Aarón Josué Meza Torres B74787

Tarea 1

Ejercicio 1

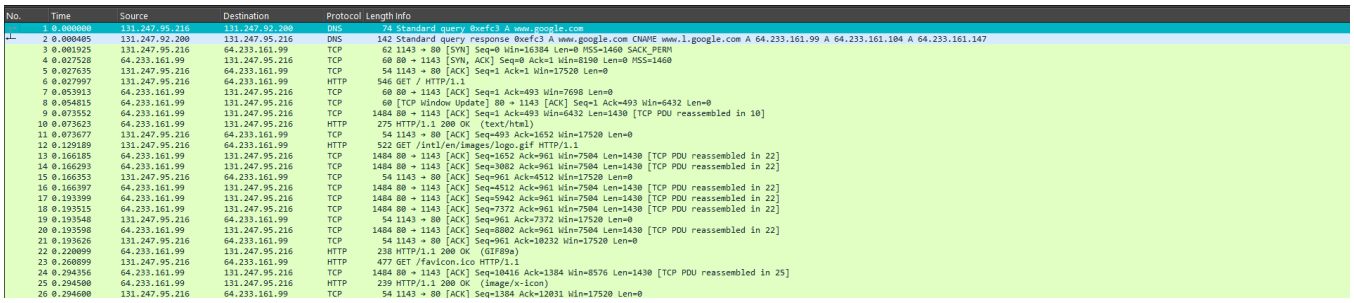
I) Exercise One

Open “Wireshark”, then use the “File” menu and the “Open” command to open the file “WS-Ex-01.pcap”. You should see 26 packets listed.

This set of packets describes a ‘conversation’ between a user’s client and a central server. This entire conversation happens automatically, after a user types something and hits enter. Look at the packets to answer the following questions in relation to this conversation.

In answering the following questions, use brief descriptions. For example, “In frame X, the client requests a web page, and in frame Y, the server delivers the content of the page.”

a) What is the IP address of the client that initiates the conversation?
b) Use the first two packets to identify the server that is going to be contacted. List the common name, and three IP addresses that can be used for the server.
c) What is happening in frames 3, 4, and 5?
d) What is happening in frames 6 and 7?
e) Ignore frame eight. However, for your information, frame eight is used to manage flow control.
f) What is happening in frames nine and ten? How are these two frames related?
g) What happens in packet 11?
h) After the initial set of packets is received, the client sends out a new request in packet 12. This occurs automatically without any action by the user. Why does this occur? See the first “hint” to the left.
i) What is occurring in packets 13 through 22?
j) Explain what happens in packets 23 through 26. See the second “hint” to the left.
k) In one sentence describe what the user was doing (Reading email? Accessing a web page? FTP? Other?).



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	131.247.95.216	131.247.92.200	DNS	74	Standard query 0xefc3 A www.google.com
2	0.000405	131.247.92.200	131.247.95.216	DNS	142	Standard query response 0xefc3 A www.google.com CNAME www.l.google.com A 64.233.161.99 A 64.233.161.104 A 64.233.161.147
3	0.001925	131.247.95.216	64.233.161.99	TCP	62	1143 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
4	0.027528	64.233.161.99	131.247.95.216	TCP	60	80 → 1143 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1460
5	0.027635	131.247.95.216	64.233.161.99	TCP	54	1143 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
6	0.027997	131.247.95.216	64.233.161.99	HTTP	546	GET / HTTP/1.1
7	0.053913	64.233.161.99	131.247.95.216	TCP	60	80 → 1143 [ACK] Seq=1 Ack=493 Win=7698 Len=0
8	0.054815	64.233.161.99	131.247.95.216	TCP	60	[TCP Window Update] 80 → 1143 [ACK] Seq=1 Ack=493 Win=6432 Len=0
9	0.073552	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=1 Ack=493 Win=6432 Len=1430 [TCP PDU reassembled in 10]
10	0.073623	64.233.161.99	131.247.95.216	HTTP	275	HTTP/1.1 200 OK (text/html)
11	0.073677	131.247.95.216	64.233.161.99	TCP	54	1143 → 80 [ACK] Seq=493 Ack=1652 Win=17520 Len=0
12	0.129189	131.247.95.216	64.233.161.99	HTTP	522	GET /intl/en/images/logo.gif HTTP/1.1
13	0.166185	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=1652 Ack=961 Win=7504 Len=1430 [TCP PDU reassembled in 22]
14	0.166293	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=3082 Ack=961 Win=7504 Len=1430 [TCP PDU reassembled in 22]
15	0.166353	131.247.95.216	64.233.161.99	TCP	54	1143 → 80 [ACK] Seq=961 Ack=4512 Win=17520 Len=0
16	0.166397	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=4512 Ack=961 Win=7504 Len=1430 [TCP PDU reassembled in 22]
17	0.193399	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=5942 Ack=961 Win=7504 Len=1430 [TCP PDU reassembled in 22]
18	0.193515	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=7372 Ack=961 Win=7504 Len=1430 [TCP PDU reassembled in 22]
19	0.193548	131.247.95.216	64.233.161.99	TCP	54	1143 → 80 [ACK] Seq=961 Ack=7372 Win=17520 Len=0
20	0.193598	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=8802 Ack=961 Win=7504 Len=1430 [TCP PDU reassembled in 22]
21	0.193626	131.247.95.216	64.233.161.99	TCP	54	1143 → 80 [ACK] Seq=961 Ack=10232 Win=17520 Len=0
22	0.220099	64.233.161.99	131.247.95.216	HTTP	238	HTTP/1.1 200 OK (GIF89a)
23	0.260899	131.247.95.216	64.233.161.99	HTTP	477	GET /favicon.ico HTTP/1.1
24	0.294356	64.233.161.99	131.247.95.216	TCP	1484	80 → 1143 [ACK] Seq=10416 Ack=1384 Win=8576 Len=1430 [TCP PDU reassembled in 25]
25	0.294500	64.233.161.99	131.247.95.216	HTTP	239	HTTP/1.1 200 OK (image/x-icon)
26	0.294600	131.247.95.216	64.233.161.99	TCP	54	1143 → 80 [ACK] Seq=1384 Ack=12031 Win=17520 Len=0

A. La dirección IP del cliente que inicia la conversación es 131.247.95.216
B. El server que quiere ser contactado es www.google.com y sus 3 IPs son 64.233.161.99, 64.233.161.104 y 64.233.161.104
C. Lo que pasa entre el frame 3 y 5 es que se inicia un proceso de three way handshake:

    a. En el frame 3, se intenta realizar una conexión con un servidor realizando una solicitud con el número de secuencia 0.
    b. En el frame 4, el servidor responde a la solicitud con su propio SYN y un ACK confirmando que recibió la solicitud.
    c. En el frame 5, el cliente aumenta su número de secuencia y envía otro ACK para poder terminar de establecer la conexión con el servidor, permitiendo de este modo el intercambio de información.

D. Una vez establecida la conexión TCP, entre el frame 6 y 7 pasa:
    a. En el frame 6, se realiza una solicitud HTTP de la página principal del servidor
    b. En el frame 7, el servidor envía un ACK indicando que recibió la solicitud.

E. Se ignora el frame 8, según indicaciones. Sin embargo, se menciona que se encarga del control de flujo.

F. En el frame 9 y 10 pasa:
    a. En el frame 9, inicia la respuesta por parte del servidor al cliente.
    b. En el frame 10, se envía la información solicitada por el cliente, terminando la respuesta del servidor. Ambos se relacionan porque juntos forman la respuesta completa del servidor. Ambos contienen información importante que una vez que se une contiene la información solicitada.

G. En el frame 11, el cliente envía un ACK al servidor para indicarle que recibió los datos enviados.

H. Esto sucede porque la solicitud del frame 12 es un elemento necesario para cargar por completo la página, por lo que después de que el usuario solicite la página, se pueden realizar solicitudes automáticas para cargar otros elementos necesarios para poder mostrar al usuario correctamente su solicitud inicial, que en este caso es la página. Esto también está relacionado con el comportamiento de la caché DNS, la cual en este caso realiza una consulta.

I. Entre el frame 13 y 22 sucede:
    a. En los frames 13,14,16,17,18 y 19 el servidor envía fragmentos de la información solicitada por el cliente, en este caso una imagen.
    b. Entre los frames 15, 19 y 21 el cliente envía un ACK al servidor para indicar que recibió la información.
    c. En el frame 23, el servidor indica la respuesta HTTP 200 OK que incluye los datos completos de la imagen y el tipo de dato.

J. Entre el frame 23 y 26 sucede:
    a. En el frame 23, el cliente solicita al servidor un archivo llamado favicon.ico, que como se menciona en la pista, es un gráfico pequeño que se utiliza como icono para identificar una página web.

K. El usuario se conectó a una página web para solicitar archivos.

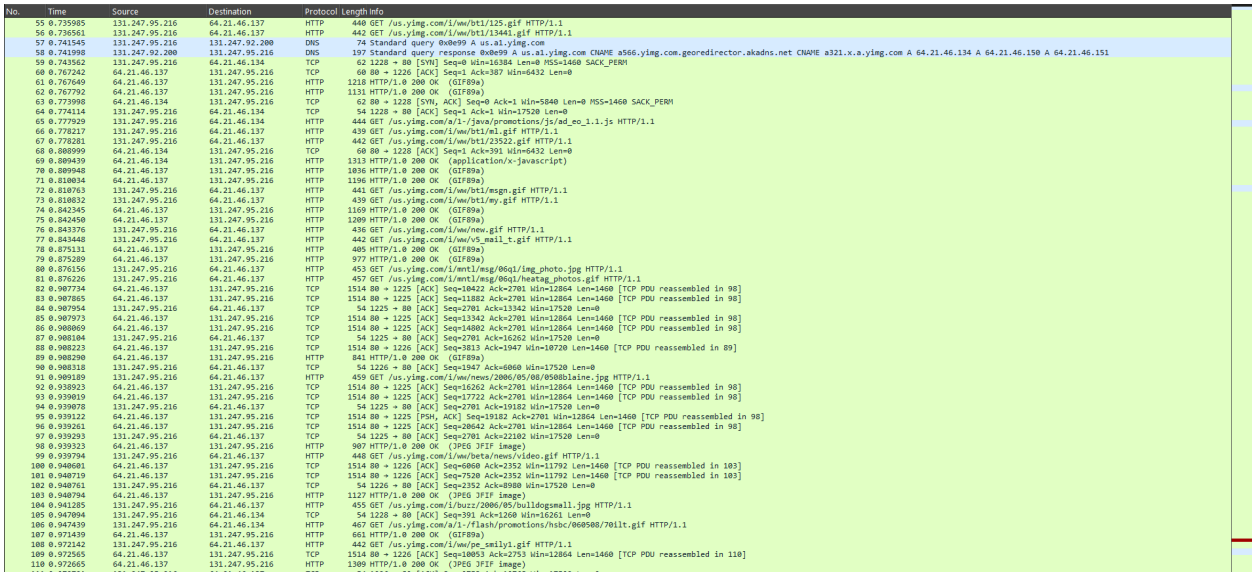
Ejercicio 2

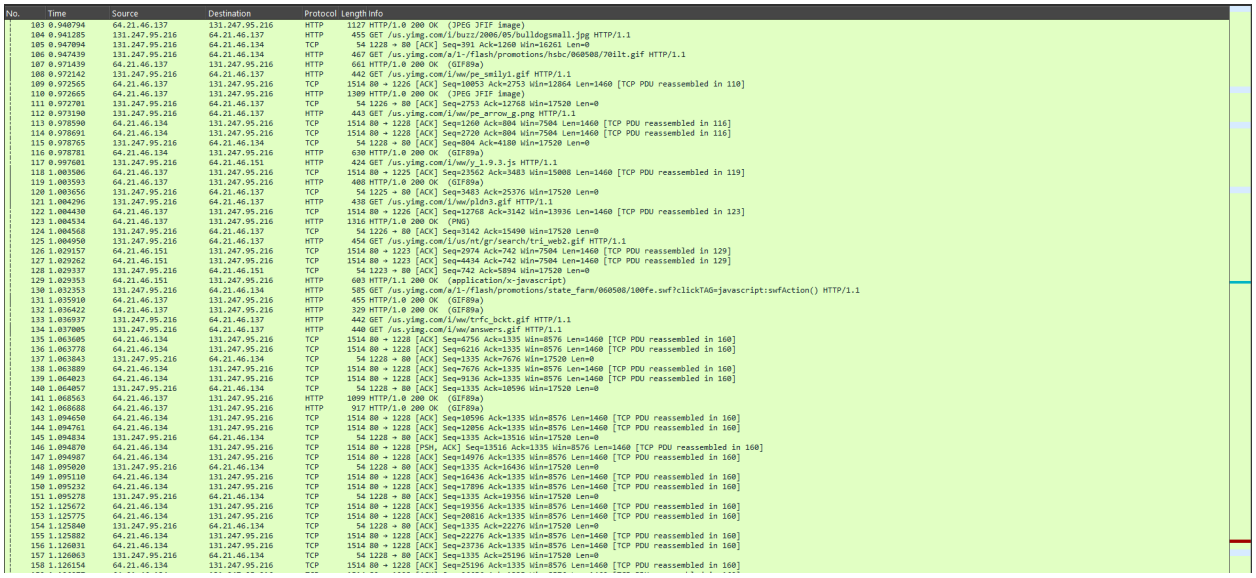
II) Exercise Two

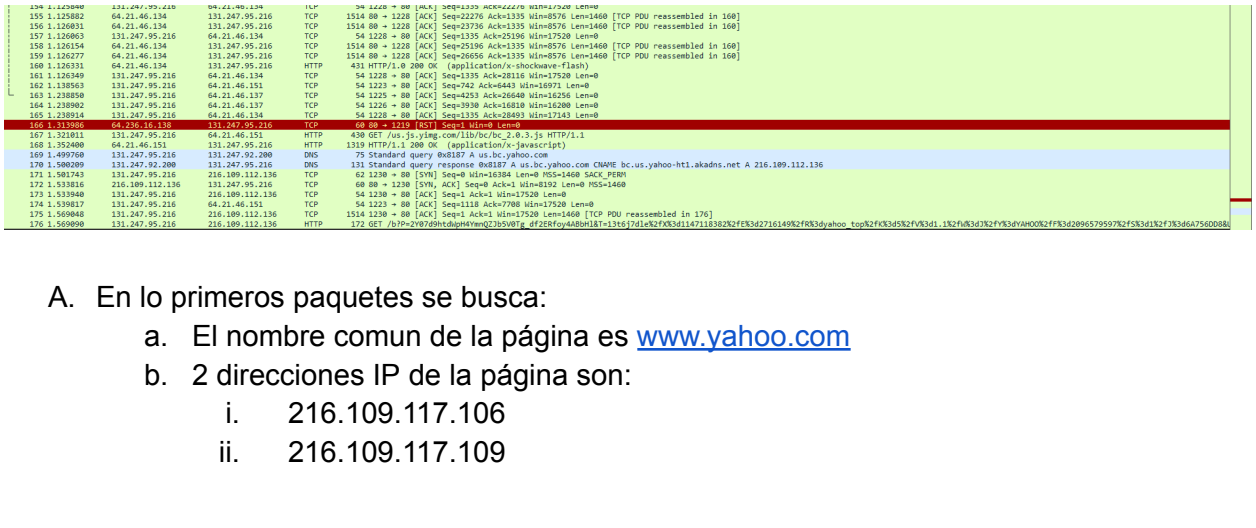
Open “Wireshark”, then use the “File” menu and the “Open” command to open the file “WS-Ex-02.pcap”. You should see 176 packets listed.

a) In the first few packets, the client machine is looking up the common name (cname) of a web site to find its IP address. What is the cname of this web site? Give two IP addresses for this web site.
b) How many packets/frames does it take to receive the web page (the answer to the first http get request only)?
c) Does this web site use gzip to compress its data for sending? Does it write cookies? In order to answer these questions, look under the payload for the reassembled packet that represents the web page. This will be the last packet from question b above. Look to see if it has “Content-Encoding” set to gzip, and to see if it has a “Set-Cookie” to write a cookie.
d) What is happening in packets 26 and 27? Does every component of a web page have to come from the same server? See the Hint to the left.
e) In packet 37 we see another DNS query, this time for us.i1.yimg.com. Why does the client need to ask for this IP address? Didn’t we just get this address in packet 26? (This is a trick question; carefully compare the two common names in packet 26 and 37.)
f) In packet 42 we see a HTTP “Get” statement, and in packet 48 a new HTTP “Get” statement. Why didn’t the system need another DNS request before the second get statement? Click on packet 42 and look in the middle window. Expand the line titled “Hypertext Transfer Protocol” and read the “Host:” line. Compare that line to the “Host:” line for packet 48.
g) Examine packet 139. It is one segment of a PDU that is reassembled with several other segments in packet 160. Look at packets 141, 142, and 143. Are these three packets also part of packet 160? What happens if a set of packets that are supposed to be reassembled do not arrive in a continuous stream or do not arrive in the proper order?
h) Return to examine frames 141 and 142. Both of these are graphics (GIF files) from the same source IP address. How does the client know which graphic to match up to each get statement? Hint: Click on each and look in the middle window for the heading line that starts with “Transmission Control Protocol”. What difference do you see in the heading lines for the two files? Return to the original “Get” statements. Can you see the same difference in the “Get” statements?

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	131.247.95.216	131.247.92.200	DNS	73	Standard query 0x159b A www.yahoo.com
2	0.001630	131.247.92.200	131.247.95.216	DNS	542	Standard query response 0x159b A www.yahoo.com CNAME www.yahoo.akadns.net A 216.109.117.106 A 216.109.117.109 A 216.109.117.110 A 216.109.117.204 A 216.109.117.206 A 216.109.118.70 A 216.1
3	0.003754	131.247.95.216	216.109.117.106	TCP	62	1221 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
4	0.028756	216.109.117.106	131.247.95.216	TCP	60	80 → 1221 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
5	0.028873	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
6	0.032252	131.247.95.216	216.109.117.106	HTTP	492	GET / HTTP/1.1
7	0.082807	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=1 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
8	0.082945	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=1461 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
9	0.082998	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [ACK] Seq=439 Ack=2921 Win=17520 Len=0
10	0.083065	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=2921 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
11	0.108891	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=4381 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
12	0.109005	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=5841 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
13	0.109040	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [ACK] Seq=439 Ack=5841 Win=17520 Len=0
14	0.109106	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=7301 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
15	0.109134	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [ACK] Seq=439 Ack=8761 Win=17520 Len=0
16	0.134687	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=8761 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
17	0.134777	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=10221 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
18	0.134834	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [ACK] Seq=439 Ack=11681 Win=17520 Len=0
19	0.134910	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=11681 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
20	0.135036	216.109.117.106	131.247.95.216	TCP	1514	80 → 1221 [ACK] Seq=13141 Ack=439 Win=65535 Len=1460 [TCP PDU reassembled in 22]
21	0.135069	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [ACK] Seq=439 Ack=14601 Win=17520 Len=0
22	0.135124	216.109.117.106	131.247.95.216	HTTP	1115	HTTP/1.1 200 OK (text/html)
23	0.135175	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [ACK] Seq=439 Ack=15663 Win=16459 Len=0
24	0.135590	131.247.95.216	216.109.117.106	TCP	54	1221 → 80 [FIN, ACK] Seq=439 Ack=15663 Win=16459 Len=0
25	0.160526	216.109.117.106	131.247.95.216	TCP	60	80 → 1221 [ACK] Seq=15663 Ack=440 Win=65535 Len=0
26	0.252293	131.247.95.216	131.247.92.200	DNS	75	Standard query 0x409b A us.js2.yimg.com
27	0.254735	131.247.92.200	131.247.95.216	DNS	198	Standard query response 0x409b A us.js2.yimg.com CNAME a321.yimg.com.georedirector.akadns.net CNAME a321.x.a.yimg.com A 64.21.46.151 A 64.21.46.134 A 64.21.46.150
28	0.257844	131.247.95.216	64.21.46.151	TCP	62	1223 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
29	0.287806	64.21.46.151	131.247.95.216	TCP	62	80 → 1223 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
30	0.287939	131.247.95.216	64.21.46.151	TCP	54	1223 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
31	0.319189	131.247.95.216	64.21.46.151	HTTP	425	GET /us.yimg.com/i/ww/ult3_1.3.js HTTP/1.1
32	0.349927	64.21.46.151	131.247.95.216	TCP	60	80 → 1223 [ACK] Seq=1 Ack=372 Win=6432 Len=0
33	0.350907	64.21.46.151	131.247.95.216	TCP	1514	80 → 1223 [ACK] Seq=1 Ack=372 Win=6432 Len=1460 [TCP PDU reassembled in 35]
34	0.351033	64.21.46.151	131.247.95.216	TCP	1514	80 → 1223 [ACK] Seq=1461 Ack=372 Win=6432 Len=1460 [TCP PDU reassembled in 35]
35	0.351062	64.21.46.151	131.247.95.216	HTTP	107	HTTP/1.1 200 OK (application/x-javascript)
36	0.351128	131.247.95.216	64.21.46.151	TCP	54	1223 → 80 [ACK] Seq=372 Ack=2974 Win=17520 Len=0
37	0.483217	131.247.95.216	131.247.92.200	DNS	74	Standard query 0xf099 A us.i1.yimg.com
38	0.484549	131.247.92.200	131.247.95.216	DNS	181	Standard query response 0xf099 A us.i1.yimg.com CNAME a943.yimg.com.georedirector.akadns.net CNAME a943.x.a.yimg.com A 64.21.46.137 A 64.21.46.144
39	0.486236	131.247.95.216	64.21.46.137	TCP	62	1225 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
40	0.517484	64.21.46.137	131.247.95.216	TCP	62	80 → 1225 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
41	0.517588	131.247.95.216	64.21.46.137	TCP	54	1225 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
42	0.521681	131.247.95.216	64.21.46.137	HTTP	423	GET /us.yimg.com/i/ww/dsf_1.1.js HTTP/1.1
43	0.552584	64.21.46.137	131.247.95.216	TCP	60	80 → 1225 [ACK] Seq=1 Ack=370 Win=6432 Len=0
44	0.553408	64.21.46.137	131.247.95.216	TCP	1514	80 → 1225 [ACK] Seq=1 Ack=370 Win=6432 Len=1460 [TCP PDU reassembled in 47]
45	0.553530	64.21.46.137	131.247.95.216	TCP	1514	80 → 1225 [ACK] Seq=1461 Ack=370 Win=6432 Len=1460 [TCP PDU reassembled in 47]
46	0.553593	131.247.95.216	64.21.46.137	TCP	54	1225 → 80 [ACK] Seq=370 Ack=2921 Win=17520 Len=0
47	0.553629	64.21.46.137	131.247.95.216	HTTP	1287	HTTP/1.0 200 OK (application/x-javascript)
48	0.594952	131.247.95.216	64.21.46.137	HTTP	440	GET /us.yimg.com/i/ww/beta/y3.gif HTTP/1.1
49	0.597939	131.247.95.216	64.21.46.137	TCP	62	1226 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
50	0.626408	64.21.46.137	131.247.95.216	TCP	1514	80 → 1225 [ACK] Seq=4154 Ack=756 Win=7504 Len=1460 [TCP PDU reassembled in 51]
51	0.626482	64.21.46.137	131.247.95.216	HTTP	765	HTTP/1.0 200 OK (GIF89a)
52	0.626534	131.247.95.216	64.21.46.137	TCP	54	1225 → 80 [ACK] Seq=756 Ack=6325 Win=17520 Len=0
53	0.628641	64.21.46.137	131.247.95.216	TCP	62	80 → 1226 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM
54	0.628746	131.247.95.216	64.21.46.137	TCP	54	1226 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
55	0.735985	131.247.95.216	64.21.46.137	HTTP	440	GET /us.yimg.com/i/ww/bt1/125.gif HTTP/1.1
56	0.736561	131.247.95.216	64.21.46.137	HTTP	442	GET /us.yimg.com/i/ww/bt1/13441.gif HTTP/1.1



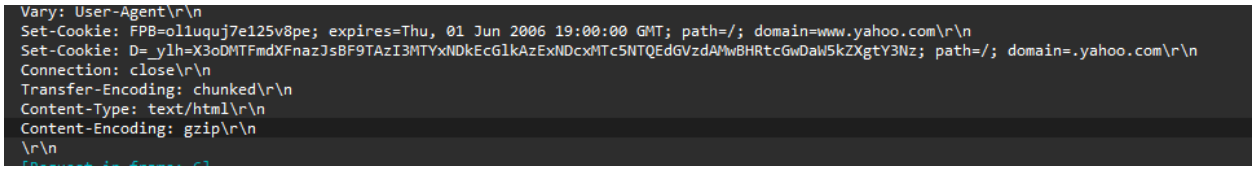




A. En lo primeros paquetes se busca:
   a. El nombre comun de la página es www.yahoo.com
   b. 2 direcciones IP de la página son:
      i. 216.109.117.106
      ii. 216.109.117.109

B. Para recibir la página se toman 20, tomando en cuenta el paquete que realiza la solicitud y el paquete de confirmación de la respuesta completa y cierre de la comunicación que van desde el frame 6 que es el inicio de la solicitud, el frame 22 que es el ensamblado de la respuesta y el frame 25 que es el frame de cierre de la comunicación. Cómo tal la respuesta acaba en el frame 22.

C. Sí, el servidor utiliza gzip para enviar contenido comprimido y escribe cookies en su respuesta, como se puede observar en la siguiente imagen:



D. En los frames 26 y 27 sucede:
   a. En el frame 26, el cliente realiza una solicitud DNS para obtener la dirección de la página js2.yimg.com
   b. En el frame 27, el servidor DNS envía la respuesta, indicando que la dirección de la página está asociada a otros dominios como a321.yimg.com.georedirector.akadns.net y a a321.x.a.yimg.com. Además, le devuelve una serie de direcciones IPs asociadas a la página, como, 64.21.46.151, 64.21.46.134 y 64.21.46.150

E. El cliente solicita la dirección IP puesto que, aunque ambas páginas pertenecen al mismo dominio, no pertenecen al mismo subdominio, lo que puede indicar que pertenecen a funciones o servidores diferentes.

F. El sistema no necesita otra solicitud DNS porque como se observa en la linea de Host tanto del frame 42 como 47, ese nombre ya se había resuelto y es el mismo en las solicitudes de ambos frames.

G. Solo los paquetes 139 y 143 son parte del paquete 160. Los paquetes 141 y 142 son respuestas independientes del paquete 160. Si los paquetes no llegan en orden, el protocolo TCP se encarga de ordenarlos, almacenando los en un buffer para ordenarlos por su número de secuencia.

H. Para poder identificar a qué solicitud get asigna cada gráfico, el cliente utiliza el número de puerto de destino. En el frame 141 el puerto es 1226 y en el 143 es 1225

Ejercicio 3

III) Exercise Three

Open "Wireshark", then use the "File" menu and the "Open" command to open the file "WS-Ex-03.pcap". You should see 22 packets listed.

These packets represent two different requests for web pages. Packets 1-7 involve the request for the web page www.yahoo.com. Packets 8-22 involve the request for the web page my.usf.edu.

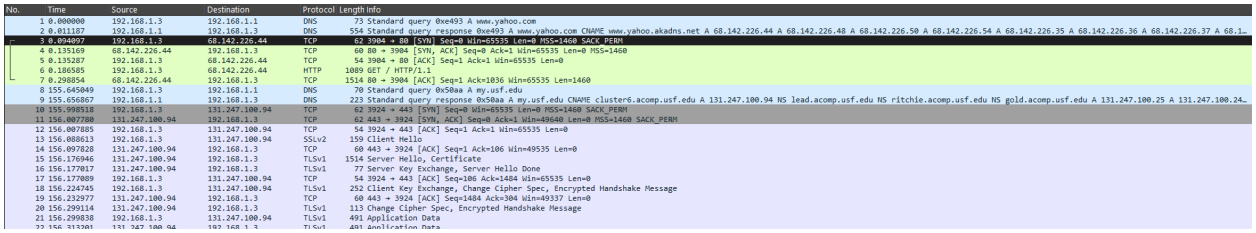
a) Compare the destination port in the TCP packet in frame 3 with the destination port in the TCP packet in frame 12. What difference do you see? What does this tell you about the difference in the two requests?

The following table compares the two requests for web pages. For example, row i) shows that frames 1-2 and frames 8-9 represent the DNS lookups for each of the web requests.

Row	www.yahoo.com frames	my.usf.com frames	Brief Explanation of Activity
i)	1-2	8-9	DNS Request to find IP address for common name & DNS Response
ii)	3-5	10-12	Three-way handshake
iii)	--	13-20	
iv)	6	21	"Get" request for web page
v)	7	22	First packet from web server with web page content.

b) Explain what is happening in row "iii" above. Why are there no frames listed for yahoo in row "iii"?

c) Look at the "Info" column on frame 6. It says: "GET / HTTP / 1.1. What is the corresponding Info field for the my.usf.com web request (frame 21)? Why doesn't it read the same as in frame 6?



A. Se puede observar que el frame 3 y el 12 tienen puertos de destino diferentes. El frame 3 tiene el puerto de destino 80, lo que sugiere que se utiliza HTTP y el frame 12 tiene como puerto de destino 443, indicando que utiliza HTTPS. Esto da a entender que el frame 12 utiliza un protocolo cifrado y el frame 3 no.

B. Al analizar la tabla se observa que entre los frames 13 y 20, se realiza el proceso para establecer una conexión segura para acceder al sitio my.usf.com. El hecho de que no aparezca nada en la fila III de la columna de yahoo.com indica que no hubieron paquetes o frames dedicados a establecer una conexión segura con esta página.

C. La razón por la cual la solicitud realizada por el frame 6 y el 21 se ven distintas es porque utilizan protocolos de comunicación diferentes. El frame 6 utiliza HTTP, mientras que el frame 21 utiliza TLSv1, el cuál es un protocolo seguro y se encarga de cifrar las solicitudes, por lo que estas no son visibles como sí lo son con HTTP.

Ejercicio 4

IV) Exercise Four

In this exercise, you are going to capture live traffic from your computer. Open up Wireshark and use the "Capture" menu to save live traffic. The Wireshark "QuickStart" guide distributed with these exercises contains more instructions on using Wireshark.

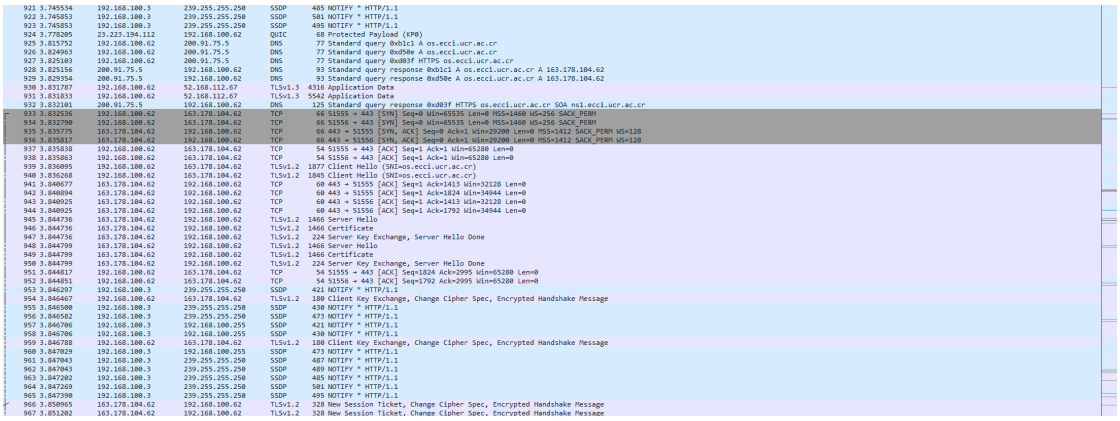
Start capturing data, visit a live web site (os.ecci.ucr.ac.cr/ci0121) using your standard Internet browser, and stop capturing data.

If you have a large amount of network traffic, the relevant data may be hidden among a lot of broadcast messages. To focus on just the key frames, you can set a display filter like this.

For the IP number enter the IP number of your client machine. Type it as shown (ip.addr==your.ip.address) in the graphic above. Then click on "Apply".

Using an approach similar to the approach in Exercise One, describe the set of frames that you captured.

- For this description think of this as a conversation – every discussion starts with a question and follows with an answer.
- For example, two of the frames will contain the DNS request for an IP address for the web site, and the DNS answer with the IP number.
- Remember that some answers may take several frames if they need to be reassembled from segmented packets.



En la imágen anterior se observan los paquetes o frames obtenidos al intentar a ingresar a la página os.ecci.ucr.ac.cr/ci0121

Entre los frames 925 y 929 ocurre la búsqueda del nombre DNS de la página. Cómo el cliente quiere entrar a os.ecci.ucr.ac.cr, consulta su dirección IP y el servidor DNS responde indicando que esta es 163.178.104.62

Entre los frames 933 y 938 se establece una conexión TCP con el servidor. Para esta conexión se realiza un three-way handshake. PAra esto el cliente envía paquetes al puerto 443 del servidor, el servidor responde con un SYN y un ACK y finalmente el cliente responde con un ACK.

Por último, entre el frame 939 y 959 se empieza el protocolo TLSv1.2, para cifrar la comunicación con el servidor. Para esto, el cliente envía un Hello al servidor, este responde con un Hello de vuelta, envía su certificado y se realiza un intercambio de claves para poder cifrar la comunicación.