COMP9121.14258490

(2) 18. 06890273094177 seconds

(3) It will take more time for my computer
to calculate it.

The worst case we need to find
nonce that could lead to 20 leading
0's need 2^{20} times of guess.

The successful probability if you guess once is:

$$\frac{1}{2^{20}}$$

SHA2-512

- SHA384
- SHA384 File
- SHA512
- SHA512 File
- SHA512/224
- SHA512/24 File
- SHA512/256
- SHA512/256 File

SHA3

- Kecak
- SHAKE
- cSHAKE
- KMAC
- RIPMD
- BLAKE

Cryptography

- AES
- DES
- Triple DES
- RC4
- ECDSA
- RSA

Encoding

- Hex (Base16)

SHA384

This SHA384 online tool helps you calculate hashes from strings. You can input UTF-8, UTF-16, Hex, Base64, or other encodings. It also supports HMAC.

Settings

Hash

Auto Update

Remember Input

Input Encoding

UTF-8

Output Encoding

Hex (Lower Case)

Enable HMAC

Output

My number is 490051481 and I love COMP9121.14258490

0000009e11ca62b1955397e52d9c54ba1507d1d9a5e0527e2c9ea1f7939620febe3776ca77ad4731a545776e837

Q9.py

```

1 start = time.time()
2
3 s_original = "My number is 490051481 and I love COMP9121."
4 originallen = len(s_original) # 41
5
6 s = bytes(s_original,'utf-8')
7 hashresult = hashlib.sha384(s).hexdigest()
8
9 # print(hashresult)
10 # ass1dc90b9de60459c8b178d24d5c8eb79226436e75a592b2bf43227647ecf53a0599ca12ab505b94efc35cbfcf5d
11
12 nonce = 0
13 while True:
14     s_with_nonce = s_original + str(nonce)
15     s = bytes(s_with_nonce,'utf-8')
16     hashresult = hashlib.sha384(s).hexdigest()
17     if hashresult[:6] == "000000":
18         break
19     nonce +=1
20
21 end = time.time()
22 execution_time = end - start
23 print(execution_time)
24 print(nonce)
25 print(hashlib.sha384("My number is 490051481 and I love COMP9121.14258490".encode('utf-8')).hexdigest()[:6] == "000000")

```

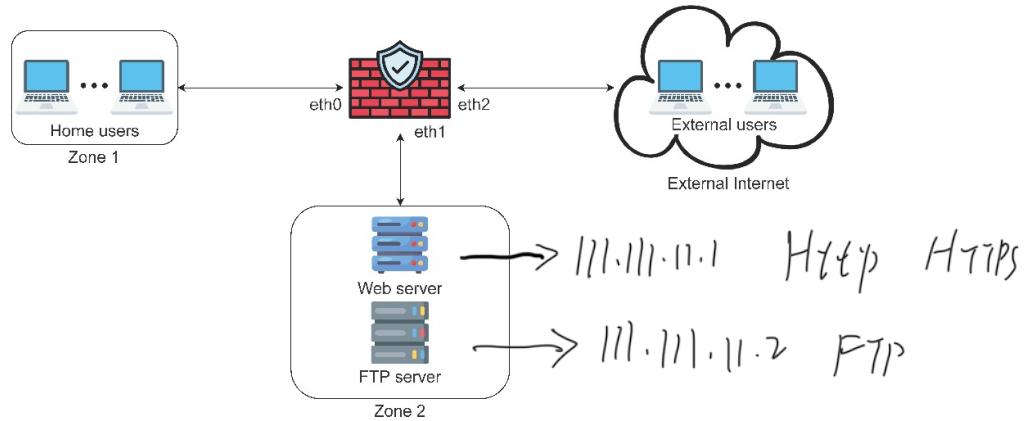
Run Q9

C:\Users\s1n\AppData\Local\Programs\Python\Python310\python.exe D:\USYD\University_Materials\USYD\Semester2_2024\COMP9121\Assignment\2\splitQ\Q9.py

18.341563191d04614
14258490
True
Process finished with exit code 0

22:15 CRLF UTF-8 4 spaces ⌂ ⓘ

10. Firewall. We will develop the firewall configuration. The figure below shows the configuration we want to achieve. The home users reside in Zone 1. The external users reside in the External Internet. A Web server (111.111.11.1) and an FTP server (111.111.11.2) are located in Zone 2. The Web server supports both http and https requests.



The security policy is as follows:

- 1) The home users can freely access any Web service, anywhere, but only if they initiate the connection themselves. *Lec 12, P49 \Rightarrow Port: 80, 443*
- 2) The home users and the external users can establish the telnet session with each other. *P48 \Rightarrow Port 23*
- 3) Everyone (home users and external users) can access the web server in Zone 2.
- 4) Only the home users can access the FTP server in Zone 2. *Week 9 lec \Rightarrow Port 21 and 20*
- 5) Block all other incoming and outgoing traffics. *P58*

Complete the table to define a stateless firewall configuration for the given scenario. You can refer to the IP ranges (source and destination) by their "zone name". Use "Ext" to indicate external Internet. Use "*" to indicate "all".

Interface	Source IP	Destination IP	Source Port	Destination Port	ACK	Action

Interface	Source IP	Destination IP	Source Port	Destination Port	ACK	Action
eth0	zone1	*	*	80, 443	Any	Allow
eth1	zone2	zone1	80, 443	*	Set	Allow
eth2	Ext	zone1	80, 443	*	Set	Allow
eth0	zone1	Ext	*	23	Any	Allow
eth2	Ext	zone1	23	*	Set	Allow
eth2	Ext	zone1	*	23	Any	Allow
eth0	zone1	Ext	23	*	Set	Allow
eth2	Ext	111.111.11.1	*	80, 443	Any	Allow
eth1	zone1	Ext	80, 443	*	Set	Allow
eth0	zone1	111.111.11.2	*	20, 21	Any	Allow
eth1	111.111.11.2	zone1	20, 21	*	Set	Allow
*	*	*	*	*	*	Deny